xAODAnaHelpers Documentation

Release a87d6f6

ATLAS Collaboration

Apr 03, 2024

CONTENTS

The xAOD analysis framework, born out of ProofAna…or not.

Welcome to the xAODAnaHelpers wiki! This is an xAOD Analysis Framework built for Run II of ATLAS.

CHAPTER

ONE

SUPPORTED RELEASES

xAODAnaHelpers supports 21 and 22 releases. This documentation is for R22, please follow this link to see the documentation for R21.

CHAPTER

TWO

LATEST VERSION

Note: If you need to add a new release to be supported, please make sure you update the GitHub Actions ci.yml workflow file first.

We support the following releases: **AnalysisBase,25.2.4, AnalysisBase,25.2.3, AnalysisBase,25.2.2, AnalysisBase,25.2.1, AnalysisBase,25.2.0, AnalysisBase,24.2.41, AnalysisBase,24.2.40, AnalysisBase,24.2.39, AnalysisBase,24.2.38, AnalysisBase,24.2.37**,

CHAPTER

THREE

PYTHON CODE QUALITY

CHAPTER

FOUR

CONTENTS

4.1 Introduction

This package is meant to be the minimal needed to use the CP tools properly to calibrate, select, and correct the physics objects used for most physics analyses. Each step of the analysis chain is done by an EL::Algorithm which utilizes TStore to pass information to the Algos down the chain. The final product can be a TTree, histograms, or a mini xAOD (coming soon!). The philosophy adopted is not to remake the EDM or to alter it but to make minimal wrapper around CP tools to help users configure them properly and connect the full chain without much hassle. To this end, some details are hidden for the user and set automatically in the tools. As much as possible we used the same names as is shipped with the xAOD objects or the CP tools themselves. The user is not meant to learn a new EDM but rather to learn the minimal needed to start doing the fun stuff - Physics!!

4.1.1 Background

An analysis is done in steps with a `EL::Algorithm` running for each. For example, one algo is used to calibrate the jet collection, another to apply some selection cuts, and then a third algorithm can contain your analysis code to calculate something with the jets or one of the general plotting algorithms that will fill a configurable set of plots. A second jet calibrator and selector can be added to have a second jet collection at the same time. A base class for a tree has also been created with some standard branches and a way for the user to add more as well. The tree is configurable with the same options as the histograming classes - with a string of categories of interest. Each algorithm can be configured via a text file read by TEnv. Example for all are given and one can look for the “configure” function to see what options are available (also noted below). Development changes and help requests can be obtained on the e-group “atlas-sw-xAODAnaHelpersFW” or directly here on GitHub.

A word on **systematics**. When the object itself is altered (i.e. JES calibration and JES systematics) a new collection is made and put into TStore. The name of the nominal collection after calibration is set from the config file. The name of the systematically varied collection uses the same name plus the name of the systematic directly from the CP tool. The next algo in the chain using these objects needs to know which collections where created. To avoid hardcoding things and all that, when the systematics are applied a vector is created containing the names of each systematic. Downstream, algos pick up this vector then know which collections to run over. Each Algo will loop over all collections before going to the next step. If selectors are configured with limits in the number of events passing the cuts, only collections passing the cuts will be passed to algos downstream. If none pass, the next event is analyzed.

4.1.2 Related Packages

Here we list some useful packages that don't depend on xAH, but can be used along with it.

xAODDumper

This python package (developed by Giordon Stark, UChicago) contains a useful python script to dump the full content of a generic xAOD file (full breakdown of available containers and attributes, basic plots, size and memory usage). See the dedicated webpage for more info.

goodruns

This python package (made by Noel Dawe, Univeristy of Melbourne) provides a useful set of tools to handle GoodRun-Lists, e.g., merging N GRLs into a single one etc. All the details and instructions are well documented here.

4.2 Installing

4.2.1 Getting the Source

Start in a work directory

```
mkdir workdir && cd $_
```

Then clone the source

```
git clone https://github.com/UCATLAS/xAODAnaHelpers
```

Note: If you have ssh-keys set up, then you can clone over SSH instead of HTTPS:

```
git clone git@github.com:UCATLAS/xAODAnaHelpers
```

At this point, you have the FULL state of the code. You can run `git log` to view the recent changes (no more ChangeLog!).

Checking out a specific tag

You can run `git tag` to view all current tags. You can checkout a specific tag (in a detached head state):

```
cd xAODAnaHelpers
git checkout tags/XX-YY-ZZ
cd ../
```

or you can use:

```
cd xAODAnaHelpers
git checkout -b XX-YY-ZZ tags/XX-YY-ZZ
cd ../
```

which switches you from main to a branch of the given version.

4.2.2 Compiling

For all sets of instructions below, make sure you run `setupATLAS` first.

CMake-based (21.2.X)

This step requires a little extra work, but compiles significantly faster. First, inside the `workdir` directory, we'll create a build and source directory. The source directory will contain all packages we build in CMake:

```
mkdir src build
```

Then we'll set up a release inside the source:

```
cd src
asetup (RELEASE),here
```

This also sets up a `CMakeLists.txt` file in this top-level directory that searches for all packages you've checked out inside it. At this point, clone/checkout all packages you need such as xAODAnaHelpers:

```
git clone <url>/UCATLAS/xAODAnaHelpers.git
```

Next, you will need to change to your build directory that builds all your checked-out packages which is separate from your source code:

```
cd ../build
```

Note: This is inside the `workdir`, so you will have `workdir/src/xAODAnaHelpers` and `workdir/build` as paths, for example.

and then run cmake to generate our makefiles, then compile:

```
cmake ../src
make
cd ../
```

The last thing you need to do is get your environment set up correctly, so you will need to source `setup.sh` (from the top-level directory):

```
source build/*/setup.sh
```

Environment variables like `${AnalysisBase_PLATFORM}` (or `${AnalysisTop_PLATFORM}`) seem to contain the correct variable which represents the architecture of the system, e.g. `x86_64-slc6-gcc49-opt`.

4.2.3 Docker

Assuming you have docker, you can always grab the latest image for a given release (e.g. 21.2.4) like so:

```
docker pull ucatlas/xah:21.2.4-latest
docker run -it --rm ucatlas/xah:21.2.4-latest bash
```

which puts you into the docker image and xAH is precompiled and the environment is set up so you can:

- compile your package on top of xAH [using cmake, make]
- run vanilla `` `xAH_run.py` `` with a config on some ROOT files

For example, if you want to have the docker image have access to ROOT files locally on your computer, you can "mount" a folder in it like so:

```
docker run -it --rm -v /path/to/data/files:/home/atlas/data ucatlas/xah:21.2.4-latest␣
↪bash
```

and /home/atlas/data inside the docker file will map to /path/to/data/files on your computer (host).

4.3 xAH_run.py

`xAH_run.py` is the xAODAnaHelpers macro written fully in python. The goal is to make it easier for a user to spin up an analysis without (potentially) writing any C++ code at all!

4.3.1 Introduction

An analysis job is defined by a few key things: - the files to run over - where to run the code - what algorithms to run

and a few other minor features such as submission directory or how many events to run. Primarily, these three things listed above are all you need to get started. `xAH_run.py` manages all of these for you.

A configuration file, written in json or python, is used to specify what algorithms to run, and in what order. You pass in a list of files you want to run over to the script itself, as well as where to run the code. It will take care of the rest for you.

4.3.2 Getting Started

To get started, we assume you are little bit familiar with xAODAnaHelpers and AnalysisBase in general. Recall that when you compile a bunch of packages, you generate a namespace under `ROOT` that all your algorithms are loaded into so that one could create an algorithm by something like `ROOT.AlgorithmName()` and then start configuring it. In fact, this is how one normally does it within python. Namespaces are automatically linked up by something like `ROOT.Namespace.AlgorithmName()` in case you wrapped the entire algorithm in a namespace.

A simple plotting example

To get started, let's just ask a simple question: "How can I make plots of Anti-Kt, R=0.4, LC-calibrated jets?" Let's assume `xAODAnaHelpers` has already been checked out and everything is compiled. We only need to know the three key things.

What algorithms to run

We will run 2 algorithms. First is *BasicEventSelection* to filter/clean events. The second is *JetHistsAlgo* which will allow us to plot the jets we want. So start with the template JSON file:

```
[
  { "class": "BasicEventSelection",
    "configs": {
    }
  },
  {
    "class": "JetHistsAlgo",
    "configs": {
    }
  }
]
```

This gets us started. We make a list of algorithms that we want to run, this list is considered *sorted*. Each list contains a dictionary object, one which defines the `class` to run and another which defines a dictionary of configurations to pass into that algorithm. An equivalent script in python looks like

```python
from xAODAnaHelpers import Config
c = Config()

c.algorithm("BasicEventSelection", {})
c.algorithm("JetHistsAlgo", {})
```

Next, we should probably add some obvious configurations that work for us. I look up the header files of each and decide to flesh it out as below:

```
[
  { "class": "BasicEventSelection",
    "configs": {
      "m_truthLevelOnly": false,
      "m_applyGRLCut": true,
      "m_GRLxml": "$ROOTCOREBIN/data/xAODAnaHelpers/data12_8TeV.periodAllYear_DetStatus-
↪v61-pro14-02_DQDefects-00-01-00_PHYS_StandardGRL_All_Good.xml",
      "m_doPUreweighting": false,
      "m_vertexContainerName": "PrimaryVertices",
      "m_PVNTrack": 2,
      "m_name": "myBaseEventSel"
    }
  },
  {
    "class": "JetHistsAlgo",
    "configs": {
      "m_inContainerName": "AntiKt4EMTopoJets",
```

(continues on next page)

(continued from previous page)

```
      "m_detailStr": "kinematic",
      "m_name": "NoPreSel"
    }
  }
]
```

and I save this into `xah_run_example.json`. If you want more variables in your plots, add other possibilities in the detailStr field, separated by a space. Equivalently in python

```
from xAODAnaHelpers import Config
c = Config()

c.algorithm("BasicEventSelection", {"m_truthLevelOnly": False,
                                    "m_applyGRLCut": True,
                                    "m_GRLxml": "$ROOTCOREBIN/data/xAODAnaHelpers/data12_
↪8TeV.periodAllYear_DetStatus-v61-pro14-02_DQDefects-00-01-00_PHYS_StandardGRL_All_Good.
↪xml",
                                    "m_doPUreweighting": False,
                                    "m_vertexContainerName": "PrimaryVertices",
                                    "m_PVNTrack": 2,
                                    "m_name": "myBaseEventSel"})
c.algorithm("JetHistsAlgo", {"m_inContainerName": "AntiKt4EMTopoJets",
                             "m_detailStr": "kinematic",
                             "m_name": "NoPreSel"})
```

The similarity is on purpose, to make it incredibly easy to switch back and forth between the two formats.

Running the script

I pretty much have everything I need to work with. So, I run the following command

```
xAH_run.py --files file1.root file2.root --config xah_run_example.json direct
```

which will run over two ROOT files locally (`direct`), using the configuration we made. Running with the python form of the configuration is just as easy

```
xAH_run.py --files file1.root file2.root --config xah_run_example.py direct
```

How to pass command line options straight to the python config file? Let's say you wish to set a variable called `var` in your config. Then, you would have to have something like this in your config:

```
import shlex
import argparse

parser = argparse.ArgumentParser(description='Test for extra options')
parser.add_argument('-var', action='store')

# note "args" is already a variable holding the arguments passed into xAH_run.py
inner_args = parser.parse_args(shlex.split(args.extra_options))
```

Then, you can pass that argument with the `--extraOptions` flag of `xAH_run.py`:

```
xAH_run.py --files file.root --config YOURCONFIGNAME.py --extraOptions="-var 2" direct
```

We're all done! That was easy :beers: .

4.3.3 Configuring Samples

Sample configuration can be done with a python script like so

```
from xAODAnaHelpers import Config
c = Config()

c.sample(410000, foo='bar', hello='world')
c.sample("p9495", foo='bar', hello='world', b=1, c=2.0, d=True)
```

where the pattern specified in `Config::sample` will be searched for inside the name of the dataset (not the name of the file!). Specifically, we just do something like `if pattern in sample.name()` in order to flag that sample. Given this, you can make this pattern generic enough to apply a configuration to a specific p-tag, or to a specific dataset ID (DSID) as well. The above will produce the following output when running

```
[WARNING]  No matching sample found for pattern 410000
[INFO   ]  Setting sample metadata for example.sample.p9495.root
[INFO   ]       - sample.meta().setDouble(c, 2.0)
[INFO   ]       - sample.meta().setString(foo, bar)
[INFO   ]       - sample.meta().setInteger(b, 1)
[INFO   ]       - sample.meta().setString(hello, world)
[INFO   ]       - sample.meta().setBool(d, True)
```

which should make it easy for you to understand what options are being set and for which sample.

4.3.4 Configuration Details

As mentioned previous, there are multiple facets to `xAH_run.py`. The below details the configurations that are possible for the script itself, not for the algorithms you use. For details on what can be configured, look up the header files of the algorithms themselves.

For everything listed below, the script contains all this information and is self-documenting. Simply type

```
xAH_run.py -h
```

to see all the help information.

Note: The `{driver}` option tells the script where to run the code. There are lots of supported drivers and more can be added if you request it. For more information, you can type `xAH_run.py -h drivers` of available drivers.

4.3.5 API Reference

Note: If you are using a CMake-based release, or you have `argcomplete` in your python environment, you can enable automatic completion of the options. For example, running something like this:

eval "$(register-python-argcomplete xAH_run.py)"

All of the following properties can be set in a user-specific dotfile located at `${HOME}/.xah`. It is an INI file, with the *general* section used for the generic options and other sections named after sub-commands. The keys in each section are the options without the preceeding dashes.

The following example configures the Slurm driver for NERCS' Cori and records usage statistics:

```
[general]
stats=1

[slurm]
optBatchSharedFileSystem=1
optBatchWait=1
optSlurmRunTime=5:00:00
optSlurmExtraConfigLines=#SBATCH --qos=shared --tasks-per-node=1 --constraint=haswell --
↪image=centos:centos7 --export=NONE
optSlurmWrapperExec=export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/global/project/
↪projectdirs/atlas/scripts/extra_libs_180822; hostname; shifter --module=cvmfs /bin/
```

4.4 Algorithms

Each algorithm will retrieve a container from either TEvent or TStore and if an output container is specified, it is saved to the TStore object such that the following algorithms can access the container. TStore will manage the memory for the user.

4.4.1 Event Selection

(moved to BasicEventSelection.h)

4.4.2 Jet Related

HLTJetGetter

The HLTJetGetter Algorithm retrieves jets from the TrigDecisionTool. The output is a shallow copy of the trigger feature that is requested via the combination of the name of the jet collection (m_inContainerName) and the jet trigger that is selected m_triggerList. The name of the copy is chosen via the m_outContainerName. The input container name should be given without any HLT_xAOD__JetContainer prefix if the collection comes directly from the xAOD. The list of triggers must be given as a regular expression using an or (|) if more than a trigger is requested. To request all triggers, use *.

JetCalibrator

The JetCalibrator Algorithm handles the calibration, cleaning, and common uncertainties associated with Jets. It interfaces with many JetETMiss packages. The output is a shallow copy of the input jet container (i.e. a new xAOD::JetCollection). A separate shallow copy is made for each systematic variation requested, and the names of the containers are saved in a vector for downstream algorithms to use. The algorithm does not apply any selection to the jets.

The parameters to control the Algorithm are described in the header documentation: https://xaodanahelpers.readthedocs.io/en/latest/JetCalibrator.html

Calibration

Jets are calibrated using JetCalibTools. The option CalibSequence determines the calibration sequence to be applied. "Insitu" in automatically added to data samples if it is not provided, and MC jobs will fail if it is included. Config file options can be provided with CalibConfigData, CalibConfigAFII, and CalibConfigFullSim. The config variable to use is determined from metadata.

1. Sort - Sort the output jets by pT
2. SetAFII - Force AFII configurations (in case metadata isn't working)
3. InputContainer - Jet Collection to use
4. OutputContainer - Output Jet Collection for calibrated jets
5. JetAlgorithm - Input Jet Algorithm
6. OutputAlgo - Output Jet Algorithm

JES/JER Uncertainties

Outputs new containers for each systematic variation.

For JES: Uses JetUncertaintiesTool. JES is different for AFII, so m_setAFII may be required if metadata isn't working. 1. JESUncertConfig - Configuration file for JES 2. JESUncertMCType - Type of JES Uncertainty to use, MC20 or MC21

For JER: Uses JERSmearingTool 1. JERUncertConfig - Configuration file for JER 2. JERFullSys - Run full list of systematics for data and MC. Not currently recommended by JER Group (10/08/2015). 3. JERApplyNominal - Apply a nominal JER Smearing.

Cleaning

Cleaning decisions are saved as auxiliary data; No selections are made to the container. The decision is saved as an auxdecor with names like clean_passLooseBadUgly clean_pass Settings: 1. JetCleanCutLevel - Cut Level, may be LooseBad or TightBad 2. JetCleanUgly - Clean Ugly Jets (in gap region) 3. SaveAllCleanDecisions - Save all four decisions (LooseBad, LooseBadUgly, TightBad, TightBadUgly) 4. CleanParent - Apply cleaning decision to the jet's parent

JetSelector

Applies cuts on the input jet collection(s). The jets can be decorated with the decision of the cuts. A new xAOD::JetCollection can be made from the ones passing the cuts.

The JVT selection is OFF by default, it can be turned ON with `` `m_doJVT` ``. The default algorithm is NNJvt. If another algorithm is needed, use corresponding index for the enum here: https://acode-browser1.usatlas.bnl.gov/lxr/source/athena/PhysicsAnalysis/Interfaces/JetAnalysisInterfaces/JetAnalysisInterfaces/IJetJvtEfficiency.h#0022 (note: this link points to the latest r22 version, i.e. master, if a release is used, please check the corresponding enum for the given release: https://gitlab.cern.ch/atlas/athena/-/tags?search=release%2F22.2&sort=updated_desc) The default JVT working point is `` `FixedEffPt` ``, it can be changed with `` `m_WorkingPointJVT` ``. JVT is no longer recalculated in JetCalibrator, it is now done in JetSelector. It can be disabled by setting `` `m_recalculateJvtScores` `` to `` `false` ``. Furthermore, the JVT algorithm is now NNJvt (default for R22).

JetHists and JetHistsAlgo

JetHists is a class of histograms for jet variables. In the constructor one gives a name which is used as the first half of the name of all the histograms in the class and the detail string which determines which plots are made. One can declare a JetHists object in the event selection algorithm and fill the hists with every jet in a collection or pick jets by hand to fill histograms.This is done in the example event selection described below. JetHistsAlgo is a algorithm that holds a JetHists object and fills it with every jet in the user specified container.

BJetEfficiencyCorrector

This Algo wraps two flavor tagging tools - one returns the decision on if the jet is tagged or not for the given operating point (OP) and the other returns the efficiency scale factors needed for MC only. Like other Algos, "All" gives all the systematic variations. Two decorations are added to the input jets: 1. The B-Tag decision. Decoration name (default) "BTag_:math:*{OP}" 2. The B-Tag efficiency scale factor. If the jet fails the cut the inefficiency scale factor is saved. Decoration name (default) "BTAG_SF_*{OP}". This decoration is not added if the OP has not been calibrated.

The available **calibrated** operating points in the August 2015 CDI file for *fixed* b-jet efficiencies: * FixedCutBEff_60 * FixedCutBEff_70 * FixedCutBEff_77 * FixedCutBEff_85

and the *fixed* cut **un-calibrated** operating points are: * FixedCutBEff_30 * FixedCutBEff_50 * FixedCutBEff_80 * FixedCutBEff_90

Finally, the *flat* efficiency **un-calibrated** operating points are: * FlatBEff_30 * FlatBEff_40 * FlatBEff_50 * FlatBEff_60 * FlatBEff_70 * FlatBEff_77 * FlatBEff_85

These decorations can be added to a TTree via HelpTreeBase. The info switch looks for "*sfFTagFix*" and "*sfFTagFlt*" for the fixed and flat efficiency OPs respectively. "*sfFTagFix607785*" will try to add the decision and efficiency/inefficiency scale factors for the fixed 60, 70, and 85% b-jet efficiency OPs to the tree. The user must also create and run BJetEfficiencyCorrector for each OP.

Note: To those paying attention. Currently the two flavor tagging tools use different names for the same operating point. This is handled internally in xAH until the flavor tagging group harmonizes the two.

4.4.3 Muon Related

MuonCalibrator

MuonEfficiencyCorrector

Produces a container of muons decorated with efficiencies and scale factors. This container is not simply the one in input, but is a deepCopy of that. This allows flexibility in decorating muons of systematically altered containers. The output container has a configurable name and is only created for MC events. When working with systematic uncertainties, a list of systematics is passed to this algorithm *m_inputAlgoSystNames*. This algorithm supports comma separated lists as inputs, which will be considered as a unique list. The systematic names in this list will be looked for to retrieve the muon containers to decorate. Systematic variations on the decorations themselves might be unnecessary for all systematic muon containers and might only be considered for the nominal container (need of deepCopy). This is the default configuration. Otherwise the option *m_decorateWithNomOnInputSys*can be set to false. The algorithm features the option *m_sysNamesForParCont*which is a list of systematic names. For each of them, a copy of the nominal muon container is put in the store carrying the name of the systematic. These containers are only decorated with the nominal efficiencies and scale factors. The use case of this are MET systematics for which one does not want systematic variations on efficiencies, but still wants to retrieve a nominal muon container in the tree algo. Retrieving the nominal would indeed carry all the unnecessary uncertainties.

MuonSelector

MuonHists and MuonHistsAlgo

4.4.4 Electron Related

ElectronCalibrator

ElectronEfficiencyCorrector

See the MuonEfficiencyCorrector

ElectronSelector

ElectronHists and ElectronHistsAlgo

4.4.5 Tau Related

TauCalibrator

TauSelector

Similar to other selectors. Tau identification/selection algorithms working points are decorated onto the tau automatically. Tha flag m_decorateWithTracks controls the decoration of the tau track information. It can be dumped in the ntuples by specifying the trackparams flag in the tau details string of the TreeAlgo.

TauEfficiencyCorrector

The TauEfficientyCorrector provides one cumulative decoration with one SF corresponding to the combination of working points used for the tau selection/identification algorithms. Several initialisations of the algorithm are needed with different combinations in order to dump in the ntuples different combined working points. Tau trigger SFs are saved separately and wrt said cumulative working point which has to be specified in the initialisation of a new instance of the algorithm together with the trigger menu.

TauJetMatching

This algorithm is introduced to match an arbitrary and configurable collection of jets with the TauJet object. This is useful for cases where the tau seedJet (LC topo jet) is not available in xAOD but one would need to get a handle on some original jet info. The algorithm has a configurable DeltaR matching criterion and for now decorates taus with the matched jet width. It should be executed before OLR. It can be used before tau selection and after tau calibration.

4.4.6 HelperFunctions

4.4.7 HelperClasses

4.4.8 NTuple Creation

HelpTreeBase is a class (not an algorithm) that creates and fills a TTree. When adding an object type i.e. jet, muon, or event level quantities, it can be easily configured with "info switches" (See HelperClasses) that take a space separated list of categories.

4.5 Development

> **Warning:** Never push to main. Always create a new branch for changes, rebase your branch with main `git pull --rebase origin main` and use the branch for creating a pull-request to merge with main. This keeps main mergeable for everyone for all development.

4.5.1 How to Document Code

The documentation for xAODAnaHelpers uses a slightly non-trivial workflow:

1. Doxygen parses the header and source files to generate an XML tree of the code
2. breathe is a sphinx wrapper that enables us to parse the XML tree from doxygen
3. sphinx is what produces the various output formats such as html, latex, e-pub from source code comments
4. ReadTheDocs.org uses `doxygen`, `breathe`, and `sphinx` to automatically produce our documentation everytime `main` changes.

Our documentation is automatically generated for us so we will always guarantee that our documentation is up-to-date for all users.

The aim of this document is to help you get started with producing your own documentation locally to help resolve errors, typos, and make sure you're formatting it the way that you want before pushing it to our github repo.

Setting it up Locally

Locally, we are going to need doxygen to do the initial parsing. Note that out of the box without doxygen, we can parse python scripts, such as xAH_run.py *API Reference*, which are included as part of xAODAnaHelpers. However, if we wish to have all of our C++ code's documentation included, we will need doxygen to do parse it.

Doxygen

Get doxygen however you want. For Macs, we can use:

```
brew install doxygen
```

to install it. At this point, one should be able to generate the XML tree by navigating to the `docs` folder and running `doxygen` with no arguments:

```
cd docs
doxygen
```

since we provide a `Doxyfile` in the `docs` directory with the correct configurations.

Python Virtual Environment

Next, I suggest setting up a python virtual environment. Luckily, this solution is the hardest part. Most (rational) people use virtualenvwrapper to manage my python dependencies and workspace. It is assumed you already have pip.

To get the entire functionality of `venvwrapper`, we just need to grab the package and update our environment when we want to use it:

```
pip install virtualenvwrapper
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bash_profile
```

Note: Don't forget to source your profile if you're going to use the same shell:

```
source ~/.bash_profile
```

From now on, we will have commands like `mkvirtualenv`, `workon`, and `rmvirtualenv` in our shell. As a first-time user, you haven't made a virtual environment yet, so the first thing we do is make one:

```
mkvirtualenv xAH
```

This will also automatically call `workon xAH`. This is something we will always run in the future to enter the virtual environment.

Note: If you ever forget the name of the virtual environment you made, just run `workon` without any arguments. There is also tab completion.

Python Packages

Note: If you choose to use a virtual environment, enter it `workon xAH`

This is super easy. We provide a `requirements.txt` file:

```
cd docs
pip install -r requirements.txt
```

which will install all the required packages for you. As of the time of this document, this contains the following packages:

```
alabaster==0.7.12
Babel==2.9.1
beautifulsoup4==4.8.1
breathe==4.35.0
bs4==0.0.1
certifi==2023.7.22
chardet==3.0.4
docutils==0.15.2
exhale==0.2.4
idna==2.8
imagesize==1.1.0
Jinja2==3.1.3
lxml==4.9.1
MarkupSafe==2.1.3
packaging==19.2
Pygments==2.15.0
pyparsing==2.4.5
pytz==2019.3
PyYAML==6.0
requests==2.31.0
six==1.13.0
snowballstemmer==2.0.0
soupsieve==1.9.5
Sphinx==4.5.0
sphinx-argparse==0.2.5
sphinx-rtd-theme==0.4.3
sphinxcontrib-applehelp==1.0.1
sphinxcontrib-devhelp==1.0.1
sphinxcontrib-htmlhelp==2.0.0
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.2
sphinxcontrib-serializinghtml==1.1.5
urllib3==1.26.18
```

Generate Docs Locally

Now that we have `doxygen` and all of the required python packages installed, all you need to do now is process everything:

```
cd docs
make clean
doxygen
make html
open _build/html/index.html
```

and we're good to go. Sphinx provides a `Makefile` in `docs/` to make the html generation much easier to work with.

You may not always run all of these pieces each time you generate documentation. For example, if you need to make a change to the header/source files of any kind, you will need to re-run `doxygen`. In the rare case that the html generation isn't working right, you might want to run `make clean` so you start over again. If you're only changing the reStructuredText (rst) files in `docs/` you might only ever need to run `make html`. All in all, it doesn't take more than 10-15 seconds to generate the necessary documentation.

Documenting Code

In most cases, we will want to follow the reStructuredText directives and formatting for doing the code documentation. We just want to use `doxygen` + `breathe` to expose those comments to `sphinx` to parse and display correctly. In what follows, we provide a set of guidelines (really, examples) to make it easier to document our code specifically.

Note: All comments for a given class, function, variable should be prior to the given item you're adding documentation for.

If you have a question about how to do something, google it in the context of reStructuredText or ask on the mailing list. Also have a look through most of our source code and compare it to the docs to figure out how we do something.

One-Line Comments

One-line comments are very useful in cases where we do not have much to say about something, perhaps because it is a rather trivial item:

```
/** @brief generically the main name assigned to all histograms */
std::string m_name;
```

which will render as

std::string *HistogramManager*::**m_name**

generically the main name assigned to all histograms

Block Comments

Block comments are very useful in all other cases. When in doubt, you can always make a block comment with just a single line, even for a variable. The flexibility allows us to include a lot more detail and formatting such as tables and latex:

```
/**
    @brief Destructor, allows the user to delete histograms that are not being recorded.
*/
virtual ~HistogramManager();
```

which will render as

virtual *HistogramManager*::**~HistogramManager**()

Destructor, allows the user to delete histograms that are not being recorded.

Doxygen `rst` directive

To tell `doxygen` and `breathe` that a given block of text should be considered as reStructuredText, we simply need to wrap it:

```
@rst
    This is now inside a doxygen directive that tells doxygen not to parse it, so that␣
↪breathe can parse it for Sphinx.

@endrst
```

which will render as expected if we were writing it inside a standard `.rst` file. As usual, we have an example:

```
/**
    @brief This is used by any class extending to pre-define a set of histograms to book␣
↪by default.
    @rst
        .. note:: The expectation is that the user does not directly use this class but␣
↪rather inherits from it.

        We expect the user to create a new group of histograms, such as for jets::

            class JetHists : public HistogramManager
            {
              public:
                JetHists(std::string name, std::string detailStr);
                virtual ~JetHists() ;

                StatusCode initialize();
                StatusCode execute( const xAOD::JetContainer* jets, float eventWeight,␣
↪int pvLoc = -1);
                StatusCode execute( const xAOD::Jet* jet, float eventWeight, int pvLoc =␣
↪-1 );
                using HistogramManager::book; // make other overloaded version of book()␣
↪to show up in subclass
                using HistogramManager::execute; // overload
            };
```

(continues on next page)

(continued from previous page)

```
        The above example is taken from our implementation in :cpp:class:`JetHists`.

    @endrst
 */
class HistogramManager {};
```

which will render as

class **HistogramManager**

This is used by any class extending to pre-define a set of histograms to book by default.

We expect the user to create a new group of histograms, such as for jets:

```
class JetHists : public HistogramManager
{
  public:
    JetHists(std::string name, std::string detailStr);
    virtual ~JetHists() ;

    bool m_debug;
    StatusCode initialize();
    StatusCode execute( const xAOD::JetContainer  jets, float eventWeight, int␣
↪pvLoc = -1);
    StatusCode execute( const xAOD::Jet  jet, float eventWeight, int pvLoc = -1 );
    using HistogramManager::book; // make other overloaded version of book() to␣
↪show up in subclass
    using HistogramManager::execute; // overload
};
```

The above example is taken from our implementation in `JetHists`.

Note: The expectation is that the user does not directly use this class but rather inherits from it.

Subclassed by *MetHists*

For everything else...

These cover the general basics of how to document code for xAODAnaHelpers. Everything else is specific to how doxygen and Sphinx and breathe work. Most of these are well-supported with a large community, so googling is always very helpful here. Otherwise, feel free to ask on the mailing list.

4.5.2 Common Issues

Missing Metadata Information

It has been noticed that some of the latest derived xAODs have missing metadata info due to some bug in Derivation Framework. If you are incurring in a nasty crash at runtime, make sure you have set the name of the derivation property DerivationName property of BasicEventSelection. If that does not work then switched off the configuration flag:

```
UseMetadata False
```

and try again.

4.5.3 Development Workflow

Changes should be tested properly (“it compiles” is not sufficient). We use the `git rebase` workflow.

New User

This is for users who do not have write access to `UCATLAS/xAODAnaHelpers` to make branches. Instead, they fork and write their changes to their own repository and submit pull-requests.

- For *very* new users, you may want to setup SSH key access to your personal repository. To do this, follow the instructions at Generating SSH keys. So you go here: https://github.com/UCATLAS/xAODAnaHelpers/ and just click the ‘Fork’ at top right. This forks a copy into your account (`yourAccount/xAODAnaHelpers`). Next, clone it. Set the upstream:

  ```
  git clone git@github.com:yourAccount/xAODAnaHelpers
  cd xAODAnaHelpers
  git remote add upstream git@github.com:UCATLAS/xAODAnaHelpers
  ```

Note: If you do not have an ssh-key set up, you may want to use the HTTPS version of the above URL:

```
git remote add upstream https://github.com/UCATLAS/xAODAnaHelpers
```

Next, make your changes for the feature/bug/fix:

```
vim Root/JetSelector.cxx
vim Root/HelpTreeBase.cxx
git status # make sure you changed the files you want to change
git diff # make sure the changes are what you want
```

Then go ahead and commit your changes:

```
git add Root/JetSelector.cxx
git commit -m "Update with new jet calibration recommendations"
git add Root/HelpTreeBase.cxx
git commit -m "make sure that the tree dumps the systematics for new calibrations"
```

When you are ready to submit a pull-request, do the following first:

```
git fetch upstream
git rebase upstream/main
```

to make sure your code is up to date with the upstream repository.

You may want to rebase all of your changes into a single commit if you wish, and that can be done via:

```
git rebase -i HEAD~N
```

where N is the number of commits to rebase. Then you just follow the instructions. Take care not to rebase through commits that are already on main of the upstream repo. Then submit a pull-request! See https://help.github.com/articles/creating-a-pull-request/ for information on this.

After the pull-request has been merged, you can bring yourself up to date with:

```
git fetch upstream
git rebase upstream/main
```

Trusted Dev User

In this case, you have write access to this repository. Any new feature you wish to add will need to be in a new branch:

```
git checkout -b feature/newFeature
```

and then make your commits… then maybe rebase all commits into a few good ones:

```
git rebase -i HEAD~N
```

where N is the number of commits to rebase. And then rebase with main to make sure your branch is as up-to-date as possible when making the pull-request:

```
git pull --rebase origin main
```

and push your commits to the remote (setting upstream):

```
git push -u origin feature/newFeature
```

and then submit a pull request by going to xAODAnaHelpers, finding your branch, and making a pull request (usually shiny green buttons). When it's been merged, you can run:

```
git checkout main
git pull --rebase origin main
git remote prune origin
```

to delete your local copy of the branch after bringing your local copy up to date.

4.5.4 Helpful Suggestions

Updating changes

If you're on branch `myBranch` and you have commits that you want to push to the remote `origin` - the first thing you should do is always update so you're current:

```
git pull --rebase
```

will do it all. If you want more control, use:

```
git fetch
git rebase origin/main
```

or:

```
git fetch origin
git rebase origin/main myBranch
```

Note:

- `git fetch` will fetch from `origin` (see `git remote -v` for what that's defined as) by default, but you can explicitly provide a different remote repository.
- `git rebase origin/main` will rebase the current branch you are on. You can specify another branch if you want.

Changing Author Info

See https://help.github.com/articles/changing-author-info/ for more information.

Renaming lots of tags

```
git tag -l "xAODAnaHelpers*" |
cut -d "-" -f 2-4 |
while read ref
do
  git tag "$ref" "xAODAnaHelpers-$ref"
  git tag -d "xAODAnaHelpers-$ref"
  git push origin ":refs/tags/xAODAnaHelpers-$ref"
  git push --tags
done
```

Tagging and releasing on svn

Only a few people should be doing this. Encourage folks to checkout tags using git. Here are my general steps:

```
git clone git@github.com:UCATLAS/xAODAnaHelpers xAHGIT
svn co svn+ssh://svn.cern.ch/reps/atlasinst/Institutes/UChicago/xAODAnaHelpers/trunk␣
↪xAODAnaHelpers
mv xAHGIT/.git xAODAnaHelpers/.git
rm -rf xAHGIT
cd xAODAnaHelpers
```

At this point, I need to reset all changes locally (these are due to svn):

```
git reset HEAD --hard
```

and then I can look at the changes with:

```
git status
```

If I'm happy with things on the `git` side, I move over to svn side with:

```
svn status
```

and make sure any new files to add `svn add newFile` and remove `svn del oldFile` are dealt with. Then I can commit to trunk:

```
svn commit -m "Release a tag for xAH"
```

then I can copy trunk to my new tag `xx-yy-zz`:

```
svn copy svn+ssh://svn.cern.ch/reps/atlasinst/Institutes/UChicago/xAODAnaHelpers/trunk␣
↪svn+ssh://svn.cern.ch/reps/atlasinst/Institutes/UChicago/xAODAnaHelpers/tags/
↪xAODAnaHelpers-XX-YY-ZZ
```

and I should be good to go.

Skimming new test files

In order to skim some new test files, I use athena:

```
asetup 20.1.4.7,here
```

with the following python file executed via `athena.py skimming.py` on an `input.root` file

```
from AthenaCommon.AppMgr import ServiceMgr as svcMgr
import AthenaPoolCnvSvc.ReadAthenaPool

svcMgr.EventSelector.InputCollections = ['input.root']

from GaudiSequencer.PyComps import PyEvtFilter
filterseq = CfgMgr.AthSequencer("AthFilterSeq")
#filterseq += PyEvtFilter("MyFilter",evt_list=[18559067]) #will execute main sequence␣
↪only for these eventnumbers

```

(continues on next page)

(continued from previous page)

```
# Create a POOL output file with the StoreGate contents:
from OutputStreamAthenaPool.MultipleStreamManager import MSMgr
xaodStream = MSMgr.NewPoolRootStream( "StreamXAOD", "xAOD.root" )

DetDescrVersion = 'ATLAS-R2-2015-03-01-00'
include("RecExCond/AllDet_detDescr.py")

# Set up its contents:
#xaodStream.AcceptAlgs(["MyFilter"])
xaodStream.GetEventStream().TakeItemsFromInput = True
#needed to have xAOD readable outside athena (but I thought this is not needed any more?)
#theApp.CreateSvc += [ "xAODMaker::EventFormatSvc" ]
xaodStream.AddMetaDataItem(["EventBookkeeperCollection#*"])
#xaodStream.AddMetaDataItem(["xAOD::EventFormat#EventFormat"])

theApp.EvtMax=5
```

which will skim 5 events. I've found that sometimes it will not work because of the wrong geometry information specified, eg:

```
GeoModelSvc         ERROR *** *** Geometry configured through jobOptions does not match␣
↪TagInfo tags! *** ***
GeoModelSvc          INFO ** Job Option configuration:
GeoModelSvc          INFO * ATLAS tag: ATLAS-R2-2015-01-01-00
...
GeoModelSvc          INFO ** TAG INFO configuration:
GeoModelSvc          INFO * ATLAS tag: ATLAS-R2-2015-03-01-00
```

and I just have to change the line in the above python file:

```
DetDescrVersion = 'ATLAS-R2-2015-03-01-00'
```

to match the correct geometry. Ask Giordon Stark for more information if you're stuck.

Files with trigger data

On tier3, we have MC:

```
/atlas/uct3/data/users/fizisist/xAOD/mc15_13TeV.361023.Pythia8EvtGen_A14NNPDF23LO_jetjet_
↪JZ3W.merge.AOD.e3668_s2576_s2132_r6630_r6264/AOD.05403652._000001.pool.root.1
```

13 TeV data:

```
/atlas/uct3/data/users/fizisist/xAOD/data15_comm.00265573.physics_L1Calo.merge.AOD.x331_
↪m1423/data15_comm.00265573.physics_L1Calo.merge.AOD.x331_m1423._lb0179-lb0183._0001.1
```

which are useful for testing using:

```
test_multiAlgo submitDir /atlas/uct3/data/users/fizisist/xAOD <sample> <root file>
```

Decorations

As a follow-up on the discussions in yesterday's meeting, xAOD decorations can be assigned and read more efficiently defining an decorators/accessors, since auxdata requires a string-comparison search for the correct branch on every call, whereas the static accessor finds this once and then no longer has the overhead.

You can define a decorator `static SG::AuxElement::Decorator<char> dec_baseline("baseline");` which then can be used like `dec_baseline(input) = isbaseline;` and then in your code you can replace:

```
input.auxdecor<char>("baseline");
```

by:

```
dec_baseline(input);
```

These are the relevant lines of code inside SUSYObjDef _xAOD:

- https://svnweb.cern.ch/trac/atlasoff/browser/PhysicsAnalysis/SUSYPhys/SUSYTools/tags/SUSYTools-00-05-00-14/Root/SUSYObjDef_xAOD.cxx#L17
- https://svnweb.cern.ch/trac/atlasoff/browser/PhysicsAnalysis/SUSYPhys/SUSYTools/tags/SUSYTools-00-05-00-14/Root/SUSYObjDef_xAOD.cxx#L595

In SUSYToolsTester there is also an example of an AuxElement::Accessor like this:

```
static SG::AuxElement::Accessor<int> acc_truthType("truthType");
if (acc_truthType.isAvailable(*trackParticle)  ) muonTruthType = acc_
↪truthType(*trackParticle);
```

in:

- https://svnweb.cern.ch/trac/atlasoff/browser/PhysicsAnalysis/SUSYPhys/SUSYTools/tags/SUSYTools-00-05-00-14/util/SUSYToolsTester.cxx#L428

Note that the difference between accessors and decorators is that accessors are for auxdata branches in general but will not let you modify a const object, whereas Decorators permit adding information to const collections.

TString versus std::string

I've noticed that TString slows us down a little bit, so try to use std::string where possible. Code changes and equivalencies look like:

```
m_inContainerName.IsNull()
m_inContainerName.empty()

m_event->retrieve(jets, m_inContainerName.Data());
m_event->retrieve(jets, m_inContainerName);

ANA_MSG_INFO(m_inContainerName.Data());
ANA_MSG_INFO(m_inContainerName);
```

Creating a new xAH::Algorithm

If you are planning to write an *xAH::Algorithm*, there are two requirements you must abide by to fit within the xAODAnaHelpers ecosystem.

1. Only allow empty constructors, no parameters or arguments passed in.
2. Constructors must initialize an *xAH::Algorithm* instance passing in the name of itself:

```
ExampleClass :: ExampleClass() : Algorithm("ExampleClass") {}
```

The first requirement is necessary to make sure streamable code (such as EventLoop) can handle and set up your algorithms correctly when submitting jobs. The second requirement is currently necessary for xAODAnaHelpers to keep track of the number of instances of a given class that has been created. This is a registry book-keeping operation that allows users to write smarter algorithms, the kind that know how many instances of itself have been created!

Adding and Initializing Tools

This is albeit a litle bit trickier for anyone new to how Athena tools work. First, I'll provide header and source code blocks showing an example for a tool, and then I will explain the concepts.

Header File:

```
// external tools include(s):
#include "AsgTools/AnaToolHandle.h"
#include "JetCalibTools/IJetCalibrationTool.h"

class JetCalibrator : public xAH::Algorithm {

  public:
    //...

  private:
    // tools
    asg::AnaToolHandle<IJetCalibrationTool> m_JetCalibrationTool_handle{
↪"JetCalibrationTool", this};//!

}
```

Source File:

```
// tools
#include "JetCalibTools/JetCalibrationTool.h"

//...

EL::StatusCode JetCalibrator :: initialize () {
  //...

  // initialize jet calibration tool
  ANA_CHECK( m_JetCalibrationTool_handle.setProperty("JetCollection",m_jetAlgo));
  //... other setProperty() calls and other logic can be in here for tool configuration
  ANA_CHECK( m_JetCalibrationTool_handle.setProperty("OutputLevel", msg().level()));
  ANA_CHECK( m_JetCalibrationTool_handle.retrieve());
```

(continues on next page)

(continued from previous page)

```
  ANA_MSG_DEBUG("Retrieved tool: " << m_JetCalibrationTool_handle);
}

EL::StatusCode JetCalibrator :: execute () {
  //...
  m_JetCalibrationTool_handle->apply(*jet);
  //...
}

//...
```

Header Discussion

First, let's talk about the header file. You need to include the header file for the tool handles `AsgTools/AnaToolHandle.h`. As this is a templated method, you really don't to try and forward-declare this or you're gonna have a bad time. Next, you'll want to include the header file for the tool's interface class, e.g. `JetCalibTools/IJetCalibrationTool.h`.

Note: To find the correct header file for a tool's interface, look in the header file for the tool itself, e.g. `JetCalibTools/JetCalibrationTool.h`, and cross-check by looking at the classes the tool inherits from. For example, `JetTileCorrectionTool` has the `IJetTileCorrectionTool` interface class because in its header file:

```
class JetTileCorrectionTool : public virtual IJetTileCorrectionTool,
                              public asg::AsgMetadataTool
```

You might wonder why we don't just include the tool's header file in our header file. One choice is that the interface header file is smaller and easier to compile quickly. This is roughly equivalent to forward-declaring our tool, where we only include the header file for our tool in the source and put a `class ClassName;` in the header.

Lastly for the header, we make the tool handle a private member of our class. Make sure that this gets constructed with a type only by specifying the tool itself, e.g. `JetCalibrationTool`. By adding the `this` parameter, we make sure that the tool handle is indeed made as a private tool for the given algorithm.

Note: We will prefer the suffix `_handle` to refer to the fact that the variable is a tool handle in xAODAnaHelpers.

Source Discussion

Next, looking at the source code… we include the header file for our tool. Although this may not always be needed, it is good practice to help others figure out where the tool is. As of writing this documentation, the interface and the tool may be defined in different packages! Moving on, we will want to put tool initializations in `initialize()` as this will only get called on files that have events. Files without events will not create a tool, conserving memory and processing power.

If you need to use/retrieve a tool created in another class, you will need to have the same name in both places for the ToolHandle to find it, and you need to make sure the tool isn't made private (don't use `this` for the second parameter for initializatiton).

If you don't set a name for the tool, only a type, the default name is the type. For example:

```
asg::AnaToolHandle<IJetCalibrationTool> test_handle{"JetCalibrationTool", this};
ANA_MSG_INFO(test_handle.name()); // will output "JetCalibrationTool"
```

Note: In ASG Software, tools created through AnaToolHandle can be found in the ToolStore via preprending `ToolSvc.` to the name of the tool:

```
asg::ToolStore::contains<Trig::TrigDecisionTool>("ToolSvc."+ m_trigDecTool_handle.name())
```

This is a slight gotcha that will trip up people. Because of this, xAODAnaHelpers prefers the convention of using `isUserConfigured()` instead as this doesn't need the additional `ToolSvc.` prepended to the tool name to look it up!

If it has `isUserConfigured()==0` (e.g. "not configured before": a tool with that type and name has not been created), then let's go ahead and configure it with `setProperty()`! One thing you should **always** do is set the output level of the tool `OutputLevel`. It is usually best to set it to the same output level that the algorithm is configured to `msg().level()` and is probably the safest prescription.

Note: For setting properties or managing tools through the tool handle, you access functions through the dot (`.`) operator. For using the tool, you access functions through the arrow (`->`) operator.

If a tool handle has been configured previously, but not initialized (such as using a tool handle of the same type and name as a previously created tool handle), then all `setProperty()` calls will be further ignored. I can demonstrate this with a neat code example:

```
// set up the players
asg::AnaToolHandle<IJetCalibrationTool> alice{"JetCalibrationTool/MyName"};
asg::AnaToolHandle<IJetCalibrationTool> bob  {"JetCalibrationTool/MyName"};

// set configurations on the first handle
ANA_CHECK(alice.setProperty("p1", v1)); // will set the underlying tool MyName->p1 = v1
ANA_CHECK(alice.setProperty("p2", v2)); // will set the underlying tool MyName->p2 = v2
ANA_CHECK(alice.retrieve()); // creates the tool MyName

ANA_CHECK(bob.setProperty("p1", v9)); // will be ignored as bob.isUserConfigured() == 1␣
↪[alice owns the tool]
ANA_CHECK(bob.setProperty("p3", v3)); // will be ignored as bob.isUserConfigured() == 1␣
↪[alice owns the tool]
ANA_CHECK(bob.retrieve()); // retrieves the existing tool MyName
```

AnaToolHandle will also not let us change the configuration of a previously initialized tool (one which `handle.retrieve()` has been called on). In this case, the tool has been `initialized`. Continuing the code example from before, if you were annoyed that the `setProperty()` calls were ignored, you might try setting it again on `alice`:

```
ANA_CHECK(alice.setProperty("p3", v3)); // will crash as alice.isInitialized() == 1␣
↪[alice already created its tool]
```

Finally, we `retrieve()` (`initialize()`) the tool of the given type and name from the tool store. `retreive()` and `initialize()` are synonyms and will almost always create a new tool. The only two exceptions are if the user configured the tool (`isUserConfigured()==1`) or if another ToolHandle created the tool as a public tool and holds on to it. But that's it, the memory will be managed for you and you do not need to delete it or do anything else but use it in your code!

Note: Did you get a bus error, segfault, or abort in the code because of the tools? If so, it is most likely due to a typo in the tool's header file. Please identify which tool causes the error and file an issue so we can inform the tool developers that their tool needs to be fixed. In the meantime, this can be fixed using a macro:

```
ANA_CHECK( ASG_MAKE_ANA_TOOL(m_JVT_tool_handle, CP::JetJvtEfficiency));
```

An example of a reported issue for the above tool is here: https://its.cern.ch/jira/browse/ATLASG-1214.

Check if a tool exists and reuse it (Trig::TrigDecisionTool)

The TrigDecisionTool is a special case that needs attention. This tool is unique in that the templaed methods require us to use the tool as its own interface. It is also a singleton which means it will complain heavily if it detects more than one instance of itself. How do we deal with this in xAODAnaHelpers?

Header File:

```
// external tools include(s):
#include "AsgTools/AnaToolHandle.h"
#include "TrigDecisionTool/TrigDecisionTool.h"

class MyAlgorithm : public xAH::Algorithm {

  public:
    /** @brief trigDecTool name for configurability if name is not default.  If empty,␣
↪use the default name. If not empty, change the name. */
    std::string m_trigDecTool_name{""};

  private:
    /** @brief Trigger decision tool.

    If you need to use a TDT that was previously created before this algorithm with a␣
↪different name, set the name in m_trigDecTool_name.
    */
    asg::AnaToolHandle<Trig::TrigDecisionTool>     m_trigDecTool_handle{
↪"Trig::TrigDecisionTool"};                            //!
};
```

Source File:

```
EL::StatusCode MyAlgorithm :: initialize(){

  // Grab the TrigDecTool from the ToolStore
  if(!m_trigDecTool_handle.isUserConfigured()){
    ANA_MSG_FATAL("A configured " << m_trigDecTool_handle.typeAndName() << " must have␣
↪been previously created! Double-check the name of the tool." );
    return EL::StatusCode::FAILURE;
  }
  ANA_CHECK( m_trigDecTool_handle.retrieve());
  ANA_MSG_DEBUG("Retrieved tool: " << m_trigDecTool_handle);

}
```

The above is an example of how one designs an algorithm that requires the TrigDecisionTool and will crash if it cannot find it. It also prints the name of the tool it is using to make it much easier for a user to debug. By convention in xAODAnaHelpers, `BasicEventSelection::m_trigDecTool_name` will default to `"xAH::TrigDecTool"`. All algorithms follow this default if they need the trigger decision tool. If there is an external algorithm that creates it and you want xAODAnaHelpers to pick it up instead of creating one, this can be done by setting `m_trigDecTool_name` to a non-empty value and you're good to go. For example, *BasicEventSelection* will create a trigger decision tool if it does not exist:

```
ANA_CHECK( m_trigDecTool_handle.setProperty( "ConfigTool", m_trigConfTool_handle ));
ANA_CHECK( m_trigDecTool_handle.setProperty( "TrigDecisionKey", "xTrigDecision" ));
ANA_CHECK( m_trigDecTool_handle.setProperty( "OutputLevel", msg().level() ));
ANA_CHECK( m_trigDecTool_handle.retrieve());
ANA_MSG_DEBUG("Retrieved tool: " << m_trigDecTool_handle);
```

so that if such a tool already was created before `BasicEventSelection` tries to create it, it will retrieve it (and the `setProperty()` calls will be ignored). If it has not been created/configured before, it will configure and then create the tool. No extra logic needed on the users' part.

4.6 Community

4.6.1 Tutorials

Brian Tuan's Tutorial

Introduction

This tutorial will introduce a new user to analysis of xAOD's on Tier 3 using the RootCore framework. We will first produce a slimmed tree from Monte Carlo on the GRID, then we will retrieve the file to the local disk and produce a weighted histogram of pT from the slimmed tree. The full source code of the tutorial is here, for those looking to get a quick start.

Setup

First, connect to the UC Tier 3 server with X-11 window forwarding enabled (so that you may use ROOT interactively later):

```
ssh -Y btuan@uct3.uchicago.edu
```

Make sure that the following lines are in your ~/.bash_profile file, which is a script that runs each time you log into the shell. These lines set up the ATLAS software environment where RootCore, among other tools, is located and depends upon:

```
# Setup ATLAS Environment
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
alias setupATLAS='source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh'
export ALRB_localConfigDir=$HOME/localConfig
source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh
```

Now, setup the RootCore environment and build it. We will use 2.3.21, which, at time of writing is the latest supported release by xAODAnaHelpers. You'll need to perform this step each time you login to the shell:

```
cd ~/work
rcSetup Base,2.3.21
```

To see other available versions of RootCore, type:

```
rcSetup -r
```

Make a work directory and checkout the xAODAnaHelpers package. This package contains a few useful tools for any analysis: e.g. event selection, jet cleaning, jet calibration etc. After checking out all the packages, return to the directory that has the "RootCore" folder – probably your home directory – and recompile, just to double check that everything works. This may take a while, be patient.

In general, always check to make sure that your code compiles properly after any changes with `rc compile`. Any time package dependencies are updated, be sure to run `rc find_packages` as well. In addition, if compilation for one of your packages fails, and all code errors seem to be fixed, but RootCore still refuses to compile your package, try running `rc clean` then `rc find_packages && rc compile`. This will clean out all of the old files that may have been improperly compiled.

It is good practice to repeat this procedure any time you change versions of any packages, RootCore included (though recompiling everything will of course take a while):

```
mkdir work && cd work
git clone https://github.com/UCATLAS/xAODAnaHelpers.git
python xAODAnaHelpers/scripts/checkoutASGtags.py 2.3.21
rc find_packages && rc compile
```

RootCore comes with a script that allows us to easily create a skeleton for your analysis. Do so:

```
. $ROOTCOREDIR/scripts/make_skeleton.sh Tutorial
```

Make a directory called run. This is where your script will be located:

```
cd work/Tutorial && mkdir run && cd run
touch makeSlimmedTree.cxx
```

Code and Data

Warning: setConfig and getConfig are eliminated as `ROOT::TEnv` support is now deprecated.

Place the following code in your ``**makeSlimmedTree.cxx**``. Skim through it to familiarize yourself with the sequence of the analysis:

```
void makeSlimmedTree (const std::string& submitDir)
{
    //===========================================
    // FOR ROOT6 WE DO NOT PUT THIS LINE
    // (ROOT6 uses Cling instead of CINT)
    // Load the libraries for all packages
    // gROOT->Macro("$ROOTCOREDIR/scripts/load_packages.C");
    // Instead on command line do:
    // > root -l -b -q '$ROOTCOREDIR/scripts/load_packages.C' 'makeSlimmedTree.cxx (
↪"submitDir")'
    // The above works for ROOT6 and ROOT5
    //==========================================

    bool f_grid = false;    // decide if we use the GRID to run our analysis. default␣
↪false.
```

(continues on next page)

(continued from previous page)

```
    // Set up the job for xAOD access:
    xAOD::Init().ignore();


    /*#####################################################
    ############# SampleHandler Configuation ##############
    #####################################################*/

    // create a new sample handler to describe the data files we use.
    SH::SampleHandler sh;

    // Use only one of the following three methods to scan for files with SampleHandler

    // (1) use SampleHandler with DQ2 to obtain the desired dataset
    // SH::scanDQ2 (sh, "data15_13TeV.00267638.physics_Main.merge.AOD.r6818_p2358/");

    // (2) use SampleHandler with a dataset list to obtain the desired dataset
    const std::string inputFilePath = gSystem->ExpandPathName("$ROOTCOREBIN/data/
↪Tutorial/inDSShort.txt");
    SH::readFileList (sh, "sample", inputFilePath );

    // (3) use SampleHandler to scan all of the subdirectories of a directory for␣
↪particular MC single file:
    // const char* inputFilePath = gSystem->ExpandPathName ("/export/t3data3/fizisist/");
    // SH::DiskListLocal list (inputFilePath);
    // SH::scanDir(sh, list);

    // set the name of the tree in our files. in the xAOD the TTree containing the EDM␣
↪containers is "CollectionTree"
    sh.setMetaString ("nc_tree", "CollectionTree");
    sh.setMetaString("nc_grid_filter", "*");    //Data files on grid to not end in .root

    // print out the samples we found
    sh.print ();


    /*#####################################################
    ################## Job Configuration ##################
    #####################################################*/

    // this is the basic description of our job
    EL::Job job;
    job.sampleHandler (sh); // use SampleHandler in this job

    // job.options()->setDouble (EL::Job::optMaxEvents, 5000); // for testing purposes,␣
↪limit to run over the first 500 events only!

    // To automatically delete submitDir
    job.options()->setDouble(EL::Job::optRemoveSubmitDir, 1);

    // For Trigger
```

(continues on next page)

(continued from previous page)

```
    job.options()->setString( EL::Job::optXaodAccessMode, EL::Job::optXaodAccessMode_
↪branch );

    // Use TTreeCache to precache data files to speed up analysis
    job.options()->setDouble (EL::Job::optCacheSize, 10*1024*1024);
    job.options()->setDouble (EL::Job::optCacheLearnEntries, 20);


 /*#####################################################
    ############### Output Configuration #################
    #####################################################*/

    std::string outputName;
    std::string userName = "btuan";

    // if running on GRID, make sure no two runs have the same output name. tag as␣
↪necessary
    std::string outputTag = ".v1/";

    if(f_grid) // follow GRID naming conventions
        outputName = "user."+userName+".%in:name[1]%.%in:name[2]%.%in:name[3]%
↪"+outputTag;
    else
        outputName = "%in:name%"+outputTag;


    /*#####################################################
    ############## Algorithm Configuration ###############
    #####################################################*/

    // basic event selection : GRL, event cleaning, NPV
    BasicEventSelection* baseEventSel = new BasicEventSelection();
    baseEventSel->setName("baseEventSel")->setConfig( "$ROOTCOREBIN/data/Tutorial/
↪baseEventSel.config" );

    // jet calibrator
    std::string systName = "None";
    float systVal = 0;
    JetCalibrator* jetCalib = new JetCalibrator();
    jetCalib->setName( "jetCalib" )->setConfig( "$ROOTCOREBIN/data/Tutorial/jetCalib_
↪AntiKt4EMTopo.config")->setSyst( systName, systVal );

    // jet selector
    JetSelector* jetSelect = new JetSelector();
    jetSelect->setName( "jetSelect" )->setConfig( "$ROOTCOREBIN/data/Tutorial/jetSelect.
↪config" );

    // tree output
    TreeAlgo* outTree = new TreeAlgo();
    outTree->setName( "outTree" )->setConfig( "$ROOTCOREBIN/data/Tutorial/outTree.config
↪" );
```

(continues on next page)

(continued from previous page)

```
    // add algorithms to analysis
    job.algsAdd (baseEventSel);
    job.algsAdd (jetCalib);
    job.algsAdd (jetSelect);
    job.algsAdd (outTree);


    /*#####################################################
    ################$ Initialize Driver #####$$$############
    #####################################################*/

    // here, we choose which driver to use with the boolean set earlier
    if (f_grid){ // run using the GRID driver
        EL::PrunDriver driver;

        driver.options()->setString("nc_outputSampleName", outputName);
        driver.options()->setDouble(EL::Job::optGridNFilesPerJob, 2);
        // driver.options()->setDouble(EL::Job::optGridMemory, 10240); //10 GB

        driver.submitOnly(job, submitDir); // submitOnly runs job without opening␣
↪monitoring loop
    }
    else { // run using a direct driver
        EL::DirectDriver driver;
        driver.options()->setString("nc_outputSampleName", outputName);
        driver.submit (job, submitDir);
    }

}
```

Update the package dependencies on the line ``**PACKAGE_DEP**`` in `cmt/Makefile.RootCore` to include xAODAnaHelpers:

```
PACKAGE_DEP = xAODAnaHelpers
```

Later on, in more driven analyses, you may find yourself adding the EventLoop and EventLoopGrid packages to the dependencies. The xAODAnaHelpers package takes care of all of the event looping for you in this case, so the only dependency is upon that package.

Since we use the DQ2 SampleHandler to obtain the datasets, you will need to set up a valid VOMS proxy (which you will need anyways to submit the job to the grid) and a DQ2 client if you want to run the job locally. You can also use the XRootD protocol with FAX to obtain the samples. The code for this is commented out in the ``**makeSlimmedTree.cxx**`` code. The gist of this is the following (on the command line):

```
voms-proxy-init -voms altas
localSetupFAX
fax-get-gLFNs data15_13TeV.00267638.physics_Main.merge.AOD.r6818_p2358 > inDS.txt
localSetupPandaClient
```

Make a directory ``**Tutorial/data**``. This will be where we put all of the data and configuration files for our package, and for xAODAnaHelpers. Once you run find_packages and compile with RootCore, you will be able to refer to this data directory with the ``**$ROOTCOREBIN**`` path variable, which is particularly useful when you have to generalize your code to run on batch machines, grid, etc:

```
mkdir ~/work/Tutorial/data/
mv inDS.txt ~/work/Tutorial/data/
```

Configuration of xAODAnaHelpers Algorithms

As mentioned earlier, xAODAnaHelpers provides a series of algorithms that are chained in sequence to provide the desired output. The input and output containers for each of the algorithms in sequence are configured by .config files – one for each algorithm. Create the following configuration files (as set in the ROOT macro in the run directory) in the data directory:

```
touch ~/work/Tutorial/data/baseEventSel.config
touch ~/work/Tutorial/data/jetCalib_AntiKt4EMTopo.config
touch ~/work/Tutorial/data/jetSelect.config
touch ~/work/Tutorial/data/outTree.config
```

Each of these configuration files will set the options for a separate part of the analysis. Include the following in each file. At present, there is no centralized documentation for all of xAODAnaHelpers – there is some on the GitHub wiki – but to view the availability of configuration options for each xAODAnaHelpers algorithm, view the header file and source code.

baseEventSel.config:

```
Debug                      False
ApplyGRL                   False
GRL                        $ROOTCOREBIN/data/Tutorial/data15_13TeV.periodAllYear_
↪DetStatus-v63-pro18-01_DQDefects-00-01-02_PHYS_StandardGRL_All_Good.xml
DoPileupReweighting        False
VertexContainer            PrimaryVertices
NTrackForPrimaryVertex     2
TruthLevelOnly             False
#Trigger                    L1_RD0_FILLED
#Trigger                    L1_.*
#Trigger                    L1_MBTS_1_1
#Trigger                    .*
Trigger                    .*_MBTS_1_1|.*_RD0_FILLED|L1_J[0-9]*|HLT_j[0-9]*|HLT_noalg_j[0-
↪9]*|L1_XE[0-9]*|HLT_XE[0-9]*|HLT_noalg_XE[0-9]*
StoreTrigDecision          True
CutOnTrigger               False
StorePassAny               True
StorePassL1                True
StorePassHLT               True
StoreTrigKeys              True
UseMetaData                False
## last option must be followed by a new line ##
```

jetCalib_AntiKt4EMTopo.config:

```
Debug                False
InputContainer       AntiKt4EMTopoJets
JetAlgorithm         AntiKt4EMTopo
#
SaveAllCleanDecisions True
```

(continues on next page)

(continued from previous page)

```
#
OutputContainer   Jets_Calib
OutputAlgo        Jets_Calib_Algo
configNameAFII    JES_Full2012dataset_AFII_January2014.config
configNameFullSim JES_MC15Prerecommendation_April2015.config
configNameData    JES_MC15Prerecommendation_April2015.config
#configNameData    JES_Full2012dataset_May2014.config
CalibSequence     JetArea_Residual_Origin_EtaJES_GSC
#
## last option must be followed by a new line ##
```

jetSelect.config:

```
Debug                   False
InputContainer          Jets_Calib
InputAlgo               Jets_Calib_Algo
OutputContainer         SignalJets
OutputAlgo              SignalJets_Algo
DecorateSelectedObjects False
CreateSelectedContainer True
# save multiple cleaning decisions instead of applying the cleaning
CleanJets               False
#
pTMin                   20e3
PassMin                 1
Sort                    True
UseCutFlow              True
# pT cut is > JVF recommended pT cut - to be added ... or JVT?
DoJVF                                   False
pTMaxJVF                        50e3
etaMaxJVF                       2.4
JVFCut                                  0.5
## last option must be followed by a new line ##
```

outTree.config:

```
Debug                                False
EventDetailStr              "pileup"
TrigDetailStr               True
JetDetailStr                "kinematic substructure rapidity energy scales truth␣
↪LeadingJets"
#JetDetailStr               "kinematic"
JetContainerName    SignalJets
SameHistsOutDir             False
## last option must be followed by a new line ##
```

Almost there! All that's left to do is copy the requisite files into the locations specified by our makeSlimmedTrees.cxx script.

The atlasstyle package is located here. Download and unzip the package, then place it in the run/ directory. Full support for ATLAS Style will be incorporated soon.

Copy the desired GRL to the data/ folder. The Good Runs List is used to specify which events will be kept and which events will be discarded, based on LHC and ATLAS operations (e.g. bad luminosity block, etc.). The minutiae are located here.

Note: Always use the most updated GRL, and use the same GRL for your luminosity calculations as you do your event selections. This tutorial uses the following GRL.

Plotting

Here is a "quick and dirty" plotting macro to be placed in the ``**run/**`` folder for a plot. An example better integrating AtlasStyle is in the works and should be updated soon:

```
/**
    * Plotter.cxx -- simple plotter for slimmed trees
    *
    * @author   Brian Tuan
    * @contact  brian.tuan@cern.ch
    * @date     21 July 2015
    *
    * Run on the command line by:
    *   root -l '$ROOTCOREDIR/scripts/load_packages.C' '$ROOTCOREBIN/data/Tutorial/
↪atlasstyle/AtlasStyle.C' 'Plotter.cxx( filePath )'
    *           If no argument indicated, Plotter will default to $PWD/submitDir/data-
↪tree/sample.root
    **/

    #include "atlasstyle/AtlasUtils.h"
    #include "atlasstyle/AtlasLabels.h"
    #include "atlasstyle/AtlasStyle.h"

    #include "atlasstyle/AtlasUtils.C"
    #include "atlasstyle/AtlasLabels.C"

    #include "TCanvas.h"
    #include "TFile.h"
    #include "TROOT.h"
    #include "TH1F.h"
    #include "TRandom.h"
    #include "TGraphErrors.h"

    void Plotter (const std::string filePath = "submitDir/data-tree/sample.root"){

        SetAtlasStyle();

        // TFile* f_input = new TFile(filePath.c_str(), "READ", "file", 1);
        TFile* f_input = new TFile("/afs/cern.ch/user/b/btuan/work/Tutorial/run/
↪submitDir/data-tree/sample.root", "READ", "file", 1);
        if( !f_input ){ std::cout<<"File not found! Exiting..."<<std::endl; return; }

        TTree* t_tree = (TTree*)f_input->Get("outTree"); // argument must be exact name␣
↪of tree

        // Create a TTreeReader named "MyTree" from the given TDirectory.
```

(continues on next page)

(continued from previous page)

```
        // The TTreeReader gives access to the TTree to the TTreeReaderValue and
        // TTreeReaderArray objects. It knows the current entry number and knows
        // how to iterate through the TTree.
        TTreeReader reader("outTree", f_input);

        // Read a single float value in each tree entry:
        TTreeReaderValue<int> evNum(reader, "eventNumber");
        TTreeReaderValue<float> weight(reader, "mcEventWeight"); // weight defaults to 1␣
↪if data

        // Read a vector from in each of the tree entries:
        TTreeReaderValue<std::vector<float>> jetPt(reader, "jet_pt");
        TTreeReaderValue<std::vector<float>> jetEMPt(reader, "jet_emScalePt");
        TTreeReaderValue<std::vector<float>> jetPUPt(reader, "jet_pileupScalePt");
        TTreeReaderValue<std::vector<float>> jetPhi(reader, "jet_phi");
        TTreeReaderValue<std::vector<float>> jetEta(reader, "jet_eta");
        TTreeReaderValue<std::vector<float>> jetWidth(reader, "jet_Width");

        // Now iterate through the TTree entries and fill a histogram.
        TH1F* h_jetPt = new TH1F("h_jetPt", "pt", 100, 0., 250.);
        h_jetPt->SetTitle("AntiKt4 Pt");
        h_jetPt->SetXTitle("Pt (GeV)");
        h_jetPt->SetYTitle("nEvents");

        while( reader.Next() ) { // dummy iterator just to keep count!
            if (reader.GetEntryStatus() != TTreeReader::kEntryValid ){
                switch (reader.GetEntryStatus()) {
                    case TTreeReader::kEntryValid:
                        // All good! Nothing to worry about.
                        break;
                    case TTreeReader::kEntryNotLoaded:
                        std::cerr << "Error: TTreeReader has not loaded any data yet!\n";
                        break;
                    case TTreeReader::kEntryNoTree:
                        std::cerr << "Error: TTreeReader cannot find a tree named \
↪"outTree\"!\n";
                        break;
                    case TTreeReader::kEntryNotFound:
                        // Can't really happen as TTreeReader::Next() knows when to stop.
                        std::cerr << "Error: The entry number doe not exist\n";
                        break;
                    case TTreeReader::kEntryChainSetupError:
                        std::cerr << "Error: TTreeReader cannot access a chain element,␣
↪e.g. file without the tree\n";
                        break;
                    case TTreeReader::kEntryChainFileError:
                        std::cerr << "Error: TTreeReader cannot open a chain element, e.
↪g. missing file\n";
                        break;
                    case TTreeReader::kEntryDictionaryError:
                        std::cerr << "Error: TTreeReader cannot find the dictionary for␣
↪some data\n";
```

(continues on next page)

(continued from previous page)

```
                    break;
                }
                return false;
            }

            // Access the jetPt as an array, whether the TTree stores this as
            // a std::vector, std::list, TClonesArray or Jet* C-style array, with
            // fixed or variable array size.
            if ((*jetPt).size() < 2 || (*jetPt)[0] < 100) //at least two jets, leading␣
↪jet > 100 GeV
                continue;

            // Access the array of taus.
            float currentWeight = *weight;
            for (int iJets = 0, nJets = (*jetPt).size(); iJets < nJets; ++iJets)
                h_jetPt->Fill( (*jetPt)[iJets] , currentWeight);
        }


        TCanvas* c1 = new TCanvas("c1","AntiKt4EMTopoJets pT",50,50,600,600);
        TPad* thePad = (TPad*)c1->cd();

        myText(       0.3,  0.85, 1, "#sqrt{s}= 14 TeV");
        myText(       0.57, 0.85, 1, "|#eta_{jet}|<0.5");
        myMarkerText( 0.55, 0.75, 1, 20, "Data 2009",1.3);
        myBoxText(    0.55, 0.67, 0.05, 5, "NLO QCD");

        ATLASLabel(0.2,0.2,"Preliminary");

        h_jetPt->Draw();

        c1->Print("Output.eps");
        c1->Print("Output.png");
        c1->Print("Output.pdf");
    }
```

Tips & Tricks

Here are a few tips and tricks that should help you avoid most errors, and prove as good practice for any analysis with AnaHelpers.

Maintaining xAODAnaHelpers

- xAODAnaHelpers is now hosted on GitHub! This means two things: first, there is a basic documentation available (*xAODAnaHelpers*) as reference. The documentation is still in progress, but what's already there should help you figure out what's going on with the package. Second, the development page (*Latest Version*) will contain information about the latest analysis base release that xAH has been tested to be compatible with.
- Should you find any errors with xAODAnaHelpers code – which should be a very rare occurrence, but programmers are still human – you can immediately report the issue to the entire xAH team in GitHub issues. Issues are tracked publicly, so you can stay posted about the resolution of your issue.
- Updating the framework should be as simple as calling `git pull !https://github.com/xAODAnaHelpers` from within the `xAODAnaHelpers` directory. Then, to make sure all the latest Good Runs Lists (GRLs) and configuration information are updated as well, run `python xAODAnaHelpers/scripts/checkoutASGtags.py $ABver` where $ABver is the version of your analysis base release, in this case `2.3.21`. The following lines of code should accomplish the same result automatically:

```
if [ -d $ROOTCOREBIN/../xAODAnaHelpers ]
    then cd $ROOTCOREBIN/../ python xAODAnaHelpers/scripts/checkoutASGtags.py $(echo
↪$ROOTCOREDIR \| sed 's/\\/cvmfs\\/atlas\\.cern\\.ch\\/repo\\/sw\\/ASG\\/AnalysisBase\\/
↪\\([0-9]\*[.][0-9]\*[.][0-9]\*\\).\*/\\1 /');
fi
```

This framework will automatically scale everything in to the GeV range for you, but the xAOD format lists all energies in MeV.

Monitoring loop with pbook show() retry() kill() bigpanda / loadpackages:

```
EL::Driver::wait()
```

Debug True gives a verbose mode.

4.6.2 Email List

For other inquiries in which you don't have a Github account or prefer to ask a question to the community at large, please feel free to both subscribe and email to atlas-sw-xAODAnaHelpersFW.

4.6.3 Who uses us?

The following list are packages / analyses searches that depend on xAH. We thank them for their hard work and hope they continue to use us!

- ttH->multileptonic final state
  - HTop - former HSG8 group
- dijet
- multijet
- hh->4b
- VBF + invisible
- g->tt susy multi-b-jet
- Jet/MET

- Jet Cleaning
- jet inputs to reconstruction
- Punch-through studies
- Multijet balance
- Standard Model inclusive jet cross section
- Voronoi Area Pileup Subtraction
- Trigger-Level Analysis
- Jet trigger group performance studies
- Dijet+ISR Analysis
- SM Full Run 2 Z+HF analysis
- SM Full Run 2 W+jets analysis

4.7 xAH FAQ

This is a list of Frequently Asked Questions about xAODAnaHelpers and analysis software. Feel free to suggest new entries!

4.7.1 How do I…

… submit a grid (prun) job? Start with a minimal environment:

```
lsetup panda
```

and a minimal configuration script:

```
from xAH_config import xAH_config
c = xAH_config()
c.algorithm("BasicEventSelection", {"m_name": "test", "m_useMetaData": False})
```

Then we can submit a job:

```
xAH_run.py --inputRucio --files "user.lgagnon.370150.Gtt.DAOD_SUSY10.e4049_s2608_
↪r6765_r6282_p2411_tag_10_v1_output_xAOD.root" \
--config=test.py prun --optGridMergeOutput=1 \
--optGridNFilesPerJob=1.0 --optGridOutputSampleName=user.gstark.test
```

… submit `xAH_run` jobs with production privileges? You can use `--optSubmitFlags="--official"` or `--optOfficial=1` (?):

```
   xAH_run.py --files MultijetAlgo/scripts/grid_samples_EXOT1_data.txt --inputList \
   --config MultijetAlgo/scripts/config_MultijetAlgo.py -f --inputDQ2 prun \
--optGridOutputSampleName="group.phys-exotics.%in:name[1]%.%in:name[2]%.%in:name[3]
↪%.v0.1_20150921/" \
--optSubmitFlags="--official"
```

… use `AnaToolHandle` for ASG CP tools? Unfortunately there's no much documentation out there, so everything written here comes from direct email question to ASG fellows, or looking at the source code

1. Make the tool handle as a member of a xAH algorithm (NB: remember to set the **interface** class of the CP tool. Just prepend an `I` to the tool type):

```
class MyAlgo : public xAH::Algorithm
{
  ...
  private:
    ...
    asg::AnaToolHandle<IMyToolType>   m_mytool_handle; //!

}
```

2. In the xAH algorithm initialisation list, call the tool handle constructor. The argument of the constructor must be a string with the tool type and tool name separated by a slash `/` . In general, the tool name in the constructor can be just a dummy string, as it can be changed afterwards:

```
MyAlgo :: MyAlgo (std::string className) :
  ...
  m_mytool_handle("MyToolType/MyToolName"),
  ...
 {
   ...
 }
```

3. In some cases the name of the tool has to be different than the one set in the constructor. E.g., for the efficiency correctors, the tool names must depend on the configuration of the algorithm, which is set only **after** the initialisation list is executed. In such situations, the name of the tool can be modified (typically this would happen in `EL::initialize()`) with:

```
EL::StatusCode BasicEventSelection :: initialize ()
{
  ...
  m_mytool_handle.make("MyToolType/MyToolNewName");
  ...
}
```

4. In `EL::initialize()`, set the properties and initialise the tool handle. After `m_mytool_handle.initialize()` has been called, it will effectively behave like a pointer to the tool itself:

```
EL::StatusCode BasicEventSelection :: initialize ()
{
  ...
  m_mytool_handle.make("MyToolType/MyToolNewName");
  m_mytool_handle.SetProperty(...);
  m_mytool_handle.initialize();
  ...
}
```

5. In the algorithm, use the tool associated to the handle via calls like `m_mytool_handle->doStuff()`.

6. The tool associated to the handle will be automatically destroyed when appropriate. Hence, no need to call `delete` anywhere.

If the same tool (identified by its name) needs to be used in another xAH algorithm downstream, just declare a tool handle member with the same `IMyToolType`, call its constructor in the initialisation list and (if needed) change its tool name with `make()`. Then in `EL::initialize()` simply call `m_mytool_handle.initialize()`, without setting any property. It will automagically get the pointer to the correct tool from a registry, and all the tool properties will be preserved from the previous initialisation.

4.7.2 SLC6 vs SLC7

If you're running into issues with grid submission because of checks for SLC7-compatible machines in *xAH_run.py* preventing you from doing so, then you can either:

- ssh into lxplus SLC7 (`lxplus.cern.ch`)
- run in a containerized SLC7 environment (`setupATLAS -c slc6`)

If you think this message is happening in error, file an issue giving us the output from the following commands:

- `lsb_release -d`
- `printenv | grep _PLATFORM`

4.8 API Reference

4.8.1 Getting Objects

HLT Jet Getter

class **HLTJetGetter** : public xAH::*Algorithm*

Public Functions

HLTJetGetter()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_triggerList** = ".*"

List of triggers whose features will be extracted from TDT.

std::string **m_inContainerName** = ""

input container name, WITHOUT the HLT_xAOD__JetContainer_ prefix

std::string **m_outContainerName** = ""

output container name

Private Members

asg::AnaToolHandle<Trig::TrigDecisionTool> **m_trigDecTool_handle** = {"Trig::TrigDecisionTool/TrigDecisionTool"}

TrigConf::xAODConfigTool ***m_trigConfTool** = nullptr

bool **m_ownTDTAndTCT** = false

flag to own TDT and TCT

4.8.2 Calibrating Objects

e

class **ElectronCalibrator** : public xAH::*Algorithm*

This is the algorithm class used to calibrate electrons.

In a nutshell, this algorithm performs the following actions:

- retrieves an `xAOD::ElectronContainer` from either `TEvent` or `TStore`
- makes a shallow copy container and fills it with energy-and-direction calibrated electrons using the `EgammaCalibrationAndSmearingTool` in Tools Used
- saves the shallow copy container to `TStore` from where it can be retrieved by algorithms downstream via name lookup

Public Functions

ElectronCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

The name of the input container for this algorithm to read from `TEvent` or `TStore`

std::string **m_outContainerName** = ""

The name of the nominal output container written by the algorithm to `TStore`

If the algorithm applies systematic variations, for each shallow copy saved to `TStore`, the systematic name will be appended to this.

bool **m_sort** = true

Sort the processed container elements by transverse momentum.

std::string **m_inputAlgoSystNames** = ""

The name of the vector containing the names of the systematically-varied containers from the upstream algorithm, which will be processed by this algorithm.

This vector is retrieved from the `TStore`. If left blank, it means there is no upstream algorithm which applies systematics. This is the case when processing straight from the original `xAOD` or `DxAOD`.

std::string **m_outputAlgoSystNames** = "ElectronCalibrator_Syst"

The name of the vector containing the names of the systematically-varied containers created by by this algorithm.

If *m_systName* is empty, the vector will contain only an empty string. When running on systematics, this is the string a downstream algorithm needs to process electrons.

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

std::string **m_esModel** = ""

std::string **m_decorrelationModel** = ""

bool **m_applyIsolationCorrection** = false

Apply isolation correction, not needed by default.

Private Members

int **m_numEvent**

int **m_numObject**

std::string **m_outAuxContainerName**

std::string **m_outSCContainerName**

std::string **m_outSCAuxContainerName**

std::vector<CP::SystematicSet> **m_systList**

CP::EgammaCalibrationAndSmearingTool ***m_EgammaCalibrationAndSmearingTool** = nullptr

CP::IsolationCorrectionTool ***m_IsolationCorrectionTool** = nullptr

apply leakage correction to calo based isolation variables for electrons

j

class **JetCalibrator** : public xAH::*Algorithm*

A wrapper to a few JetETMiss packages. By setting the configuration parameters detailed in the header documentation, one can:

- calibrate a given jet collection
- apply systematic variations for JES
- apply systematic variations for JER
- decorate the jet with the decision of the Jet Cleaning tool

When considering systematics, a new `xAOD::JetCollection` is created for each systematic variation. The names are then saved in a vector for downstream algorithms to use.

Public Functions

JetCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

The name of the input container for this algorithm to read from `TEvent` or `TStore`

std::string **m_outContainerName** = ""

The name of the nominal output container written by the algorithm to `TStore`

If the algorithm applies systematic variations, for each shallow copy saved to `TStore`, the systematic name will be appended to this.

std::string **m_jetAlgo** = ""

set to `AntiKt4EMTopo` for `AntiKt4EMTopoJets`

std::string **m_outputAlgo** = ""

name of vector holding names of jet systematics given by the JetEtmiss Tools

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

bool **m_recalibrateHLTJets** = false

whether to run HLT jet re-calibration

std::string **m_HLTVertexContainerName** = "HLT_IDVertex_FS"

vertex container name to use for HLT jet re-calibration

std::string **m_HLTAvgMuDecor** = "EventInfo.AvgMu"

HLT average mu decoration on EventInfo after formatting.

std::string **m_EvtInfoHLTNPVDecor** = ""

location of the HLT NPV on EventInfo object (e.g. EventInfo.NPV) this defaults to an empty string and is only configured in JetCalibrationTool when a non-empty string is provided

std::string **m_calibGSCDepth** = ""

GSCDepth property to override GSCDepth in config file when set to a non-empty string and GSC is in the calibration sequence.

std::string **m_calibConfigDir** = ""

config for JetCalibrationTool ConfigDir, set it to override tool defaults std::string **`m_calibConfigData`** = "JES_data2017_2016_2015_Recommendation_Aug2018_rel21.config"

config for JetCalibrationTool for Data

std::string **`m_calibConfigFullSim`** = "JES_data2017_2016_2015_Recommendation_Aug2018_rel21.config"

config for JetCalibrationTool for Full Sim MC

std::string **`m_calibConfigAFII`** = "JES_MC16Recommendation_AFII_EMTopo_April2018_rel21.config"

config for JetCalibrationTool for AFII MC

std::string **`m_calibSequence`** = ""

List of calibration steps. Auto-configured to the Jet/Etmiss recommendation if left blank.

std::string **`m_uncertConfig`** = ""

config for Jet Uncertainty Tool

std::string **`m_uncertMCType`** = ""

MC type for Jet Uncertainty Tool (need to be set for FullSim)

std::string **`m_overrideCalibArea`** = ""

Override CalibArea tag (default recommended)

std::string **`m_overrideUncertCalibArea`** = ""

Override uncertainties CalibArea tag (default recommended)

std::string **`m_overrideAnalysisFile`** = ""

Set analysis-specific jet flavour composition file for JetUncertainties (default: unknown comp.)

std::string **`m_overrideUncertPath`** = ""

Override uncertainties path (not recommended)

bool **`m_forceInsitu`** = false

when running data "_Insitu" is appended to calibration sequence

bool **`m_forceSmear`** = false

when running FullSim "_Smear" is appended to calibration sequence

bool **`m_jetCalibToolsDEV`** = false

when using DEV mode of JetCalibTools

bool **`m_addGhostMuonsToJets`** = false

Run muon-to-jet ghost association (recommended for MET)

bool **`m_doCleaning`** = true

enable to apply jet cleaning decoration

std::string **m_jetCleanCutLevel** = "LooseBad"
Cut Level.

bool **m_saveAllCleanDecisions** = false
Save all cleaning decisions as decorators.

bool **m_jetCleanUgly** = false
Do Ugly cleaning (i.e. TileGap 3)

bool **m_sort** = true
Sort the processed container elements by transverse momentum.

bool **m_cleanParent** = false
Apply jet cleaning to parent jet.

bool **m_applyFatJetPreSel** = false

bool **m_useLargeRTruthLabelingTool** = true
Use large-R jet truth labeling tool (needed for systematics)

std::string **m_truthLabelName** = "R10TruthLabel_R21Consolidated"
Name of the large-R jet truth labeling definition.

bool **m_isTruthJetCol** = false
Flag to indicate if using a truth jet collection.

bool **m_useTRUTH3** = true
Flag to indicate if input xAOD uses TRUTH3 style containers.

std::string **m_truthParticleContainerName** = "TruthParticles"
Name of the truth particle container if not using TRUTH3 containers.

std::string **m_truthBosonContainerName** = "TruthBosonsWithDecayParticles"
Name of the truth boson container if using TRUTH3 containers.

std::string **m_truthTopQuarkContainerName** = "TruthTopQuarkWithDecayParticles"
Name of the truth top quark container if using TRUTH3 containers.

bool **m_doJetTileCorr** = false
jet tile correction

bool **m_pseudoData** = false
needed in case want to treat MC as pseudoData for JER uncertainty propagation

bool **m_mcAndPseudoData** = false

Treat MC as usual, then run the JER uncertainties on it a second time treating it as pseudodata. Overrides m_pseudodata if true.

Private Functions

EL::StatusCode **executeSystematic**(const CP::SystematicSet &thisSyst, const xAOD::JetContainer *inJets, std::pair<xAOD::JetContainer*, xAOD::ShallowAuxContainer*> &calibJetsSC, std::vector<std::string> &vecOutContainerNames, bool isPDCopy)

EL::StatusCode **initializeUncertaintiesTool**(asg::AnaToolHandle<ICPJetUncertaintiesTool> &uncToolHandle, bool isData)

Private Members

bool **m_runSysts** = false

set to true if systematics asked for and exist

int **m_numEvent**

int **m_numObject**

std::string **m_calibConfig**

std::vector<CP::SystematicSet> **m_systList**

asg::AnaToolHandle<IJetCalibrationTool> **m_JetCalibrationTool_handle** = {"JetCalibrationTool", this}

asg::AnaToolHandle<ICPJetUncertaintiesTool> **m_JetUncertaintiesTool_handle** = {"JetUncertaintiesTool", this}

asg::AnaToolHandle<ICPJetUncertaintiesTool> **m_pseudodataJERTool_handle** = {"PseudodataJERTool", this}

asg::AnaToolHandle<IJetSelector> **m_JetCleaningTool_handle** = {"JetCleaningTool", this}

std::vector<asg::AnaToolHandle<IJetSelector>> **m_AllJetCleaningTool_handles**

std::vector<std::string> **m_decisionNames**

μ

class **MuonCalibrator** : public xAH::*Algorithm*

Public Functions

MuonCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName** = ""

std::string **m_calibrationMode** = "noOption"

Set calibrationMode property if different than noOption.

bool **m_isRun3Geo** = false

Switch on Run3 geometry for muon selector tool.

bool **m_do2StationsHighPt** = false

bool **m_sort** = true

std::string **m_inputAlgoSystNames** = ""

this is the name of the vector of names of the systematically varied containers produced by the upstream algo (e.g., the SC containers with calibration systematics)

std::string **m_outputAlgoSystNames** = "MuonCalibrator_Syst"

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systVal** = 0.0

std::string **m_systName** = ""

bool **m_forceDataCalib** = false

Force `MuonCalibrationPeriodTool.h` to calibrate data.

`MuonSelectorTool` depends on a specific decoration existing on Muons, namely `MuonSpectrometerPt`. This is decorated by the `MuonCalibrationAndSmearingTool`. However, you do not calibrate data by default so this tool would not be run on data.

In the case where you need the tool to be forced to run on data in order to have this decoration on your muons, you need to flip this boolean. See the Muon Combined Performance Working Group twiki for more information.

Note: This should not* modify the momentum of muons in data (according to the tool as of `MuonMomentumCorrections-01-00-37`).

Private Members

int **m_numEvent**

int **m_numObject**

std::string **m_outAuxContainerName**

std::string **m_outSCContainerName**

std::string **m_outSCAuxContainerName**

std::vector<CP::SystematicSet> **m_systList**

asg::AnaToolHandle<CP::MuonCalibTool> **m_muonCalibrationTool_handle** = {"CP::MuonCalibTool/MuonCalibrationTool", this} *τ*

class **TauCalibrator** : public xAH::*Algorithm*

Public Functions

TauCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName** = ""

std::string **m_RecommendationTag** = ""

bool **m_applyMVATESQualityCheck** = false

std::string **m_generator** = ""

std::string **m_campaign** = ""

bool **m_setAFII** = false

bool **m_setAF3** = false

bool **m_skipTruthMatchCheck** = false

bool **m_sort** = true

std::string **m_inputAlgoSystNames** = ""

this is the name of the vector of names of the systematically varied containers produced by the upstream algo (e.g., the SC containers with calibration systematics)

std::string **m_outputAlgoSystNames** = "TauCalibrator_Syst"

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

Private Members

int **m_numEvent**

int **m_numObject**

std::string **m_outAuxContainerName**

std::string **m_outSCContainerName**

std::string **m_outSCAuxContainerName**

std::vector<CP::SystematicSet> **m_systList**

asg::AnaToolHandle<TauAnalysisTools::ITauSmearingTool> **m_tauSmearingTool_handle** = {"TauAnalysisTools::TauSmearingTool/TauSmearingTool", this}

γ

class **PhotonCalibrator** : public xAH::*Algorithm*

Public Functions

PhotonCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName** = ""

std::string **m_overridePhotonCalibMap** = ""

std::string **m_tightIDConfigPath** = "ElectronPhotonSelectorTools/offline/20180825/PhotonIsEMTightSelectorCutDefs.conf"

std::string **m_mediumIDConfigPath** = "ElectronPhotonSelectorTools/offline/mc15_20150712/PhotonIsEMMediumSelectorCutDefs.conf"

std::string **m_looseIDConfigPath** = "ElectronPhotonSelectorTools/offline/mc15_20150712/PhotonIsEMLooseSelectorCutDefs.conf"

bool **m_sort** = true

std::string **m_inputAlgoSystNames** = ""

this is the name of the vector of names of the systematically varied containers produced by the upstream algo (e.g., the SC containers with calibration systematics)

std::string **m_outputAlgoSystNames** = "PhotonCalibrator_Syst"

this is the name of the vector of names of the systematically varied containers produced by THIS algo (these will be the m_inputAlgoSystNames of the algo downstream

bool **m_useAFII** = false

bool **m_useAF3** = false

float **m_systVal** = 0.0

std::string **m_systName** = ""

std::string **m_esModel** = "es2017_R21_v1"

std::string **m_decorrelationModel** = ""

int **m_randomRunNumber** = -1

bool **m_readIDFlagsFromDerivation** = false

To read PID decision from DAOD, rather than recalculate with tool.

Private Functions

EL::StatusCode **decorate**(xAOD::Photon *photon)

Private Members

std::string **m_outAuxContainerName**

std::string **m_outSCContainerName**

std::string **m_outSCAuxContainerName**

std::vector<CP::SystematicSet> **m_systList**

CP::EgammaCalibrationAndSmearingTool ***m_EgammaCalibrationAndSmearingTool** = nullptr

asg::AnaToolHandle<CP::IIsolationCorrectionTool> **m_isolationCorrectionTool_handle** = {"CP::IsolationCorrectionTool/IsolationCorrectionTool", this}

ElectronPhotonVariableCorrectionTool ***m_photonVarCorrectionTool** = nullptr

AsgPhotonIsEMSelector ***m_photonTightIsEMSelector** = nullptr

AsgPhotonIsEMSelector ***m_photonMediumIsEMSelector** = nullptr

AsgPhotonIsEMSelector ***m_photonLooseIsEMSelector** = nullptr

asg::AnaToolHandle<IAsgPhotonEfficiencyCorrectionTool> **m_photonTightEffTool_handle** = {"AsgPhotonEfficiencyCorrectionTool/tight", this}

asg::AnaToolHandle<IAsgPhotonEfficiencyCorrectionTool> **m_photonMediumEffTool_handle** = {"AsgPhotonEfficiencyCorrectionTool/medium", this}

asg::AnaToolHandle<IAsgPhotonEfficiencyCorrectionTool> **m_photonLooseEffTool_handle** = {"AsgPhotonEfficiencyCorrectionTool/loose", this}

4.8.3 Efficiency Correcting

b-jet

class **BJetEfficiencyCorrector** : public xAH::*Algorithm*

Public Functions

BJetEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

EL::StatusCode **executeEfficiencyCorrection**(const xAOD::JetContainer *inJets, const xAOD::EventInfo *eventInfo, bool doNominal)

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

unsigned int **getMCIndex**(int dsid)

void **makeMCIndexMap**(std::string effCalib)

std::string **getFlavorLabel**(const xAOD::Jet &jet) const

Public Members

std::string **m_inContainerName** = ""

std::string **m_inputAlgo** = ""

The name of the vector containing the names of the systematically-varied jet-related containers from the upstream algorithm, which will be processed by this algorithm.

Only jet calibration systematics or any other that create shallow copies of jet containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

std::string **m_systName** = ""

std::string **m_outputSystName** = "BJetEfficiency_Algo"

bool **m_writeSystToMetadata** = false

std::string **m_corrFileName** =
"xAODBTaggingEfficiency/13p6TeV/2023-22-13p6TeV-MC21-CDI_Test_2023-08-1_v1.root"

std::string **m_jetAuthor** = "AntiKt4EMPFlowJets"

float **m_minPt** = 20e3

Minimum pT in MeV for taggable jets.

std::string **m_taggerName** = "DL1r"

bool **m_useDevelopmentFile** = true

bool **m_coneFlavourLabel** = true

std::string **m_systematicsStrategy** = "SFEigen"

bool **m_errorOnTagWeightFailure** = true

BTaggingSelectionTool throws an error on missing tagging weights. If false, a warning is given instead.

bool **m_alwaysGetTagWeight** = false

Decorate tag weights even if we're not doing pseudocontinuous b-tagging.

std::string **m_operatingPt** = "FixedCutBEff_70"

Operating point.

std::string **m_operatingPtCDI** = ""

Operating point that CDI will understand.

bool **m_getScaleFactors** = false

will only get scale factors for calibrated working points

bool **m_useContinuous** = false

will get tagWeight, quantile, SF and InefficiencySF

std::string **m_decor** = "BTag"

The decoration key written to passing objects.

bool **m_tagDecisionOnly** = false

Only apply b-tag decision decoration; don't retrieve scale factors (Not recommended. For expert use.)

bool **m_setMapIndex** = false

Select an efficiency map for use in MC/MC and inefficiency scale factors, based on user specified selection of efficiency maps.

std::string **m_DSIDtoGenerator_filename** = "xAODAnaHelpers/DSIDtoGenerator.txt"

float **m_orBJetPtUpperThres** = -1

upper pt threshold of b-jet in OR in unit of GeV, negative value means no pt threshold

std::string **m_EfficiencyCalibration** = ""

Calibration to use for MC (EfficiencyB/C/T/LightCalibrations), "auto" to determine from sample name (multiple samples can be provided as long as they are separated by ';')

Example: "410470;410250;410558;410464" (Pythia8,Sherpa22,Herwig7,MG)

std::string **m_EigenvectorReductionB** = "Loose"

To change NP scheme for b-tagging systematics - Loose is the default value in athena.

std::string **m_EigenvectorReductionC** = "Loose"

std::string **m_EigenvectorReductionLight** = "Loose"

Private Members

std::string **m_decorSF** = ""

The decoration key written to passing objects.

std::string **m_decorWeight** = ""

std::string **m_decorQuantile** = ""

std::string **m_decorInefficiencySF** = ""

std::map<int, std::string> **m_DSIDtoGenerator**

std::map<std::string, unsigned int> **m_MCIndexes**

std::vector<std::string> **m_inputAlgoList**

bool **m_runAllSyst** = false

asg::AnaToolHandle<IBTaggingSelectionTool> **m_BJetSelectTool_handle** = {"BTaggingSelectionTool", this}

asg::AnaToolHandle<IBTaggingEfficiencyTool> **m_BJetEffSFTool_handle** = {"BTaggingEfficiencyTool", this}

std::vector<CP::SystematicSet> **m_systList** *e*

class **ElectronEfficiencyCorrector** : public xAH::*Algorithm*

This is the algorithm class that applies generic corrections to electrons. At the moment, only data/MC efficiency correction is included (electron trigger SF and others will follow…).

In a nutshell, this algorithm performs the following actions:

- retrieves an `xAOD::ElectronContainer` from either `TEvent` or `TStore`
- adds a scale factor (SF) decoration for each electron in the input container calculated via the `AsgElectronEfficiencyCorrectionTool` in Tools Used
- the nominal SF and all the systematically-varied ones are saved as a `vector<double>` decoration for each electron

Note: Bear in mind that this algorithm must be called after *`ElectronSelector`*. In fact, the configuration file(s) being used must have the same working point as the one chosen in the selector.

Public Functions

`ElectronEfficiencyCorrector`()

virtual EL::StatusCode **`setupJob`**(EL::Job &job)

virtual EL::StatusCode **`fileExecute`**()

virtual EL::StatusCode **`histInitialize`**()

virtual EL::StatusCode **`changeInput`**(bool firstFile)

virtual EL::StatusCode **`initialize`**()

virtual EL::StatusCode **`execute`**()

virtual EL::StatusCode **`postExecute`**()

virtual EL::StatusCode **`finalize`**()

virtual EL::StatusCode **`histFinalize`**()

virtual EL::StatusCode **`executeSF`**(const xAOD::ElectronContainer *inputElectrons, bool nominal, bool writeSystNames)

Public Members

std::string **`m_inContainerName`** = ""

The name of the input container for this algorithm to read from `TEvent` or `TStore` std::string **m_inputSystNamesElectrons**

The name of the vector containing the names of the systematically-varied electrons-related containers from the upstream algorithm, which will be processed by this algorithm.

Only electron calibration systematics or any other that create shallow copies of electron containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systValPID** = 0.0

float **m_systValIso** = 0.0

float **m_systValReco** = 0.0

float **m_systValTrig** = 0.0

std::string **m_systNamePID** = ""

std::string **m_systNameIso** = ""

std::string **m_systNameReco** = ""

std::string **m_systNameTrig** = ""

std::string **m_outputSystNamesPID** = "EleEffCorr_PIDSyst"

std::string **m_outputSystNamesIso** = "EleEffCorr_IsoSyst"

std::string **m_outputSystNamesReco** = "EleEffCorr_RecoSyst"

std::string **m_outputSystNamesTrig** = "EleEffCorr_TrigSyst"

std::string **m_correlationModel** = "FULL"

Systematic correlation model.

std::string **m_WorkingPointPID** = ""

PID working point (LooseBLayer, Medium, Tight)

std::string **m_WorkingPointIso** = ""

Isolation working point.

std::string **m_WorkingPointReco** = ""
Reconstruction working point (Reconstruction only)

std::string **m_WorkingPointTrig** = ""
Trigger working point.

bool **m_usePerElectronTriggerSFs** = true

std::string **m_overrideMapFilePath** = ""
Override corrections map file (not recommended)

Private Members

int **m_numEvent**

int **m_numObject**

std::vector<CP::SystematicSet> **m_systListPID**

std::vector<CP::SystematicSet> **m_systListIso**

std::vector<CP::SystematicSet> **m_systListReco**

std::vector<CP::SystematicSet> **m_systListTrig**

AsgElectronEfficiencyCorrectionTool ***m_asgElEffCorrTool_elSF_PID** = nullptr

std::string **m_pidEffSF_tool_name**

AsgElectronEfficiencyCorrectionTool ***m_asgElEffCorrTool_elSF_Iso** = nullptr

std::string **m_IsoEffSF_tool_name**

AsgElectronEfficiencyCorrectionTool ***m_asgElEffCorrTool_elSF_Reco** = nullptr

std::string **m_RecoEffSF_tool_name**

AsgElectronEfficiencyCorrectionTool ***m_asgElEffCorrTool_elSF_Trig** = nullptr

std::string **m_TrigEffSF_tool_name**

AsgElectronEfficiencyCorrectionTool ***m_asgElEffCorrTool_elSF_TrigMCEff** = nullptr std::string **m_TrigMCEff_tool_name**

μ

class **MuonEfficiencyCorrector** : public xAH::*Algorithm*

Public Functions

MuonEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **executeSF**(const xAOD::EventInfo *eventInfo, const xAOD::MuonContainer *inputMuons, bool nominal, bool writeSystNames)

Public Members

std::string **m_inContainerName** = ""

std::string **m_overrideCalibRelease** = ""

Recommendations release (not recommended to change)

std::string **m_WorkingPointReco** = "Loose"

std::string **m_WorkingPointIso** = "LooseTrackOnly"

bool **m_AllowZeroSF** = false

Use with caution!!!

std::string **m_MuTrigLegs** = "HLT_mu26_imedium"

list of comma-separated single-mu trigger corrections. Individual legs of di-mu menus can be parsed bool **m_usePerMuonTriggerSFs** = true

Get per-muon trigger SF (default: true) [if false it will take into account combinatorics using all muons from the input muon container].

std::string **m_WorkingPointTTVA** = "TTVA"

std::string **m_inputSystNamesMuons** = ""

The name of the vector containing the names of the systematically-varied muons-related containers from the upstream algorithm, which will be processed by this algorithm.

Only muon calibration systematics or any other that create shallow copies of electron containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systValReco** = 0.0

float **m_systValIso** = 0.0

float **m_systValTrig** = 0.0

float **m_systValTTVA** = 0.0

std::string **m_systNameReco** = ""

std::string **m_systNameIso** = ""

std::string **m_systNameTrig** = ""

std::string **m_systNameTTVA** = ""

std::string **m_outputSystNamesReco** = "MuonEfficiencyCorrector_RecoSyst"

std::string **m_outputSystNamesIso** = "MuonEfficiencyCorrector_IsoSyst"

std::string **m_outputSystNamesTrig** = "MuonEfficiencyCorrector_TrigSyst"

std::string **m_outputSystNamesTTVA** = "MuonEfficiencyCorrector_TTVASyst"

Private Members

int **m_numEvent**

int **m_numObject**

std::vector<CP::SystematicSet> **m_systListReco**

std::vector<CP::SystematicSet> **m_systListIso**

std::vector<CP::SystematicSet> **m_systListTrig**

std::vector<CP::SystematicSet> **m_systListTTVA**

std::string **m_outputSystNamesTrigBase**

asg::AnaToolHandle<CP::IPileupReweightingTool> **m_pileup_tool_handle** = {"CP::PileupReweightingTool/Pileup"}

asg::AnaToolHandle<CP::IMuonEfficiencyScaleFactors> **m_muRecoSF_tool**

std::string **m_recoEffSF_tool_name**

asg::AnaToolHandle<CP::IMuonEfficiencyScaleFactors> **m_muIsoSF_tool**

std::string **m_isoEffSF_tool_name**

asg::AnaToolHandle<CP::IMuonTriggerScaleFactors> **m_muTrigSF_tool**

std::string **m_trigEffSF_tool_name**

asg::AnaToolHandle<CP::IMuonEfficiencyScaleFactors> **m_muTTVASF_tool**

std::string **m_TTVAEffSF_tool_name**

std::map<std::string, std::string> **m_SingleMuTriggerMap** *τ*

class **TauEfficiencyCorrector** : public xAH::*Algorithm*

Public Functions

TauEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **executeSF**(const xAOD::EventInfo *eventInfo, const xAOD::TauJetContainer *inputTaus, bool nominal, bool writeSystNames)

Public Members

std::string **m_RecommendationTag** = ""

std::string **m_inContainerName** = ""

std::string **m_WorkingPointReco** = ""

std::string **m_WorkingPointEleOLRHadTau** = ""

std::string **m_WorkingPointTauEleID** = ""

std::string **m_WorkingPointTauJetID** = ""

std::string **m_TriggerName** = ""

std::string **m_inputSystNamesTaus** = ""

The name of the vector containing the names of the systematically-varied taus-related containers from the upstream algorithm, which will be processed by this algorithm.

Only tau systematics or any other that create shallow copies of tau containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systVal** = 0.0

std::string **m_systName** = ""

std::string **m_outputSystNames** = "TauEfficiencyCorrector_Syst"

Private Members

int **m_numEvent**

int **m_numObject**

std::vector<CP::SystematicSet> **m_systList**

asg::AnaToolHandle<CP::IPileupReweightingTool> **m_pileup_tool_handle** = {"CP::PileupReweightingTool/Pileup"}

asg::AnaToolHandle<TauAnalysisTools::ITauEfficiencyCorrectionsTool> **m_tauEffCorrTool_handle** = {"TauAnalysisTools::TauEfficiencyCorrectionsTool/TauEfficiencyCorrectionsTool", this}

asg::AnaToolHandle<TauAnalysisTools::ITauSelectionTool> **m_tauSelTool_handle** = {"TauAnalysisTools::TauSelectionTool/TauSelectionTool"}

4.8.4 Selecting Objects

Event

class **BasicEventSelection** : public xAH::*Algorithm*

This algorithm performs the very basic event selection. This should be the first algo in the algo chain. It can create weighted and unweighted cutflow objects to be picked up downstream by other xAH algos, and your own. The selection applied in data only is:

- GRL (can be turned off)
- LAr Error

- Tile Error
- Core Flag

In both data and simulation (MC), the following cuts are applied

- the highest sum p_T^2 primary vertex has 2 or more tracks (see *m_applyPrimaryVertexCut*)
- trigger requirements (see *m_applyTriggerCut*)

For derivations, the metadata can be accessed and added to the cutflow for normalization. The parameters to control the trigger are described in this header file. If one wants to write out some of the trigger information into a tree using *HelpTreeBase*, flags must be set here.

Note: For MC only, the pileup reweight can also be applied.

Public Functions

BasicEventSelection()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

bool **m_isTLAData** = false

Flag to determine when running on TLA data for different handling of TDT.

bool **m_truthLevelOnly** = false

Protection when running on truth xAOD.

bool **m_setAFII** = false

SimulationFlavour will be determined from the sample MetaData, unless AFII or FS is explicitely requested with the following flags.

bool **m_setAF3** = false

bool **m_setFS** = false

bool **m_applyGRLCut** = false

Apply GRL selection.

std::string **m_GRLxml** = ""

Path to GRL XML file.

std::string **m_GRLExcludeList** = ""

Run numbers to skip in GRL.

bool **m_cleanPowheg** = false

Clean Powheg huge weight.

bool **m_reweightSherpa22** = false

Reweight Sherpa 2.2 Samples.

bool **m_doPUreweighting** = false

Reweight pile-up profile μ

bool **m_doPUreweightingSys** = false

std::string **m_lumiCalcFileNames** = ""

Comma separated list of filenames.

std::string **m_PRWFileNames** = ""

Comma separated list of filenames.

bool **m_autoconfigPRW** = false

Automatically configure PRW using config files from SUSYTools instead of using m_PRWFileNames.

bool **m_useCommonPRWFiles** = false

Configure PRW using common files instead of DSID-specific files.

std::string **m_prwActualMu2016File** = ""

actualMu configuration file for the MC16a campaign (2015/2016). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2017File** = ""

actualMu configuration file for the MC16d campaign (2017). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2018File** = ""

actualMu configuration file for the MC16e campaign (2018). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2022File** = ""

actualMu configuration file for the MC23a campaign (2022). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2023File** = ""

actualMu configuration file for the MC23d campaign (2023). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_commonPRWFileMC20a** = "PileupReweighting/mc20_common/mc20a.284500.physlite.prw.v1.root"

Common PRW file for the MC20a campaign (2015/16). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC20d** = "PileupReweighting/mc20_common/mc20d.300000.physlite.prw.v1.root"

Common PRW file for the MC20d campaign (2017). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC20e** = "PileupReweighting/mc20_common/mc20e.310000.physlite.prw.v1.root"

Common PRW file for the MC20e campaign (2018). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC23a** = "PileupReweighting/mc23_common/mc23a.410000.physlite.prw.v2.root"

Common PRW file for the MC23a campaign (2022). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC23c** = "PileupReweighting/mc23_common/mc23c.450000.physlite.prw.v1.root"

Common PRW file for the MC23c campaign (2023). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC23d** = "PileupReweighting/mc23_common/mc23d.450000.physlite.prw.v1.root"

Common PRW file for the MC23d campaign (2023). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_mcCampaign**

mc16(acd) to bypass the automatic campaign determination from AMI, several campaigns can be separated by a comma. Only used when m_autoconfigPRW is true

std::string **m_periodConfig** = "auto"

Use Period Configuration or auto.

bool **m_checkStreams** = false

Print streamTags (only in debug mode)

int **m_actualMuMin** = -1

The minimum threshold for `EventInfo::actualInteractionsPerCrossing()`

int **m_actualMuMax** = -1

The maximum threshold for `EventInfo::actualInteractionsPerCrossing()`

bool **m_calcBCIDInfo** = false

Calculate distance to nearest empty and unpaired BCIDs.

bool **m_applyPrimaryVertexCut** = false

Enable to apply a primary vertex cut.

int **m_PVNTrack** = 2

Minimum number of tracks from **the** primary vertex (Harmonized Cut)

bool **m_applyEventCleaningCut** = false

bool **m_applyCoreFlagsCut** = false

bool **m_applyJetCleaningEventFlag** = false

recommended way to clean all jets, but especially collections other than EMTopo … equivalent to "loose" jet-by-jet cleaning!

bool **m_applyIsBadBatmanFlag** = false

should only ever be used in 2015 and 2016 data, for analyses which may be of interest for analyses where fake MET can be an issue

bool **m_printBranchList** = false

std::string **m_triggerSelection** = ""

RegEx expression to choose triggers to consider to be cut on with *m_applyTriggerCut*

std::string **m_extraTriggerSelection** = ""

Decisions of triggers which are saved but not cut on.

bool **m_applyTriggerCut** = false

Skip events in which the trigger string *m_triggerSelection* does not fire

bool **m_storeTrigDecisions** = false

Save string of fired triggers matching *m_triggerSelection*

bool **m_storePassL1** = false

Save if any L1 trigger fired, e.g. `"L1_.*"`

bool **m_storePassHLT** = false

Save if any HLT trigger fired, e.g. "HLT_.*"

bool **m_storeTrigKeys** = false

Save master, L1, and HLT key.

bool **m_storePrescaleWeight** = true

Save the trigger prescale weight.

std::string **m_derivationName** = ""

The name of the derivation (use this as an override)

bool **m_useMetaData** = true

Retrieve and save information on DAOD selection.

std::string **m_metaDataStreamName** = "metadata"

std::string **m_duplicatesStreamName** = "duplicates_tree"

bool **m_checkDuplicatesData** = false

Check for duplicated events in data

bool **m_checkDuplicatesMC** = false

Check for duplicated events in MC

Private Functions

StatusCode **autoconfigurePileupRWTool**()

Automatically add the required PRW config file for the DSID being processed to the PRW tool.

helper functions The PRW config files stored by SUSYTools are added to the m_pileup_tool_handle. If the m_mcCampaign is not set, the campaign is determined automatically. If it is set, then all of the campaings listed in the setting are added.

Private Members

std::set<std::pair<uint32_t, uint32_t>> **m_RunNr_VS_EvtNr**

std::vector<std::string> **m_triggerUnprescaleList**

std::vector<std::string> **m_extraTriggerSelectionList**

asg::AnaToolHandle<IGoodRunsListSelectionTool> **m_grl_handle** = {"GoodRunsListSelectionTool", this} asg::AnaToolHandle<CP::IPileupReweightingTool> **m_pileup_tool_handle** = {"CP::PileupReweightingTool/Pileup"}

asg::AnaToolHandle<TrigConf::ITrigConfigTool> **m_trigConfTool_handle** = {"TrigConf::xAODConfigTool/xAODConfigTool", this}

asg::AnaToolHandle<Trig::TrigDecisionTool> **m_trigDecTool_handle** = {"Trig::TrigDecisionTool/TrigDecisionTool"}

int **m_eventCounter**

TH1D ***m_histSumW** = nullptr

TH1D ***m_histEventCount** = nullptr

uint64_t **m_MD_initialNevents**

uint64_t **m_MD_finalNevents**

double **m_MD_initialSumW**

double **m_MD_finalSumW**

double **m_MD_initialSumWSquared**

double **m_MD_finalSumWSquared**

std::string **m_mcCampaignMD**

TH1D ***m_cutflowHist** = nullptr

TH1D ***m_cutflowHistW** = nullptr

int **m_cutflow_all**

int **m_cutflow_init**

int **m_cutflow_duplicates**

int **m_cutflow_grl**

int **m_cutflow_lar**

int **m_cutflow_tile**

int **m_cutflow_SCT**

int **m_cutflow_core**

int **m_cutflow_jetcleaning**

int **m_cutflow_isbadbatman**

int **m_cutflow_npv**

int **m_cutflow_trigger**

TH1D ***m_el_cutflowHist_1** = nullptr

TH1D ***m_el_cutflowHist_2** = nullptr

TH1D ***m_mu_cutflowHist_1** = nullptr

TH1D ***m_mu_cutflowHist_2** = nullptr

TH1D ***m_ph_cutflowHist_1** = nullptr

TH1D ***m_tau_cutflowHist_1** = nullptr

TH1D ***m_tau_cutflowHist_2** = nullptr

TH1D ***m_jet_cutflowHist_1** = nullptr

TH1D ***m_trk_cutflowHist_1** = nullptr

TH1D ***m_truth_cutflowHist_1** = nullptr

TTree ***m_duplicatesTree** = nullptr

TTree for duplicates bookeeping

int **m_duplRunNumber**

long int **m_duplEventNumber**

Overlap Removal

class **OverlapRemover** : public xAH::*Algorithm*

A wrapper of the overlap removal tool in the ASG AssociationUtils package.

The logic of the OLR belongs to the ASG tool itself, and is described extensively in the Analysis Harmonisation Task Force note.

If you wish to apply a custom OLR scheme, please contact the author marco.milesi@cern.ch for detailed instructions.

The idea behind this algorithm is to consistently thread together the inputs from upstream xAODAnaHelpers algorithms based on user's configuration, handling also the case where systematics on the input physics objects are taken into account. Here follows a usage example.

Consider the simplified scenario where we care only about *jets** and **electrons**. Assuming the typical xAODAnaHelpers analysis configuration through `xAH_config`, the analysis workflow could look like the following:

```
c = xAH_config()
# ...
c.algorithm("JetSelector", JetSelectorDict)
c.algorithm("ElectronSelector", ElectronSelectorDict)
# ...
c.algorithm("OverlapRemover", OverlapRemoverDict)
# ...
```

where each algorithm has the following I/O systematics configuration (via python dictionaries):

```
 JetSelectorDict = {
     # ...
     "m_inputAlgo" : "JetCalibrator_Syst",
     "m_outputAlgo" : "JetSelector_Syst",
     # ...
 }

 ElectronSelectorDict = {
     # ...
     "m_inputAlgo" : "ElectronCalibrator_Syst",
     "m_outputAlgo" : "ElectronSelector_Syst",
     # ...
}

OverlapRemoverDict = {
     # ...
     "m_inputAlgoJets" : "JetSelector_Syst", # leave empty when not considering jet␣
↪systematics
     "m_inputAlgoElectrons" : "ElectronSelector_Syst", # leave empty when not␣
↪considering electron systematics
     # ...
}
```

In this way the overlap removal algorithm will be able to correctly work out all the combinatorics, generating output `xAOD` containers for jets and electrons for each input systematics combination to be subsequently used downstream according to the user's needs. The overlap removal algorithm creates an output systematic list that is a combination of systematics from all input containers.

Public Functions

OverlapRemover()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **fillObjectCutflow**(const xAOD::IParticleContainer *objCont, const std::string &overlapFlag = "passOR", const std::string &selectFlag = "passSel")

Fill the cutflow histograms.

Parameters

- **objCont** – The `xAOD` container to be considered
- **overlapFlag** – The string identifying objects not overlapping with another object, to be kept (default is `"passOR"`)
- **selectFlag** – The string identifying selected objects (default is `"passSel"`)

virtual EL::StatusCode **executeOR**(const xAOD::ElectronContainer *inElectrons, const xAOD::MuonContainer *inMuons, const xAOD::JetContainer *inJets, const xAOD::PhotonContainer *inPhotons, const xAOD::TauJetContainer *inTaus, *SystType* syst_type = NOMINAL, std::vector<std::string> *sysVec = nullptr, std::vector<std::string> *sysVecOut = nullptr)

Function that internally calls the OLR tool for the input containers (and systematics)

Parameters

- **inElectrons** – Input `xAOD` container for electrons
- **inMuons** – Input `xAOD` container for muons
- **inJets** – Input `xAOD` container for jets
- **inPhotons** – Input `xAOD` container for photons
- **inTaus** – Input `xAOD` container for taus
- **syst_type** – The type of object for which input systematics should be considered. Default is `NOMINAL`

- **sysVec** – The list of the input systematics for a given object. Must match with the choice of syst_type. Default is nullptr

EL::StatusCode **setCutFlowHist**()

Setup cutflow histograms.

EL::StatusCode **setCounters**()

Initialise counters for events/objects.

Public Members

bool **m_useCutFlow** = true

Fill the cutflow histogram(s) for object counting.

bool **m_decorateSelectedObjects**

Decorate selected objects (the default decoration string is passOR)

std::string **m_decor** = "passOR"

bool **m_createSelectedContainers**

Make a copy of input container(s) with selected objects (using SG::VIEW_ELEMENTS to be light weight)

bool **m_useSelected** = false

In the OLR, consider only objects passing a (pre)selection.

std::string **m_bTagWP** = ""

Use b-tagging decision, set previously with the given decoration name, to remove electrons and muons.

Note: This is automatically set by *BJetEfficiencyCorrector*

bool **m_linkOverlapObjects** = true

Create a link between overlapped objects.

bool **m_useBoostedLeptons** = false

Use boosted object working point.

bool **m_doEleEleOR** = false

Do overlap removal between electrons (HSG2 prescription)

bool **m_applyRelPt** = false

Turn ON ApplyRelPt in MuJetOverlapTool (default is false)

bool **m_lepFavWP** = false

Turn ON Lepton favored working point (HSG2 prescription) std::string **m_outputAlgoSystNames** = "ORAlgo_Syst"

Output systematics list container name.

std::string **m_inContainerName_Electrons** = ""

Input container name.

std::string **m_outContainerName_Electrons** = ""

Output container name.

std::string **m_inputAlgoElectrons** = ""

Name of the `std::vector` of systematics coming from the upstream algorithm

std::string **m_inContainerName_Muons** = ""

std::string **m_outContainerName_Muons** = ""

std::string **m_inputAlgoMuons** = ""

std::string **m_inContainerName_Jets** = ""

std::string **m_outContainerName_Jets** = ""

std::string **m_inputAlgoJets** = ""

std::string **m_inContainerName_Photons** = ""

std::string **m_outContainerName_Photons** = ""

std::string **m_inputAlgoPhotons** = ""

std::string **m_inContainerName_Taus** = ""

std::string **m_outContainerName_Taus** = ""

std::string **m_inputAlgoTaus** = ""

Protected Types

enum **SystType**

An enum encoding systematics according to the various objects.

Values:

enumerator **NOMINAL**

enumerator **ELSYST**

enumerator **MUSYST**

enumerator **JETSYST**

enumerator **PHSYST**

enumerator **TAUSYST**

Protected Attributes

int **m_numEvent**

A counter for the number of processed events.

int **m_numObject**

A counter for the number of processed objects.

int **m_numEventPass**

A counter for the number of passed events.

int **m_weightNumEventPass**

A counter for the number of passed *weighted* events.

int **m_numObjectPass**

A counter for the number of passed objects.

bool **m_useElectrons** = false

Consider electrons in the OLR.

This is set to `false` if *m_inContainerName_Electrons* is set as an empty string. Electrons (unlike jets) are considered "optional" objetcs in the OLR.

bool **m_useMuons** = false

Consider muons in the OLR.

This is set to `false` if *m_inContainerName_Muons* is set as an empty string. Muons (unlike jets) are considered "optional" objects in the OLR.

bool **m_usePhotons** = false

Consider photons in the OLR.

This is set to `false` if *m_inContainerName_Photons* is set as an empty string. Photons (unlike jets) are considered "optional" objects in the OLR.

bool **m_useTaus** = false

Consider taus in the OLR.

This is set to `false` if *m_inContainerName_Taus* is set as an empty string. Taus (unlike jets) are considered "optional" objects in the OLR.

std::string **m_outAuxContainerName_Electrons**

Output auxiliary container name.

std::string **m_outAuxContainerName_Muons**

Output auxiliary container name.

std::string **m_outAuxContainerName_Jets**

Output auxiliary container name.

std::string **m_outAuxContainerName_Photons**

Output auxiliary container name.

std::string **m_outAuxContainerName_Taus**

Output auxiliary container name.

ORUtils::ToolBox **m_ORToolbox**

Pointer to the CP Tool which performs the actual OLR.

TH1D ***m_el_cutflowHist_1** = nullptr

Pointer to the histogram for the electron cutflow.

TH1D ***m_mu_cutflowHist_1** = nullptr

Pointer to the histogram for the muon cutflow.

TH1D ***m_jet_cutflowHist_1** = nullptr

Pointer to the histogram for the jet cutflow.

TH1D ***m_ph_cutflowHist_1** = nullptr

Pointer to the histogram for the photon cutflow.

TH1D ***m_tau_cutflowHist_1** = nullptr

Pointer to the histogram for the tau cutflow.

int **m_el_cutflow_OR_cut**

int **m_mu_cutflow_OR_cut**

int **m_jet_cutflow_OR_cut**

int **m_ph_cutflow_OR_cut**

int **m_tau_cutflow_OR_cut**

e

class **ElectronSelector** : public xAH::*Algorithm*

This is the algorithm class that selects electrons according to user's choice.

In a nutshell, this algorithm performs the following actions:

- retrieves an `xAOD::ElectronContainer` from either `TEvent` or `TStore`
- iterates over the input container, and if electron passes selection, copies it in a `ConstDataVector(SG::VIEW_ELEMENTS)` container. Otherwise, the electron is skipped
- saves the view container to `TStore`, from where it can be retrieved by algorithms downstream via a name lookup

Public Functions

ElectronSelector()

~ElectronSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::ElectronContainer *inElectrons, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::ElectronContainer> *selectedElectrons)

virtual int **passCuts**(const xAOD::Electron *electron, const xAOD::Vertex *primaryVertex)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

The name of the input container for this algorithm read from `TEvent` or `TStore`

std::string **m_outContainerName** = ""

The name of the nominal output container written by the algorithm to `TStore`

std::string **m_inputAlgoSystNames** = ""

The name of the vector containing the names of the systematically-varied containers from the upstream algorithm, which will be processed by this algorithm.

This vector is retrieved from the `TStore`. If left blank, it means there is no upstream algorithm which applies systematics. This is the case when processing straight from the original `xAOD` or `DxAOD`.

std::string **m_outputAlgoSystNames** = "ElectronSelector_Syst"

The name of the vector containing the names of the systematically-varied containers created by by this algorithm.

If *m_systName* is empty, the vector will contain only an empty string. When running on systematics, this is the string a downstream algorithm needs to process electrons.

bool **m_decorateSelectedObjects** = true

Adds a `passSel` decoration for objects that pass selection.

bool **m_createSelectedContainer** = false

Fill using a read-only container (`SG::VIEW_ELEMENTS`) to `TStore`

int **m_nToProcess** = -1

Number of objects to process, set `n=-1` to look at all.

int **m_pass_min** = -1

Require event to have minimum number of objects passing selection.

int **m_pass_max** = -1

Require event to have maximum number of objects passing selection.

float **m_pT_max** = 1e8

[MeV] Require objects to have maximum transverse momentum threshold

float **m_pT_min** = 1e8

[MeV] Require objects to have minimum transverse momentum threshold

float **m_eta_max** = 1e8

Require objects to have maximum $|\eta|$ value bool **m_vetoCrack** = true

Require objects to have $|\eta|$ outside the crack region using `caloCluster->eta()`

float **m_d0_max** = 1e8

Require objects to have a maximum d_0 [mm] (transverse impact parameter)

float **m_d0sig_max** = 1e8

Require objects to have a maximum d_0 significance at BL

float **m_z0sintheta_max** = 1e8

Require objects to have maximum $z_0 \sin(\theta)$ [mm] (longitudinal impact paramter) at BL - corrected with vertex info

bool **m_doAuthorCut** = true

Perform author kinematic cut.

bool **m_doOQCut** = true

Perform object quality cut.

bool **m_readIDFlagsFromDerivation** = false

To read electron PID decision from DAOD, rather than recalculate with tool.

bool **m_doModifiedEleId** = false

To correct egamma bug, see ATLSUSYSW-445.

bool **m_doLHPID** = true

Instantiate and perform the electron Likelihood PID.

bool **m_doLHPIDcut** = false

Cut on electron Likelihood PID (recommended)

std::string **m_LHOperatingPoint** = "Loose"

Loosest Likelihood PID operating point to save.

bool **m_doCutBasedPID** = false

Instantiate and perform the electron cut-based PID.

bool **m_doCutBasedPIDcut** = false

Cut on electron cut-based PID.

std::string **m_CutBasedOperatingPoint** = "Loose"

Loosest cut-based PID operating point to save.

std::string **m_MinIsoWPCut** = ""

reject objects which do not pass this isolation cut - default = “” (no cut) std::string **m_IsoWPList** = "FCLoose,FCTight,Gradient,FCHighPtCaloOnly"

decorate objects with `isIsolated_*` flag for each WP in this input list - default = all current ASG WPs

std::string **m_CaloIsoEff** = "0.1*x+90"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_TrackIsoEff** = "98"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_CaloBasedIsoType** = "topoetcone20"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_TrackBasedIsoType** = "ptvarcone20"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_singleElTrigChains** = ""

A comma-separated string w/ alll the HLT single electron trigger chains for which you want to perform the matching. This is passed by the user as input in configuration If left empty (as it is by default), no trigger matching will be attempted at all.

std::string **m_diElTrigChains** = ""

A comma-separated string w/ alll the HLT di-electron trigger chains for which you want to perform the matching. This is passed by the user as input in configuration If left empty (as it is by default), no trigger matching will be attempted at all.

double **m_minDeltaR** = 0.07

Recommended threshold for egamma triggers: see https://svnweb.cern.ch/trac/atlasoff/browser/Trigger/TrigAnalysis/TriggerMatchingTool/trunk/src/TestMatchingToolAlg.cxx.

bool **m_applyCrackVetoCleaning** = false

Apply fix to EGamma Crack-Electron topocluster association bug for MET (PFlow) / false by default.

bool **m_merged_electrons** = false

Element links need to be updated if merged electrons are used (LRT + std) / false by default.

std::string **m_trigInputPrefix** = ""

Input prefix of trigger decision tool.

std::string **m_isoDecSuffix** = ""

Private Members

bool **m_doBLTrackQualityCut**

Performs the Likelihood PID B-Layer cut locally.

> **Note:** Occurs automatically only if *m_LHOperatingPoint* is LooseBL and *m_readIDFlagsFromDerivation* is true

std::string **m_outAuxContainerName**

the name of the auxiliary store for the output container

int **m_numEvent**

keep track of the total number of events processed

int **m_numObject**

keep track of the total number of objects processed

int **m_numEventPass**

keep track of the number of passed events, and fill the cutflow (relevant only if using the algo to skim events: see *m_pass_max* and *m_pass_min* above)

int **m_weightNumEventPass**

keep track of the number of weighted passed events, and fill the cutflow (relevant only if using the algo to skim events: see *m_pass_max* and *m_pass_min* above)

int **m_numObjectPass**

keep track of the number of selected objects

TH1D ***m_cutflowHist** = nullptr

histogram for event cutflow

TH1D ***m_cutflowHistW** = nullptr

histgram for weighted event cutflow

int **m_cutflow_bin**

index of bin corresponding to this step of the full cutflow

bool **m_isUsedBefore** = false

checks if the algorithm has been used already

TH1D ***m_el_cutflowHist_1** = nullptr

TH1D ***m_el_cutflowHist_2** = nullptr

int **m_el_cutflow_all**

int **m_el_cutflow_author_cut**

int **m_el_cutflow_OQ_cut**

int **m_el_cutflow_ptmax_cut**

int **m_el_cutflow_ptmin_cut**

int **m_el_cutflow_eta_cut**

int **m_el_cutflow_z0sintheta_cut**

int **m_el_cutflow_d0_cut**

int **m_el_cutflow_d0sig_cut**

int **m_el_cutflow_BL_cut**

int **m_el_cutflow_PID_cut**

int **m_el_cutflow_iso_cut**

std::vector<std::string> **m_IsoKeys**

asg::AnaToolHandle<CP::IIsolationSelectionTool> **m_isolationSelectionTool_handle** = {"CP::IsolationSelectionTool/IsolationSelectionTool", this}

MC15 ASG tool for isolation.

CP::IsolationSelectionTool ***m_isolationSelectionTool** = {nullptr}

asg::AnaToolHandle<Trig::TrigDecisionTool> **m_trigDecTool_handle** = {"Trig::TrigDecisionTool/TrigDecisionTool"}

asg::AnaToolHandle<Trig::IMatchingTool> **m_trigElectronMatchTool_handle**

asg::AnaToolHandle<Trig::IMatchScoringTool> **m_scoreTool** = {"Trig::DRScoringTool/DRScoringTool"}

bool **m_doTrigMatch** = true

This internal variable gets set to false if no triggers are defined or if TrigDecisionTool is missing.

ElectronLHPIDManager ***m_el_LH_PIDManager** = nullptr

class to manage LH PID selection/decorations - see ISSUE for explaination

ElectronCutBasedPIDManager ***m_el_CutBased_PIDManager** = nullptr

class to manage cut-based PID selection/decorations - see ISSUE for explaination

std::vector<std::string> **m_singleElTrigChainsList**

contains all the HLT trigger chains tokens extracted from *m_singleElTrigChains*

std::vector<std::string> **m_diElTrigChainsList**

contains all the HLT trigger chains tokens extracted from *m_diElTrigChains*

j

class **JetSelector** : public xAH::*Algorithm*

Public Functions

JetSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual bool **executeSelection**(const xAOD::JetContainer *inJets, float mcEvtWeight, bool count, std::string outContainerName, bool isNominal)

virtual int **PassCuts**(const xAOD::Jet *jet)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_truthJetContainer** = "AntiKt4TruthJets"

truth jet container name (used for JVT SF)

std::string **m_inputAlgo** = ""

input type - from xAOD or from xAODAnaHelper Algo output

std::string **m_outputAlgo** = ""

output type - this is how the vector<string> w/ syst names will be saved in TStore

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

std::string **m_jetScaleType** = ""

Type of Scale Momementum.

std::string **m_decor** = "passSel"

The decoration key written to passing objects.

bool **m_decorateSelectedObjects** = true

decorate selected objects? defaul passSel

bool **m_createSelectedContainer** = false

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1

look at n objects

bool **m_cleanJets** = true

require cleanJet decoration to not be set and false

int **m_cleanEvtLeadJets** = -1

kill event if any of the N leading jets are not clean

bool **m_cleanEvent** = false

Kill event if any passing jets are not clean.

Note: The jets need the *cleanJet* decoration which is set when you enable `JetCalibrator::m_doCleaning`

bool **m_markCleanEvent** = false

Mark event with decorator if any passing jets are not clean.

std::string **m_jetScale4Selection** = "Final"

Choose the scale at which the selection is performed (default "Final", i.e. default 4vector)

bool **m_doMCCleaning** = false

(MC-only) Kill pileup overlay event if reconstructed jets avg(pT1,pT2) > 1.4*(truth jet pT1)

float **m_mcCleaningCut** = 1.4

Change the default 1.4 cut to x > 1.0.

int **m_pass_min** = -1

minimum number of objects passing cuts

int **m_pass_max** = -1

maximum number of objects passing cuts

float **m_pT_max** = 1e8

require pT < pt_max

float **m_pT_min** = 1e8

require pT > pt_min

float **m_ET_max** = 1e8

require ET < ET_max

float **m_ET_min** = 1e8

require ET > ET_min

float **m_eta_max** = 1e8

require eta < eta_max

float **m_eta_min** = 1e8

require eta > eta_min

float **m_detEta_max** = 1e8

require detEta < detEta_max

float **m_detEta_min** = 1e8
    require detEta > detEta_min

float **m_mass_max** = 1e8
    require mass < mass_max

float **m_mass_min** = 1e8
    require mass > mass_min

float **m_rapidity_max** = 1e8
    require rapidity < rapidity_max

float **m_rapidity_min** = 1e8
    require rapidity > rapidity_min

int **m_truthLabel** = -1
    require truth level on truth jets

bool **m_useHadronConeExcl** = true
    use HadronConeExclTruthLabelID for truth match (default)

bool **m_doJVF** = false
    check JVF

float **m_pt_max_JVF** = 50e3
    max pT [GeV] (JVF is a pileup cut)

float **m_eta_max_JVF** = 2.4
    detector eta cut

float **m_JVFCut** = 0.5
    cut value

bool **m_doJVT** = false
    check JVT

bool **m_noJVTVeto** = false
    keep JVT-rejected jets and decorate passing status

bool **m_dofJVT** = false
    check forward JVT

bool **m_dofJVTVeto** = true
    Remove jets that fail fJVT. Like JVT, the default is to clean the collection.

float **m_pt_max_JVT** = 60e3

max pT [GeV] (JVT is a pileup cut)

float **m_eta_max_JVT** = 2.4

detector eta cut

bool **m_jvtUsedBefore** = false

was JVT already run in an earlier instance of *JetSelector*?

bool **m_haveTruthJets** = true

Does the input have truth jets? If not, cannot decorate with true hard scatter / pileup info.

bool **m_getJVTSF** = true

Retrieve JVT SFs (true by default, when false: allows to get JVT decision w/o needing truth jets)

float **m_JVTCut** = -1.0

Minimum value of JVT for selecting jets.

> **Warning:** If set to a non-negative value (default is -1.0), it will override any set value for `JetSelector::m_WorkingPointJVT`

std::string **m_WorkingPointJVT** = "FixedEffPt"

Available working points for JVT cut from the `CP::IJetJvtEfficiency` tool.

The corresponding data/MC SF will be saved as a `std::vector<float>` decoration (for MC only), for nominal WP and the available systematics.

Value	JVT Cut	Efficiency
"Medium"	(Default) 0.59	92%
"Loose"	0.11	97%
"Tight"	0.91	85%

std::string **m_SFFileJVT** = "DummySFs.root"

Configuration containting JVT scale factors.

The configuration file with the scale factors calculated by the `CP::IJetJvtEfficiency`.

See :https://twiki.cern.ch/twiki/bin/view/AtlasProtected/JVTCalibration for latest recommendation.

std::string **m_outputSystNamesJVT** = "JetJvtEfficiency_JVTSyst"

int **m_JvtTaggingAlg** = CP::JvtTagger::NNJvt

Tagging algorithm to be used to veto PU jets in central region - default in R22 is NNJvt. If another algorithm is needed, use corresponding index for the enum here: https://acode-browser1.usatlas.bnl.gov/lxr/source/athena/PhysicsAnalysis/Interfaces/JetAnalysisInterfaces/JetAnalysisInterfaces/IJetJvtEfficiency.h#0022

(note: this link points to the latest r22 version, i.e. master, if a release is used, please check the corresponding enum for the given release: https://gitlab.cern.ch/atlas/athena/-/tags?search=release%2F22.2&sort=updated_desc)

bool **m_recalculateJvtScores** = true

Do re-calculation of NNJvt - scores need to be re-evaluated in case jet pt changed w.r.t. derivation.

float **m_systValJVT** = 0.0

std::string **m_systNameJVT** = ""

std::string **m_WorkingPointfJVT** = "Loose"

Available working points for fJVT cut from the `CP::IJetJvtEfficiency` tool.

The corresponding data/MC SF will be saved as a `std::vector<float>` decoration (for MC only), for nominal WP and the available systematics.

Value	HS Efficiency	PU Fake Rate
"Medium"	87.1-97.0%	53.4-60.9%
"Tight"	79.9-95.6%	45.4-50.3%

See :https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/FJVTCalibration for more information.

std::string **m_SFFilefJVT** = ""

Configuration containting fJVT scale factors.

The configuration file with the scale factors calculated by the `CP::IJetJvtEfficiency`.

See :https://twiki.cern.ch/twiki/bin/view/AtlasProtected/FJVTCalibration for latest recommendation.

std::string **m_outputSystNamesfJVT** = "JetJvtEfficiency_fJVTSyst"

float **m_systValfJVT** = 0.0

std::string **m_systNamefJVT** = ""

bool **m_fjvtUsedBefore** = false

was fJVT already run in an earlier instance of *JetSelector*?

bool **m_doJetTimingCut** = false

Timing cut.

float **m_jetTiming_max** = -1

bool **m_doBTagCut** = false

Flag to apply btagging cut, if false just decorate decisions.

std::string **m_corrFileName** = "xAODBTaggingEfficiency/cutprofiles_22072015.root"

std::string **m_jetAuthor** = "AntiKt4EMPFlowJets"

std::string **m_taggerName** = "DL1r"

std::string **m_operatingPt** = "FixedCutBEff_70"

double **m_b_eta_max** = 2.5

double **m_b_pt_min** = 20e3

bool **m_doHLTBTagCut** = false

std::string **m_HLTBTagTaggerName** = "DL1r"

float **m_HLTBTagCutValue** = -0.4434

bool **m_requireHLTVtx** = false

bool **m_requireNoHLTVtx** = false

std::string **m_passAuxDecorKeys** = ""

std::string **m_failAuxDecorKeys** = ""

std::string **m_singleJetTrigChains** = ""

A comma-separated string w/ alll the HLT single jet trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

std::string **m_diJetTrigChains** = ""

A comma-separated string w/ all the HLT dijet trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

bool **m_removeDuplicates** = false

remove duplicate jets (exactly the same eta)

int **m_count_events_with_duplicates** = 0

number of events with duplicates

bool **m_sort** = false

sort jets (normally done by *JetCalibrator*, but HLT jets need sorting and don't get calibrated here)

Private Members

int **m_numEvent**

int **m_numObject**

int **m_numEventPass**

int **m_weightNumEventPass**

int **m_numObjectPass**

int **m_pvLocation**

bool **m_isEMjet**

bool **m_isLCjet**

TH1D ***m_cutflowHist** = nullptr

TH1D ***m_cutflowHistW** = nullptr

int **m_cutflow_bin**

std::vector<std::string> **m_passKeys**

std::vector<std::string> **m_failKeys**

TH1D ***m_jet_cutflowHist_1** = nullptr

int **m_jet_cutflow_all**

int **m_jet_cutflow_cleaning_cut**

int **m_jet_cutflow_ptmax_cut**

int **m_jet_cutflow_ptmin_cut**

int **m_jet_cutflow_etmax_cut**

int **m_jet_cutflow_etmin_cut**

int **m_jet_cutflow_eta_cut**

int **m_jet_cutflow_jvt_cut**

int **m_jet_cutflow_timing_cut**

int **m_jet_cutflow_btag_cut**

std::vector<CP::SystematicSet> **m_systListJVT**

std::vector<CP::SystematicSet> **m_systListfJVT**

std::vector<std::string> **m_singleJetTrigChainsList**

std::vector<std::string> **m_diJetTrigChainsList**

/* contains all the HLT trigger chains tokens extracted from m_singleJetTrigChains */

asg::AnaToolHandle<CP::IJetJvtEfficiency> **m_JVT_tool_handle** = {"CP::IJetJvtEfficiency/JVT"}

/* contains all the HLT trigger chains tokens extracted from m_diJetTrigChains */

asg::AnaToolHandle<CP::IJetJvtEfficiency> **m_fJVT_eff_tool_handle** = {"CP::JetJvtEfficiency/fJVT"}

asg::AnaToolHandle<IBTaggingSelectionTool> **m_BJetSelectTool_handle** = {"BTaggingSelectionTool"}

asg::AnaToolHandle<Trig::IMatchingTool> **m_trigJetMatchTool_handle**

asg::AnaToolHandle<Trig::TrigDecisionTool> **m_trigDecTool_handle** =
{"Trig::TrigDecisionTool/TrigDecisionTool"}

asg::AnaToolHandle<Trig::IMatchScoringTool> **m_scoreTool** = {"Trig::DRScoringTool/DRScoringTool"}

bool **m_doTrigMatch** = true

This internal variable gets set to false if no triggers are defined or if TrigDecisionTool is missing.

std::string **m_outputJVTPassed** = "JetJVT_Passed"

std::string **m_outputfJVTPassed** = "JetfJVT_Passed"

μ

class **MuonSelector** : public xAH::*Algorithm*

Public Functions

MuonSelector()

~MuonSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::MuonContainer *inMuons, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::MuonContainer> *selectedMuons)

virtual int **passCuts**(const xAOD::Muon *muon, const xAOD::Vertex *primaryVertex)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_outAuxContainerName**

output auxiliary container name

std::string **m_inputAlgoSystNames** = ""

std::string **m_outputAlgoSystNames** = "MuonSelector_Syst" bool **m_decorateSelectedObjects** = true

decorate selected objects - default “passSel”

bool **m_createSelectedContainer** = false

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1

look at n objects

int **m_pass_min** = -1

minimum number of objects passing cuts

int **m_pass_max** = -1

maximum number of objects passing cuts

float **m_pT_max** = 1e8

require pT < pt_max

float **m_pT_min** = 1e8

require pT > pt_min

bool **m_pT_NaNcheck** = false

check if pT is NaN

std::string **m_muonQualityStr** = "Medium"

require quality

bool **m_isRun3Geo** = false

Switch on Run3 geometry for muon selector tool.

float **m_eta_max** = 1e8

require type require |eta| < eta_max

float **m_d0_max** = 1e8

require d0 < m_d0_max

float **m_d0sig_max** = 1e8

require d0 significance (at BL) < m_d0sig_max

float **m_z0sintheta_max** = 1e8

require z0*sin(theta) (at BL - corrected with vertex info) < m_z0sintheta_max

bool **m_removeCosmicMuon** = false

Remove cosmic muons that fail absolute z0 and d0 selections.

bool **m_removeEventBadMuon** = true

Remove events with a bad muon, defined by poor q/p.

bool **m_doIsolation** = true

enable or disable isolation

std::string **m_MinIsoWPCut** = ""

reject objects which do not pass this isolation cut - default = “” (no cut)

std::string **m_IsoWPList** = "FCTightTrackOnly_FixedRad,FCLoose_FixedRad,FCTight_FixedRad,FixedCutPflowTight,FixedCutPflowLoose"

decorate objects with ‘isIsolated_*’ flag for each WP in this input list - default = all current ASG WPs

std::string **m_CaloIsoEff** = "0.1*x+90"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_TrackIsoEff** = "98"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_CaloBasedIsoType** = "topoetcone20"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_TrackBasedIsoType** = "ptvarcone30"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_singleMuTrigChains** = ""

A comma-separated string w/ alll the HLT single muon trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

std::string **m_diMuTrigChains** = ""

A comma-separated string w/ all the HLT dimuon trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

double **m_minDeltaR** = 0.1

Recommended threshold for muon triggers: see https://svnweb.cern.ch/trac/atlasoff/browser/Trigger/TrigAnalysis/TriggerMatchingTool/trunk/src/TestMatchingToolAlg.cxx.

bool **m_merged_muons** = false

Element links need to be updated if merged muons are used (LRT + std) / false by default.

std::string **m_trigInputPrefix** = ""

Input prefix of trigger decision tool.

bool **m_doLRT** = false

add LRT muon information

std::string **m_isoDecSuffix** = ""

Private Members

int **m_muonQuality**

int **m_numEvent**

int **m_numObject**

int **m_numEventPass**

int **m_weightNumEventPass**

int **m_numObjectPass**

TH1D ***m_cutflowHist** = nullptr

TH1D ***m_cutflowHistW** = nullptr

int **m_cutflow_bin**

bool **m_isUsedBefore** = false

TH1D ***m_mu_cutflowHist_1** = nullptr

TH1D ***m_mu_cutflowHist_2** = nullptr

int **m_mu_cutflow_all**

int **m_mu_cutflow_eta_and_quaility_cut**

int **m_mu_cutflow_ptmax_cut**

int **m_mu_cutflow_ptmin_cut**

int **m_mu_cutflow_ptnan_check**

int **m_mu_cutflow_type_cut**

int **m_mu_cutflow_z0sintheta_cut**

int **m_mu_cutflow_d0_cut**

int **m_mu_cutflow_d0sig_cut**

int **m_mu_cutflow_iso_cut**

int **m_mu_cutflow_cosmic_cut**

std::vector<std::string> **m_IsoKeys**

std::vector<std::string> **m_singleMuTrigChainsList**

std::vector<std::string> **m_diMuTrigChainsList**

/* contains all the HLT trigger chains tokens extracted from m_singleMuTrigChains */

asg::AnaToolHandle<CP::IIsolationSelectionTool> **m_isolationSelectionTool_handle** = {"CP::IsolationSelectionTool/IsolationSelectionTool", this}

/* contains all the HLT trigger chains tokens extracted from m_diMuTrigChains */

CP::IsolationSelectionTool ***m_isolationSelectionTool** = {nullptr}

asg::AnaToolHandle<CP::IMuonSelectionTool> **m_muonSelectionTool_handle** = {"CP::MuonSelectionTool/MuonSelectionTool", this}

asg::AnaToolHandle<Trig::IMatchingTool> **m_trigMuonMatchTool_handle**

asg::AnaToolHandle<Trig::TrigDecisionTool> **m_trigDecTool_handle** = {"Trig::TrigDecisionTool/TrigDecisionTool"}

asg::AnaToolHandle<Trig::IMatchScoringTool> **m_scoreTool** = {"Trig::DRScoringTool/DRScoringTool"}

bool **m_doTrigMatch** = true

This internal variable gets set to false if no triggers are defined or if TrigDecisionTool is missing.

γ

class **PhotonSelector** : public xAH::*Algorithm*

Public Functions

PhotonSelector()

~PhotonSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::PhotonContainer *inPhotons, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::PhotonContainer> *selectedPhotons)

virtual bool **passCuts**(const xAOD::Photon *photon)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

configuration variables input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_inputAlgoSystNames** = ""

output auxiliary container name

std::string **m_outputAlgoSystNames** = "PhotonSelector_Syst"

bool **m_decorateSelectedObjects** = true

decorate selected objects - default "passSel"

bool **m_createSelectedContainer** = true

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1
: look at n objects

int **m_pass_min** = -1
: minimum number of objects passing cuts

int **m_pass_max** = -1
: maximum number of objects passing cuts

float **m_pT_max** = 1e8
: require pT < pt_max

float **m_pT_min** = 1e8
: require pT > pt_min

float **m_eta_max** = 1e8
: require |eta| < eta_max

bool **m_vetoCrack** = true
: require |eta| outside crack region

bool **m_doAuthorCut** = true

bool **m_doOQCut** = true

bool **m_readOQFromDerivation** = false
: read object quality from derivation, rather than calculating it on the fly

std::string **m_photonIdCut** = "None"
: Name of ID variable to cut

std::string **m_MinIsoWPCut** = ""
: reject objects which do not pass this isolation cut - default = “” (no cut)

std::string **m_IsoWPList** = "FixedCutTightCaloOnly,FixedCutTight,FixedCutLoose"
: decorate objects with ‘isIsolated_*’ flag for each WP in this input list - default = all current ASG WPs

Private Members

std::string **m_outAuxContainerName**

int **m_numEvent**

int **m_numObject**

int **m_numEventPass**

int **m_weightNumEventPass**

int **m_numObjectPass**

TH1D *__m_cutflowHist__ = nullptr

TH1D *__m_cutflowHistW__ = nullptr

int **m_cutflow_bin**

TH1D *__m_ph_cutflowHist_1__ = nullptr

int **m_ph_cutflow_all**

int **m_ph_cutflow_author_cut**

int **m_ph_cutflow_OQ_cut**

int **m_ph_cutflow_PID_cut**

int **m_ph_cutflow_ptmax_cut**

int **m_ph_cutflow_ptmin_cut**

int **m_ph_cutflow_eta_cut**

int **m_ph_cutflow_iso_cut**

std::vector<std::string> **m_IsoKeys**

CP::IsolationSelectionTool *__m_IsolationSelectionTool__ = nullptr

T

class **TauSelector** : public xAH::*Algorithm*

Public Functions

TauSelector()

~TauSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::TauJetContainer *inTaus, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::TauJetContainer> *selectedTaus)

virtual int **passCuts**(const xAOD::TauJet *tau)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

std::string **m_outContainerName**

std::string **m_outAuxContainerName**

std::string **m_inputAlgoSystNames** = ""

std::string **m_outputAlgoSystNames** = "TauSelector_Syst"

bool **m_decorateWithTracks** = false

bool **m_decorateSelectedObjects** = true

std::string **m_decorationName** = "passSel"

bool **m_createSelectedContainer** = false

int **m_nToProcess** = -1

int **m_pass_min** = -1

int **m_pass_max** = -1

std::string **m_ConfigPath** = "xAODAnaHelpers/TauConf/00-01-19/Selection/recommended_selection_mc15.conf"

float **m_minPtDAOD** = 15e3

std::string **m_JetIDWP** = ""

std::string **m_EleRNNWP** = ""

bool **m_EleID** = true

std::string **m_singleTauTrigChains** = ""

std::string **m_diTauTrigChains** = ""

Private Members

int **m_numEvent**

int **m_numObject**

int **m_numEventPass**

int **m_weightNumEventPass**

int **m_numObjectPass**

TH1D ***m_cutflowHist** = nullptr

TH1D ***m_cutflowHistW** = nullptr

int **m_cutflow_bin**

bool **m_isUsedBefore** = false

TH1D ***m_tau_cutflowHist_1** = nullptr

TH1D ***m_tau_cutflowHist_2** = nullptr

int **m_tau_cutflow_all**

int **m_tau_cutflow_selected**

std::vector<std::string> **m_singleTauTrigChainsList**

std::vector<std::string> **m_diTauTrigChainsList**

/* contains all the HLT trigger chains tokens extracted from m_singleTauTrigChains */

asg::AnaToolHandle<TauAnalysisTools::ITauSelectionTool> **m_tauSelTool_handle** = {"TauAnalysisTools::TauSelectionTool/TauSelectionTool", this}

/* contains all the HLT trigger chains tokens extracted from m_diTauTrigChains */

asg::AnaToolHandle<Trig::TrigDecisionTool> **m_trigDecTool_handle** = {"Trig::TrigDecisionTool/TrigDecisionTool"}

asg::AnaToolHandle<Trig::IMatchingTool> **m_trigTauMatchTool_handle**

asg::AnaToolHandle<Trig::IMatchScoringTool> **m_scoreTool** = {"Trig::DRScoringTool/DRScoringTool"}

bool **m_doTrigMatch** = true

This internal variable gets set to false if no triggers are defined or if TrigDecisionTool is missing.

Tracks

class **TrackSelector** : public xAH::*Algorithm*

Public Functions

TrackSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

EL::StatusCode **executeTrackCollection**(float mcEvtWeight)

EL::StatusCode **executeTracksInJets**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual int **PassCuts**(const xAOD::TrackParticle *jet, const xAOD::Vertex *pvx)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_inJetContainerName** = ""

input jet container name

bool **m_decorateSelectedObjects** = true

decorate selected objects? defaul passSel

bool **m_createSelectedContainer** = false

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1

look at n objects

int **m_pass_min** = -1
    minimum number of objects passing cuts

int **m_pass_max** = -1
    maximum number of objects passing cuts

std::string **m_cutLevelString** = ""
    available: Loose LoosePrimary TightPrimary LooseMuon LooseElectron MinBias HILoose HITight

float **m_pT_max** = 1e8
    require pT < pt_max

float **m_pT_min** = 1e8
    require pT > pt_max

float **m_p_min** = 1e8
    require |p| > p_min

float **m_eta_max** = 1e8
    require |eta| < eta_max

float **m_eta_min** = 1e8
    require |eta| > eta_min

float **m_etaSigned_min** = 1e8
    require eta > eta_min

float **m_etaSigned_max** = 1e8
    require eta < eta_max

float **m_d0_max** = 1e8
    require |d0| < d0_max

float **m_z0_max** = 1e8
    require |z0| < z0_max

float **m_sigmad0_max** = 1e8
    maximum error on d0

float **m_d0oversigmad0_max** = 1e8
    maximum significance of |d0|

float **m_z0sinT_max** = 1e8
    require |z0xsin(theta)| < z0sintheta_max

float **m_sigmaz0_max** = 1e8
    maximum error on z0

float **m_sigmaz0sintheta_max** = 1e8
    maximum error on z0*sin(theta)

float **m_z0oversigmaz0_max** = 1e8
    max |z0| significance

float **m_z0sinthetaoversigmaz0sintheta_max** = 1e8
    max |z0sin(theta)| significance

int **m_nPixelHits_min** = 1e8
    minimum pixel hits (counting dead sensors)

int **m_nPixelHitsPhysical_min** = 1e8
    minimum pixel hits (no dead sensors)

int **m_nSctHits_min** = 1e8
    minimum SCT hits (counting dead sensors)

int **m_nSctHitsPhysical_min** = 1e8
    minimum SCT hits (no dead sensors)

int **m_nSi_min** = 1e8
    require nSi >= nSi_min (nSi = nPix + nSct)

int **m_nSiPhysical_min** = 1e8
    require nSi >= nSi_min (nSi = nPix + nSct, no dead sensors)

int **m_nPixHoles_max** = 1e8
    require nPixHoles <= nPixHoles_max

int **m_nSctHoles_max** = 1e8
    require nSCTHoles <= nSCTHoles_max

int **m_nSiHoles_max** = 1e8
    maximum silicon holes

int **m_nInnermostPixel_min** = 1e8
    minimum nIBL (if expected)

int **m_nNextToInnermostPixel_min** = 1e8
    minimum nBL (if expected)

int **m_nBothInnermostLayersHits_min** = 1e8
  minimum nIBL + nBL (if every hit that is not expected, we require one less)

int **m_nPixelSharedHits_max** = 1e8
  maximum pixel hits shared with other tracks

int **m_nSctSharedHits_max** = 1e8
  maximum SCT hits shared with other tracks

int **m_nSiSharedHits_max** = 1e8
  maximum silicon hits shared with other tracks

int **m_nSiSharedModules_max** = 1e8
  maximum (pixel + SCT/2) shared hits

float **m_chi2NdofCut_max** = 1e8
  require chi2/ndof < chi2NdofCut_max

float **m_chi2Prob_max** = 1e8
  require TMath::Prob(chi2,ndof) < chi2ProbMax

float **m_chi2Prob_min** = 1e8
  require TMath::Prob(chi2,ndof) > chi2ProbMax

int **m_nBL_min** = 1e8
  require nIBL >= nBL_min (not recommended; for downward compatibility)

std::string **m_passAuxDecorKeys** = ""

std::string **m_failAuxDecorKeys** = ""

bool **m_doTracksInJets** = false
  do track selection on track within jets

Private Members

std::vector<std::string> **m_passKeys**

std::vector<std::string> **m_failKeys**

asg::AnaToolHandle<InDet::IInDetTrackSelectionTool> **m_trkSelTool_handle** = {"InDet::InDetTrackSelectionTool/TrackSelectionTool", this}

int **m_numEvent**

int **m_numObject**

int **m_numEventPass**

int **m_numObjectPass**

TH1D *__m_cutflowHist__ = nullptr

TH1D *__m_cutflowHistW__ = nullptr

int **m_cutflow_bin**

Truth

class **TruthSelector** : public xAH::*Algorithm*

Public Functions

TruthSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual bool **executeSelection**(const xAOD::TruthParticleContainer *inTruthParts, float mcEvtWeight, bool count, std::string outContainerName)

virtual int **PassCuts**(const xAOD::TruthParticle *truthPart)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""
: input container name

std::string **m_outContainerName** = ""
: output container name

std::string **m_decor** = "passSel"
: The decoration key written to passing objects.

bool **m_decorateSelectedObjects** = true
: decorate selected objects? defaul passSel

bool **m_createSelectedContainer** = false
: fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1
: look at n objects

int **m_pass_min** = -1
: minimum number of objects passing cuts

int **m_pass_max** = -1
: maximum number of objects passing cuts

float **m_pT_max** = 1e8
: require pT < pt_max

float **m_pT_min** = 1e8
: require pT > pt_min

float **m_eta_max** = 1e8
: require eta < eta_max

float **m_eta_min** = 1e8
: require eta > eta_max

float **m_mass_max** = 1e8
: require mass < mass_max

float **m_mass_min** = 1e8
: require mass > mass_max

float **m_rapidity_max** = 1e8

require rapidity < rapidity_max

float **m_rapidity_min** = 1e8

require rapidity > rapidity_min

unsigned int **m_type** = 1000

require classifierParticleType == type (defined by TruthClassifier: https://gitlab.cern.ch/atlas/athena/blob/21.2/PhysicsAnalysis/MCTruthClassifier/MCTruthClassifier/MCTruthClassifierDefs.h)

std::string **m_typeOptions**

require classifierParticleType to match any of the "|" separated type values (e.g. "1|2|3|4")

unsigned int **m_origin** = 1000

require classifierParticleOrigin == origin (defined by TruthClassifier: https://gitlab.cern.ch/atlas/athena/blob/21.2/PhysicsAnalysis/MCTruthClassifier/MCTruthClassifier/MCTruthClassifierDefs.h)

std::string **m_originOptions**

require classifierParticleOrigin to match any of the "|" separated origin values (e.g. "10|12|13")

float **m_pT_dressed_min** = 1e8

require pt_dressed > pt_dressed_min

float **m_eta_dressed_min** = 1e8

require eta_dressed > eta_dressed_min

float **m_eta_dressed_max** = 1e8

require eta_dressed > eta_dressed_max

Private Members

int **m_numEvent**

int **m_numObject**

int **m_numEventPass**

int **m_weightNumEventPass**

int **m_numObjectPass**

TH1D ***m_cutflowHist** = nullptr

TH1D ***m_cutflowHistW** = nullptr

int **m_cutflow_bin**

TH1D ***m_truth_cutflowHist_1** = nullptr

int **m_truth_cutflow_all**

int **m_truth_cutflow_ptmax_cut**

int **m_truth_cutflow_ptmin_cut**

int **m_truth_cutflow_eta_cut**

4.8.5 Histograms

There are three generic levels to include when building up an analysis that involves plotting: * HistogramManager * JetHists, ElectronHists, MuonHists, etc… * JetHistsAlgo, ElectronHistsAlgo, MuonHistsAlgo, etc…

In order: *HistogramManager* should rarely be changed. This manages the histograms for you in *EventLoop* algorithms by initializing histograms and adding it to the worker. *JetHists*, etc are plotting classes to pre-define the set of plots you want to use for a given set of objects – as well as how to plot them. Finally, *JetHistsAlgo*, etc… are *EventLoop* algorithms that you would include in your jobs and run to actually apply those plots.

HistogramManager

This is the base class from which all histogram management classes are made for Muons, Jets, Electrons, etcetera. It is meant to be flexible enough for someone to use it to create their own set of histograms to produce for an algorithm from scratch using the class.

In particular, the `book()` functions are overloaded for good reason - they all do the same thing except the number of arguments supplied tells us what kind of histogram you want to make: 1D, 2D, or 3D. All histograms take in a `name` and a `title` which get concatenated to provide the `stored name` of the histogram (`name+title`). If you wish to use TDirectoryFiles automagically, append a forward-slash to the end of the `name`, such as `"AntiKt10/"`. The `book()` function will create the histogram, set up the title, the labels, append it to `m_allHists`, and returns a pointer to the newly created histogram. The last argument is `sumw2` which tells the function whether to enable `sumw2()` for the histogram or not, this defaults to `true`. The order of the arguments are listed in the table.

class **HistogramManager**

This is used by any class extending to pre-define a set of histograms to book by default.

We expect the user to create a new group of histograms, such as for jets:

```
class JetHists : public HistogramManager
{
  public:
    JetHists(std::string name, std::string detailStr);
```

(continues on next page)

(continued from previous page)

```
    virtual ~JetHists() ;

    bool m_debug;
    StatusCode initialize();
    StatusCode execute( const xAOD::JetContainer  jets, float eventWeight, int␣
↪pvLoc = -1);
    StatusCode execute( const xAOD::Jet  jet, float eventWeight, int pvLoc = -1 );
    using HistogramManager::book; // make other overloaded version of book() to␣
↪show up in subclass
    using HistogramManager::execute; // overload
};
```

The above example is taken from our implementation in `JetHists`.

Note: The expectation is that the user does not directly use this class but rather inherits from it.

Subclassed by *MetHists*

Public Types

typedef std::unordered_map<std::string, TH1*> **HistMap_t**

Typedef for convenience.

Public Functions

HistogramManager(std::string name, std::string detailStr)

Initialization.

Parameters

- **name** – The top-level path in which all histograms are stored under (think of `TDirectory`)
- **detailStr** – Specify the various details of which to plot. For example, jets might want `"kinematic substructure"`.

virtual **~HistogramManager**()

Destructor, allows the user to delete histograms that are not being recorded.

inline virtual StatusCode **initialize**()

Initialize and book all histograms.

Example implementation:

```
StatusCode JetHists::initialize() {
  m_jetPt          = book(m_name, "jetPt",  "jet p_{T} [GeV]", 120, 0, 3000.);
  return StatusCode::SUCCESS;
}
```

Note: This should call the overloaded functions *HistogramManager::book()* to create the histograms so that the user can call *hists->record(wk())* to record all histograms to the EventLoop worker.

inline virtual StatusCode **execute**()

Execute by filling in the histograms.

Example implementation:

```
StatusCode JetHists::execute( const xAOD::JetContainer  jets, float eventWeight␣
↪){
  for(const auto& jet:  jets)
    m_jetPt->Fill( jet->pt()/1.e3, eventWeight );
  return StatusCode::SUCCESS;
}
```

inline virtual StatusCode **finalize**()

Finalize anything that needs to be finalized.

Warning: This should rarely be used. There is not a good use case for this functionality but it needs to exist in the off-chance that a user comes along and needs it for their histogram class.

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh)

record a histogram and call various functions

Note: This is an overloaded function. It will build the right histogram given the correct number of input arguments.

Parameters

- **name** – name of histogram, access it in ROOT file like `h_jetPt->Draw()`
- **title** – usually pointless,put a description of the histogram in here
- **xlabel** – label to put on the x-axis
- **xbins** – number of xbins to use
- **xlow** – lower bound on xbins
- **xhigh** – upper bound on xbins
- **xbinsArr** – variable xbins, test math (x_1, y_1) and (x_2, y_2)
- **ylabel** – label to put on the y-axis
- **ylow** – lower bound on ybins
- **yhigh** – upper bound on ybins
- **ybinsArr** – variable ybins
- **zlabel** – label to put on the z-axix
- **zlow** – lower bound on zbins

- **zhigh** – upper bound on zbins
- **zbinsArr** – variable zbins

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string xyabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, double ylow, double yhigh, std::string zlabel, int zbins, double zlow, double zhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr, std::string zlabel, int zbins, const Double_t *zbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, double ylow, double yhigh, std::string option = "")

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, const Double_t *xbinsArr, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, double xlow, double xhigh, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

void **record**(EL::IWorker *wk)
record all histograms from *HistogramManager::m_allHists* to the worker

MsgStream &**msg**() const
the standard message stream for this algorithm

MsgStream &**msg**(int level) const
allow ANA_MSG_XXXX macros to be used within algorithms for a given level

TH1 ***findHist**(const std::string &histName)
Return the pointer to the histogram.

void **fillHist**(const std::string &histName, double value)
Fill a histogram by name. Can be overloaded with weight.

Parameters

- **histName** – The name of the histogram to be filled
- **value** – The value to fill the histogram with

void **fillHist**(const std::string &histName, double value, double weight)
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

void **fillHist**(const std::string &histName, double valueX, double valueY, double weight)

void **fillHist**(const std::string &histName, double valueX, double valueY, double valueZ, double weight)

void **fillProfile**(const std::string &histName, double valueX, double valueY, double weight)

Public Members

HistMap_t **m_histMap**
The map of histogram names to their pointers.

Protected Attributes

std::string **m_name**
generically the main name assigned to all histograms

std::string **m_detailStr**
a detail level in the form of a string

std::vector<TH1*> **m_allHists**
a container holding all generated histograms

mutable MsgStream **m_msg**
hold the MsgStream object

Private Functions

void **Sumw2**(TH1 *hist, bool flag = true)

Turn on Sumw2 for the histogram.

Parameters

- **hist** – The histogram to modify
- **flag** – Pass in whether to turn on Sumw2 or not

void **record**(TH1 *hist)

Push the new histogram to *HistogramManager::m_allHists* and add its name to *HistogramManager::m_histMap*.

void **SetLabel**(TH1 *hist, std::string xlabel)

Set the labels on a histogram.

Parameters

- **hist** – The histogram to set the labels on
- **xlabel** – The xlabel to set
- **ylabel** – The ylabel to set
- **zlabel** – The zlabel to set

void **SetLabel**(TH1 *hist, std::string xlabel, std::string ylabel)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

void **SetLabel**(TH1 *hist, std::string xlabel, std::string ylabel, std::string zlabel)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Classes

This is a class that predefines all the histograms, defines the *execute* function which fills in the histograms for you, given an object or a collection of objects, and handles a lot of other logic. This class extends *HistogramManager*.

ClusterHists

> **Warning:** doxygenclass: Cannot find class "ClusterHists" in doxygen xml output for project "xAH" from directory: ./doxygen/xml

JetHists

> **Warning:** doxygenclass: Cannot find class "JetHists" in doxygen xml output for project "xAH" from directory: ./doxygen/xml

MetHists

class **MetHists** : public *HistogramManager*

Public Functions

MetHists(std::string name, std::string detailStr)

virtual **~MetHists**()

virtual StatusCode **initialize**()

Initialize and book all histograms.

Example implementation:

```
StatusCode JetHists::initialize() {
  m_jetPt          = book(m_name, "jetPt",  "jet p_{T} [GeV]", 120, 0, 3000.);
  return StatusCode::SUCCESS;
}
```

Note: This should call the overloaded functions `HistogramManager::book()` to create the histograms so that the user can call *hists->record(wk())* to record all histograms to the EventLoop worker.

StatusCode **execute**(const xAOD::MissingETContainer *met, float eventWeight)

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh)

record a histogram and call various functions

Note: This is an overloaded function. It will build the right histogram given the correct number of input arguments.

Parameters

- **name** – name of histogram, access it in ROOT file like `h_jetPt->Draw()`
- **title** – usually pointless,put a description of the histogram in here
- **xlabel** – label to put on the x-axis
- **xbins** – number of xbins to use
- **xlow** – lower bound on xbins
- **xhigh** – upper bound on xbins
- **xbinsArr** – variable xbins, test math (x_1, y_1) and (x_2, y_2)
- **ylabel** – label to put on the y-axis
- **ylow** – lower bound on ybins
- **yhigh** – upper bound on ybins
- **ybinsArr** – variable ybins
- **zlabel** – label to put on the z-axix

- `zlow` – lower bound on zbins
- `zhigh` – upper bound on zbins
- `zbinsArr` – variable zbins

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string xyabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, double ylow, double yhigh, std::string zlabel, int zbins, double zlow, double zhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr, std::string zlabel, int zbins, const Double_t *zbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, double ylow, double yhigh, std::string option = "")

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, const Double_t *xbinsArr, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, double xlow, double xhigh, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

inline virtual StatusCode **execute**()

Execute by filling in the histograms.

Example implementation:

```
StatusCode JetHists::execute( const xAOD::JetContainer  jets, float eventWeight␣
↪){
  for(const auto& jet:  jets)
    m_jetPt->Fill( jet->pt()/1.e3, eventWeight );
  return StatusCode::SUCCESS;
}
```

Public Members

bool **m_debug**

Protected Attributes

HelperClasses::*METInfoSwitch* ***m_infoSwitch**

Private Members

TH1F ***m_metFinalClus**

TH1F ***m_metFinalClusPx**

TH1F ***m_metFinalClusPy**

TH1F ***m_metFinalClusSumEt**

TH1F ***m_metFinalClusPhi**

TH1F ***m_metFinalTrk**

TH1F ***m_metFinalTrkPx**

TH1F ***m_metFinalTrkPy**

TH1F ***m_metFinalTrkSumEt**

TH1F ***m_metFinalTrkPhi**

MuonHists

> **Warning:** doxygenclass: Cannot find class "MuonHists" in doxygen xml output for project "xAH" from directory: ./doxygen/xml

TrackHists

> **Warning:** doxygenclass: Cannot find class "TrackHists" in doxygen xml output for project "xAH" from directory: ./doxygen/xml

VtxHists

> **Warning:** doxygenclass: Cannot find class "VtxHists" in doxygen xml output for project "xAH" from directory: ./doxygen/xml

Algorithms

This is an EL Algorithm that incorporates the correspondingly-named class.

ClusterHistsAlgo

class **ClusterHistsAlgo** : public xAH::*Algorithm*

Public Functions

ClusterHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_detailStr** = ""

Private Members

ClusterHists ***m_plots** = nullptr

JetHistsAlgo

class **JetHistsAlgo** : public *IParticleHistsAlgo*

Public Functions

JetHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **execute**()
: Calls execute<IParticleContainer>

virtual EL::StatusCode **AddHists**(std::string name)
: Calls AddHists<IParticleHists>

Parameters **name** – Name of the systematic

MetHistsAlgo

class **MetHistsAlgo** : public xAH::*Algorithm*

Public Functions

MetHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_detailStr** = ""

Private Members

MetHists ***m_plots** = nullptr

MuonHistsAlgo

class **MuonHistsAlgo** : public *IParticleHistsAlgo*

Public Functions

MuonHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **execute**()

Calls execute<IParticleContainer>

virtual EL::StatusCode **AddHists**(std::string name)

Calls AddHists<IParticleHists>

Parameters **name** – Name of the systematic

TrackHistsAlgo

class **TrackHistsAlgo** : public xAH::*Algorithm*

Public Functions

TrackHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_detailStr** = ""

Private Members

TrackHists ***m_plots** = nullptr

4.8.6 Tools Used

This page provides a list of all the tools used in the framework and where to find documentation for that particular tool (eg: their twiki page). See this twiki for more general details.

Event Level

- PileupReweightingTool
- TriggerDecisionTool

e and γ

- IsolationSelectionTool
- ElectronPhotonFourMomentumCorrection
- ElectronPhotonSelectorTools
- IsolationCorrectionTool

μ

- MuonSelectionTool

j

- JetCalibrationTool
- JERSmearingTool
- JetSelectorTools
- JVT
- BTaggingEfficiencyTool
- JetCleaning2016

τ jets

- TauAnalysisTools

4.8.7 Producing Outputs

TTree Outputs

Tree Maker Base Class

class **HelpTreeBase**

Public Functions

HelpTreeBase(xAOD::TEvent *event, TTree *tree, TFile *file, const float units = 1e3, bool debug = false, xAOD::TStore *store = nullptr, std::string nominalTreeName = "nominal")

HelpTreeBase(TTree *tree, TFile *file, xAOD::TEvent *event = nullptr, xAOD::TStore *store = nullptr, const float units = 1e3, bool debug = false, std::string nominalTreeName = "nominal")

virtual **~HelpTreeBase**()

void **AddEvent**(const std::string &detailStr = "")

void **AddTrigger**(const std::string &detailStr = "")

void **AddJetTrigger**(const std::string &detailStr = "")

void **AddMuons**(const std::string &detailStr = "", const std::string &muonName = "muon")

void **AddElectrons**(const std::string &detailStr = "", const std::string &elecName = "el")

void **AddPhotons**(const std::string &detailStr = "", const std::string &photonName = "ph")

void **AddClusters**(const std::string &detailStr = "", const std::string &clusterName = "cl")

void **AddJets**(const std::string &detailStr = "", const std::string &jetName = "jet")

void **AddL1Jets**(const std::string &jetName = "")

void **AddTruthParts**(const std::string &detailStr = "", const std::string &truthName = "xAH_truth")

void **AddTrackParts**(const std::string &detailStr = "", const std::string &trackName = "trk")

void **AddVertices**(const std::string &detailStr = "", const std::string &vertexName = "vertex")

void **AddTruthVertices**(const std::string &detailStr = "", const std::string &vertexName = "truth_vertex")

void **AddFatJets**(const std::string &detailStr = "", const std::string &fatjetName = "fatjet", const std::string &subjetDetailStr = "", const std::string &suffix = "")

Declare a new collection of fatjets to be written to the output tree.

Parameters

- **detailStr** – A (space-separated) list of detail options. These keywords specify exactly which information about each jet is written out. Current influential options are: `kinematic` `substructure` `constituent` `constituentAll`
- **fatjetName** – The (prefix) name of the container. Default: `fatjet`.
- **subjetDetailStr** – List of detail options to pass to the subjet container. See :cpp:member:`HelpTreeBase::AddJets` for list of supported values.

void **AddTruthFatJets**(const std::string &detailStr = "", const std::string &truthFatJetName = "truth_fatjet")

void **AddTaus**(const std::string &detailStr = "", const std::string &tauName = "tau")

void **AddMET**(const std::string &detailStr = "", const std::string &metName = "met")

void **FillEvent**(const xAOD::EventInfo *eventInfo, xAOD::TEvent *event = nullptr, const xAOD::VertexContainer *vertices = nullptr)

void **FillTrigger**(const xAOD::EventInfo *eventInfo)

void **FillJetTrigger**()

void **FillMuons**(const xAOD::MuonContainer *muons, const xAOD::Vertex *primaryVertex, const std::string &muonName = "muon")

void **FillMuon**(const xAOD::Muon *muon, const xAOD::Vertex *primaryVertex, const std::string &muonName = "muon")

void **FillElectrons**(const xAOD::ElectronContainer *electrons, const xAOD::Vertex *primaryVertex, const std::string &elecName = "el") void **FillElectron**(const xAOD::Electron *elec, const xAOD::Vertex *primaryVertex, const std::string &elecName = "el")

void **FillPhotons**(const xAOD::PhotonContainer *photons, const std::string &photonName = "ph")

void **FillPhoton**(const xAOD::Photon *photon, const std::string &photonName = "ph")

void **FillClusters**(const xAOD::CaloClusterContainer *clusters, const std::string &clusterName = "cl")

void **FillCluster**(const xAOD::CaloCluster *cluster, const std::string &clusterName = "cl")

void **FillJets**(const xAOD::JetContainer *jets, int pvLocation = -1, const std::string &jetName = "jet")

void **FillJet**(const xAOD::Jet *jet_itr, const xAOD::Vertex *pv, int pvLocation, const std::string &jetName = "jet")

void **FillLegacyL1Jets**(const xAOD::JetRoIContainer *jets, const std::string &jetName = "L1Jet", bool sortL1Jets = false)

template<typename **T**>
inline void **FillPhase1L1Jets**(*T* *&jets, const std::string &jetName = "L1Jet", bool sortL1Jets = false)

void **FillTruth**(const xAOD::TruthParticleContainer *truth, const std::string &truthName = "xAH_truth")

void **FillTruth**(const xAOD::TruthParticle *truthPart, const std::string &truthName)

void **FillTracks**(const xAOD::TrackParticleContainer *tracks, const std::string &trackName = "trk")

void **FillTrack**(const xAOD::TrackParticle *trackPart, const std::string &trackName)

void **FillVertices**(const xAOD::VertexContainer *vertices, const std::string &vertexName = "vertex")

void **FillTruthVertices**(const xAOD::TruthVertexContainer *truthVertices, const std::string &truthVertexName = "truth_vertex")

void **FillFatJets**(const xAOD::JetContainer *fatJets, int pvLocation = 0, const std::string &fatjetName = "fatjet", const std::string &suffix = "")

Write a container of jets to the specified container name (and optionally suffix). The container name and suffix should be declared beforehand using *AddFatJets()*. This clears the current branch state for the collection so it only makes sense to call once per call to `Fill()`.

Parameters

- **fatJets** – A container of jets to be written out.
- **fatjetName** – The name of the output collection to write to.
- **suffix** – The suffix of the output collection to write to.

void **FillFatJet**(const xAOD::Jet *fatjet_itr, int pvLocation = 0, const std::string &fatjetName = "fatjet", const std::string &suffix = "")

void **FillTruthFatJets**(const xAOD::JetContainer *truthFatJets, int pvLocation = 0, const std::string &truthFatJetName = "truth_fatjet")

void **FillTruthFatJet**(const xAOD::Jet *truth_fatjet_itr, int pvLocation = 0, const std::string &truthFatJetName = "truth_fatjet")

void **FillTaus**(const xAOD::TauJetContainer *taus, const std::string &tauName = "tau") void **FillTau**(const xAOD::TauJet *tau, const std::string &tauName = "tau")

void **FillMET**(const xAOD::MissingETContainer *met, const std::string &metName = "met")

void **Fill**()

void **ClearEvent**()

void **ClearTrigger**()

void **ClearJetTrigger**()

void **ClearMuons**(const std::string &jetName = "muon")

void **ClearElectrons**(const std::string &elecName = "el")

void **ClearPhotons**(const std::string &photonName = "ph")

void **ClearClusters**(const std::string &clusterName = "cl")

void **ClearJets**(const std::string &jetName = "jet")

void **ClearL1Jets**(const std::string &jetName = "L1Jet")

void **ClearTruth**(const std::string &truthName)

void **ClearTracks**(const std::string &trackName)

void **ClearFatJets**(const std::string &fatjetName, const std::string &suffix = "")

void **ClearTruthFatJets**(const std::string &truthFatJetName = "truth_fatjet")

void **ClearTaus**(const std::string &tauName = "tau")

void **ClearMET**(const std::string &metName = "met")

void **ClearVertices**(const std::string &vertexName = "vertex")

void **ClearTruthVertices**(const std::string &vertexName = "truth_vertex")

bool **writeTo**(TFile *file)

inline virtual void **AddEventUser**(const std::string &detailStr = "")

inline virtual void **AddTriggerUser**(const std::string &detailStr = "")

inline virtual void **AddJetTriggerUser**(const std::string &detailStr = "")

inline virtual void **AddMuonsUser**(const std::string &detailStr = "", const std::string &muonName = "muon")

inline virtual void **AddElectronsUser**(const std::string &detailStr = "", const std::string &elecName = "el")

inline virtual void **AddPhotonsUser**(const std::string &detailStr = "", const std::string &photonName = "ph")

inline virtual void **AddClustersUser**(const std::string &detailStr = "", const std::string &clusterName = "cl")

inline virtual void **AddJetsUser**(const std::string &detailStr = "", const std::string &jetName = "jet")

inline virtual void **AddTruthUser**(const std::string &truthName = "", const std::string &detailStr = "xAH_truth") inline virtual void **AddTracksUser**(const std::string &trackName = "", const std::string &detailStr = "trk")

inline virtual void **AddFatJetsUser**(const std::string &detailStr = "", const std::string &fatjetName = "", const std::string &suffix = "")

Declare a new fat jet collection. Automatically called once per call to *AddFatJets()*; override this if you want to provide your own additional branches for fatjets.

Parameters

- **detailStr** – The space-separated list of detail requested by the called.
- **fatjetName** – The (prefix) name of the output collection.
- **suffix** – A suffix to be appeneded to the end of the output branch name(s).

inline virtual void **AddTruthFatJetsUser**(const std::string &detailStr = "", const std::string &truthFatJetName = "truth_fatjet")

inline virtual void **AddTausUser**(const std::string &detailStr = "", const std::string &tauName = "tau")

inline virtual void **AddMETUser**(const std::string &detailStr = "", const std::string &metName = "met")

inline virtual void **ClearEventUser**()

inline virtual void **ClearTriggerUser**()

inline virtual void **ClearMuonsUser**(const std::string&)

inline virtual void **ClearElectronsUser**(const std::string&)

inline virtual void **ClearPhotonsUser**(const std::string&)

inline virtual void **ClearClustersUser**(const std::string&)

inline virtual void **ClearTruthUser**(const std::string&)

inline virtual void **ClearTracksUser**(const std::string&)

inline virtual void **ClearJetsUser**(const std::string&)

inline virtual void **ClearFatJetsUser**(const std::string&, const std::string&)

inline virtual void **ClearTruthFatJetsUser**(const std::string&)

inline virtual void **ClearTausUser**(const std::string&)

inline virtual void **ClearMETUser**(const std::string&)

inline virtual void **FillEventUser**(const xAOD::EventInfo*)

inline virtual void **FillMuonsUser**(const xAOD::Muon*, const std::string&, const xAOD::Vertex*)

inline virtual void **FillElectronsUser**(const xAOD::Electron*, const std::string&, const xAOD::Vertex*)

inline virtual void **FillPhotonsUser**(const xAOD::Photon*, const std::string&)

inline virtual void **FillClustersUser**(const xAOD::CaloCluster*, const std::string&)

inline virtual void **FillJetsUser**(const xAOD::Jet*, const std::string&)

inline virtual void **FillTruthUser**(const xAOD::TruthParticle*, const std::string&)

inline virtual void **FillTracksUser**(const xAOD::TrackParticle*, const std::string&)

inline virtual void **FillFatJetsUser**(const xAOD::Jet*, int, const std::string&, const std::string&)

Called once per call to *FillFatJets()*.Ooverride this if you want to any additional information to your jet collection.

Parameters

- **jet** – a pointer to the current xAOD::Jet object that should be written to the output branch(s).
- **fatjetName** – the (prefix) name of the output collection
- **suffix** – the suffix to append to output branches.

inline virtual void **FillTruthFatJetsUser**(const xAOD::Jet*, int, const std::string&)

inline virtual void **FillTausUser**(const xAOD::TauJet*, const std::string&)

inline virtual void **FillMETUser**(const xAOD::MissingETContainer*, const std::string&)

inline virtual void **FillTriggerUser**(const xAOD::EventInfo*)

inline virtual void **FillJetTriggerUser**()

Public Members

xAOD::TEvent ***m_event**

xAOD::TStore ***m_store**

std::string **m_vertexContainerName** = "PrimaryVertices"

Name of vertex container.

std::string **m_truthVertexContainerName** = "TruthVertices"

HelperClasses::*TriggerInfoSwitch* ***m_trigInfoSwitch**

std::string **m_triggerSelection**

TrigConf::xAODConfigTool ***m_trigConfTool**

Trig::TrigDecisionTool ***m_trigDecTool**

Public Static Functions

static std::string **FatJetCollectionName**(const std::string &fatjetName = "fatjet", const std::string &suffix = "")

Helper function to lookup each fatjet container name/suffix combo in the internal map of vectors for vectors. You probably don't need this but it might be useful if you're implementing `[Add/Fill/Clear]FatJetsUser()`.

Parameters

- **fatjetName** – The (prefix) name of the container.
- **suffix** – The container branch suffix.

Returns a string that uniquely identifies the collection name/suffix in the lookup map.

Protected Functions

template<typename **T**, typename **U**, typename **V**>
void **safeFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<*T*> &accessor, std::vector<*U*> &destination, *U* defaultValue, int m_units = 1)

template<typename **T**, typename **U**, typename **V**>
void **safeVecFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<std::vector<*T*>> &accessor, std::vector<std::vector<*U*>> &destination, int m_units = 1)

template<typename **T**>
void **setBranch**(std::string prefix, std::string varName, std::vector<*T*> *localVectorPtr)

Protected Attributes

TTree ***m_tree**

int **m_units**

bool **m_debug**

bool **m_isMC**

std::string **m_nominalTreeName**

bool **m_nominalTree**

xAH::*EventInfo* ***m_eventInfo**

int **m_passL1**

int **m_passHLT**

unsigned int **m_masterKey**

unsigned int **m_L1PSKey**

unsigned int **m_HLTPSKey**

std::vector<std::string> **m_elTrigForMatching**

std::vector<std::string> **m_passedTriggers**

std::vector<std::string> **m_disabledTriggers**

std::vector<float> **m_triggerPrescales**

std::vector<float> **m_triggerPrescalesLumi**

std::vector<std::string> **m_isPassBitsNames**

std::vector<unsigned int> **m_isPassBits**

std::map<std::string, xAH::*JetContainer**> **m_jets**

std::map<std::string, xAH::*L1JetContainer**> **m_l1Jets**

std::map<std::string, xAH::*TruthContainer**> **m_truth**

std::map<std::string, xAH::*TrackContainer**> **m_tracks**

std::map<std::string, xAH::*FatJetContainer**> **m_fatjets**

std::map<std::string, xAH::*FatJetContainer**> **m_truth_fatjets**

std::map<std::string, xAH::*MuonContainer**> **m_muons**

std::map<std::string, std::vector<std::string>> **m_MuonRecoEff_SF_sysNames**

std::map<std::string, std::vector<std::string>> **m_MuonIsoEff_SF_sysNames**

std::map<std::string, std::map<std::string, std::vector<std::string>>> **m_MuonTrigEff_SF_sysNames**

std::vector<std::string> **m_MuonTTVAEff_SF_sysNames** std::map<std::string, xAH::*ElectronContainer**> **m_elecs**

std::map<std::string, xAH::*PhotonContainer**> **m_photons**

std::map<std::string, xAH::*ClusterContainer**> **m_clusters**

std::map<std::string, xAH::*TauContainer**> **m_taus**

std::map<std::string, xAH::*MetContainer**> **m_met**

std::map<std::string, xAH::*VertexContainer**> **m_vertices**

std::map<std::string, xAH::*VertexContainer**> **m_truth_vertices**

Tree Maker Algorithm

class **TreeAlgo** : public xAH::*Algorithm*

Public Functions

TreeAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual *HelpTreeBase* ***createTree**(xAOD::TEvent *event, TTree *tree, TFile *file, const float units, bool debug, xAOD::TStore *store)

Public Members

bool **m_outHistDir** = false

std::string **m_evtDetailStr** = ""

std::string **m_trigDetailStr** = ""

std::string **m_muDetailStr** = ""

std::string **m_elDetailStr** = ""

std::string **m_jetDetailStr** = ""

std::string **m_trigJetDetailStr** = ""

std::string **m_truthJetDetailStr** = ""

std::string **m_fatJetDetailStr** = ""

std::string **m_truthFatJetDetailStr** = ""

std::string **m_tauDetailStr** = ""

std::string **m_METDetailStr** = ""

std::string **m_METReferenceDetailStr** = ""

std::string **m_photonDetailStr** = ""

std::string **m_clusterDetailStr** = ""

std::string **m_truthParticlesDetailStr** = ""

std::string **m_trackParticlesDetailStr** = ""

std::string **m_vertexDetailStr** = ""

std::string **m_evtContainerName** = ""

std::string **m_muContainerName** = ""

std::string **m_elContainerName** = ""

std::string **m_jetContainerName** = ""

std::string **m_jetBranchName** = "jet"

std::string **m_truthJetContainerName** = ""

std::string **m_truthJetBranchName** = "truthJet"

std::string **m_trigJetContainerName** = ""

std::string **m_trigJetBranchName** = "trigJet"

std::string **m_fatJetContainerName** = ""

std::string **m_fatJetBranchName** = ""

std::string **m_truthFatJetContainerName** = ""

std::string **m_truthFatJetBranchName** = "truth_fatjet"

std::string **m_tauContainerName** = ""

std::string **m_METContainerName** = ""

std::string **m_METReferenceContainerName** = ""

std::string **m_photonContainerName** = ""

std::string **m_clusterContainerName** = ""

std::string **m_clusterBranchName** = "CaloCalTopoClusters"

std::string **m_truthParticlesContainerName** = ""

std::string **m_truthParticlesBranchName** = "xAH_truth"

std::string **m_trackParticlesContainerName** = ""

std::string **m_l1JetContainerName** = ""

std::string **m_l1JetBranchName** = "L1Jet"

std::string **m_vertexBranchName** = "vertex"

bool **m_sortL1Jets** = false

bool **m_retrievePV** = true

std::string **m_muSystsVec** = ""

std::string **m_elSystsVec** = ""

std::string **m_tauSystsVec** = ""

std::string **m_jetSystsVec** = ""

std::string **m_photonSystsVec** = ""

std::string **m_fatJetSystsVec** = ""

std::string **m_metSystsVec** = ""

float **m_units** = 1e3

unit conversion from MeV, default is GeV

int **m_autoFlush** = 0

Set to a large negative number, such as -1000000, to ensure that the tree flushes memory after a reasonable amount of time. Otherwise, jobs with a lot of systematics use too much memory.

Protected Attributes

std::vector<std::string> **m_jetDetails**

std::vector<std::string> **m_trigJetDetails**

std::vector<std::string> **m_fatJetDetails**

std::vector<std::string> **m_jetContainers**

std::vector<std::string> **m_truthJetContainers**

std::vector<std::string> **m_trigJetContainers** std::vector<std::string> **m_fatJetContainers**

std::vector<std::string> **m_l1JetContainers**

std::vector<std::string> **m_vertexContainers**

std::vector<std::string> **m_truthParticlesContainers**

std::vector<std::string> **m_jetBranches**

std::vector<std::string> **m_truthJetBranches**

std::vector<std::string> **m_trigJetBranches**

std::vector<std::string> **m_fatJetBranches**

std::vector<std::string> **m_l1JetBranches**

std::vector<std::string> **m_vertexBranches**

std::vector<std::string> **m_truthParticlesBranches**

std::vector<std::string> **m_clusterDetails**

std::vector<std::string> **m_clusterContainers**

std::vector<std::string> **m_clusterBranches**

std::vector<std::string> **m_vertexDetails**

std::map<std::string, *HelpTreeBase**> **m_trees**

xAOD Outputs

Mini-xAOD

class **MinixAOD** : public xAH::*Algorithm*

Produce xAOD outputs.

I can think up the following cases when a user is doing an EL Algorithm:

input containers in TEvent (simple) deep-copied containers in TStore (deep-copy) shallow-copied containers in TStore (shallow) CDV containers in TStore (cdv)

For the above use-cases, we might produce outputs like so:

write the input container to the output. This uses `TEvent::copy()`. write the deep-copied containers to the output. This calls `TStore::retrieve()` and then `TEvent::record()`. two options when we have shallow-copies:

1. `shallowIO=false`: write to the output as a deep-copy like in the previous option
2. `shallowIO=true`: write to the output as a shallow-copy, but make sure the original container is also written to the output

make a deep-copy of the ConstDataVector and then move from `TStore` to `TEvent`. The problem is that we point to local memory that will not persist when making the CDV.

The trickiest case is with shallow copies because those could be our systematics – and you might want to copy the original container, and only copy over systematics via true shallow copies to conserve memory and space.

> **Warning:** Care must be taken when managing memory and using copies. You need to think about how copies point to each other and whether you can use shallow copies or deep copies or both.

Public Functions

`MinixAOD`()

virtual EL::StatusCode **`setupJob`**(EL::Job &job)

virtual EL::StatusCode **`fileExecute`**()

virtual EL::StatusCode **`histInitialize`**()

virtual EL::StatusCode **`changeInput`**(bool firstFile)

virtual EL::StatusCode **`initialize`**()

virtual EL::StatusCode **`execute`**()

virtual EL::StatusCode **`postExecute`**()

virtual EL::StatusCode **`finalize`**()

virtual EL::StatusCode **`histFinalize`**()

Public Members

std::string **`m_outputFileName`** = "out_miniXAOD"

name of the output file to use for xAOD dumping

bool **`m_createOutputFile`** = true

enable to create the output file for xAOD dumping

bool **`m_copyFileMetaData`** = false

copy the file metadata over

bool **m_copyTriggerInfo** = false

copy the trigger containers and meta data over

bool **m_copyCutBookkeeper** = false

copy the cutbookkeeper data over

std::string **m_simpleCopyKeys** = ""

names of containers to copy from the input file

Container names should be space-delimited:

```
"m_simpleCopyKeys": "EventInfo AntiKt4EMTopoJets"
```

std::string **m_storeCopyKeys** = ""

names of containers in the TStore to copy over

Container names should be space-delimited:

```
"m_storeCopyKeys": "BrandNewJetContainer ReclusteredJets"
```

Note: This option is appropriate for deep-copied containers.

std::string **m_shallowCopyKeys** = ""

names of containers that have been shallow-copied

This option is a little different because shallow-copied containers have parent containers. However, there are two options depending on the `setShallowIO` option

True If this is set to true, you will want to specify the parent container so that we copy it over as well (it is assumed that the parent container is in TStore or TEvent):

```
"m_shallowCopyKeys": "SCAntiKt4EMTopoJets|AntiKt4EMTopoJets␣
↪SCMuons|Muons_Presel"
```

False If this is set to false, you will not want to specify the parent container

”m_shallowCopyKeys”: “SCAntiKt4EMTopoJets| SCMuons|”

Always specify your string in a space-delimited format where pairs are split up by `shallow container name|parent container name`.

Note: This option is appropriate for shallow-copied containers.

> **Warning:** Please note that the `shallowIO` option is what determines how the memory is managed. If you run into issues with shallow-copied containers here, make sure you know whether this option was enabled or not before asking for help.

std::string **`m_deepCopyKeys`** = ""

names of containers that have been shallow-copied

Here, we will do the deep-copying for you, so that the containers can be correctly recorded into the output. Due to the way view-only containers work, we can't figure out whether the memory points to a specific parent container we can copy, or to a non-persistable, local (stack) memory. The best option is to just deep-copy and allocate new memory instead:

```
"m_deepCopyKeys": "AntiKt4EMTopoJets|DeepCopyAntiKt4Jets Muons|DeepCopyMuons"
```

Always specify your string in a space-delimited format where pairs are split up by `input container name|output container name`.

> **Note:** This option is appropriate for view-only containers such as `ConstDataVector`.

std::string **`m_vectorCopyKeys`** = ""

names of vectors that have container names for its contents

Here, we will do the copying for you by retrieving the vector of container names and copy each one over. See how *`MinixAOD::m_shallowCopyKeys`* works.

Always specify your string in a space-delimited format where pairs are split up by `vector name|parent container name`.

> **Note:** This option is appropriate for groups shallow-copied containers such as when you are dealing with systematics.

Private Members

std::vector<std::string> **`m_simpleCopyKeys_vec`**

A vector of containers that are in TEvent that just need to be written to the output.

std::vector<std::pair<std::string, std::string>> **`m_shallowCopyKeys_vec`**

A vector of (container name, parent name) pairs for shallow-copied objects — if parent is empty, deep-copy it.

std::vector<std::pair<std::string, std::string>> **`m_deepCopyKeys_vec`**

A vector of (in container, output container) that need to be deep-copied first before moving to TStore.

std::vector<std::pair<std::string, std::string>> **m_vectorCopyKeys_vec**

A vector of (name of vector of container names, parent name) pairs for shallow-copied objects (like systematics) — if parent is empty, deep-copy it.

std::vector<std::string> **m_copyFromStoreToEventKeys_vec**

A vector of containers (and aux-pairs) in TStore to record in TEvent.

xAODMaker::FileMetaDataTool ***m_fileMetaDataTool** = nullptr

Pointer for the File MetaData Tool.

xAOD::CutBookkeeperContainer ***m_outputCBKContainer** = nullptr

Pointer for the TriggerMenu MetaData Tool.

Pointer for our CutBookkeeper

xAOD::CutBookkeeperAuxContainer ***m_outputCBKContainer_aux** = nullptr

xAOD::CutBookkeeperContainer ***m_outputInCBKContainer** = nullptr

xAOD::CutBookkeeperAuxContainer ***m_outputInCBKContainer_aux** = nullptr

xAOD::CutBookkeeper ***m_outputCBK** = nullptr

4.8.8 Utilities

Debug Tool

class **DebugTool** : public xAH::*Algorithm*

Public Functions

DebugTool()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

bool **m_printStore** = false

Helper Classes

namespace **HelperClasses**

Enums

enum class **ContainerType**

Values:

enumerator **UNKNOWN**

enumerator **CONSTDV**

enumerator **CONSTCONT**

enum class **ToolName**

Values:

enumerator **MUONSELECTOR**

enumerator **ELECTRONSELECTOR**

enumerator **PHOTONSELECTOR**

enumerator **JETSELECTOR**

enumerator **BJETSELECTOR**

enumerator **CALIBRATOR**

enumerator **CORRECTOR**

enumerator **SELECTOR**

enumerator **DEFAULT**

template<typename **T**>

class **EnumParser**

#include <HelperClasses.h> template enum parser. Copied from: http://stackoverflow.com/a/726681

class **InfoSwitch**

#include <HelperClasses.h> A struct that is used for parsing configuration strings and assigning booleans to various properties. Currently used in plotting code.

Strings are used to turn on and off histograms and branches in the tree The following structs hold the bools used to control the content and also have the string which is necessary to turn a set on. See the derived members for more information about what is supported. Each derived member should provide a table of parameters, patterns, and type of matching scheme used. The pattern will use standard PCRE-syntax when appropriate.

We support two major matching schemes:

Exact If a variable is matched exactly to a string, then a boolean is set to True or False based on whether an exact match exists or not.

Partial If a variable is partially matched to a string, then there is some specific pattern we are extracting that will succeed the partial match that determines what the variable will be set to (usually not a bool).

Subclassed by *HelperClasses::EventInfoSwitch*, *HelperClasses::IParticleInfoSwitch*, *HelperClasses::METInfoSwitch*, *HelperClasses::TrackInfoSwitch*, *HelperClasses::TriggerInfoSwitch*

Public Functions

inline **InfoSwitch**(const std::string configStr)

Constructor. Take in input string, create vector of tokens.

Parameters **configStr** – The configuration string to split up.

inline bool **has_exact**(const std::string flag)

Search for an exact match in *m_configDetails*.

Parameters **flag** – The string we search for.

inline bool **has_match**(const std::string flag)

Search for a partial match in *m_configStr*.

Parameters **flag** – The string we search for.

std::string **get_working_point**(const std::string flag)

Search for a single flag in *m_configDetails* and parse out the working point.

Parameters **flag** – The string we search for.

std::vector<std::string> **get_working_points**(const std::string flag)

Search for multiple flags in *m_configDetails* and parse out the working points.

Parameters **flag** – The string we search for.

Protected Attributes

const std::string **m_configStr**

The input configuration string from which we split up into tokens.

std::set<std::string> **m_configDetails**

The vector of tokens from which we search through for finding matches.

class **EventInfoSwitch** : public *HelperClasses*::*InfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::InfoSwitch` struct for Event Information.

Parameter	Pattern	Match
m_noDataInfo	noDataInfo	exact
m_eventCleaning	eventCleaning	exact
m_bcidInfo	bcidInfo	exact
m_pileup	pileup	exact
m_pileupsys	pileupsys	exact
m_shapeEM	shapeEM	exact
m_shapeEMPFLOW	shapeEMPFLOW	exact
m_shapeLC	shapeLC	exact
m_truth	truth	exact
m_caloClus	caloClusters	exact
m_weightsSys	weightsSys	exact
m_beamspotweight	beamspotweight	exact

class **TriggerInfoSwitch** : public *HelperClasses*::*InfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::InfoSwitch` struct for Trigger Information.

Parameter	Pattern	Match
m_basic	basic	exact
m_menuKeys	menuKeys	exact
m_passTriggers	passTriggers	exact
m_passTrigBits	passTrigBits	exact
m_prescales	prescales	exact
m_prescalesLumi	prescalesLumi	exact

Note: `m_prescales` contains information from the `TrigDecisionTool` for every trigger used in event selection and event trigger-matching. `m_prescalesLumi` contains information retrieved from the pile-up reweighting tool based on the actual luminosities of triggers.

class **IParticleInfoSwitch** : public *HelperClasses*::*InfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::InfoSwitch` struct for IParticle Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_numLeading	NLeading	partial
m_useTheS	useTheS	exact

Note: `m_numLeading` requires a number `XX` to follow it, defining the number of leading partiles and associate it with that variable.

For example:

```
m_configStr = "... NLeading4 ..."
```

will define `int m_numLeading = 4`.

Subclassed by *HelperClasses::ClusterInfoSwitch*, *HelperClasses::ElectronInfoSwitch*, *HelperClasses::JetInfoSwitch*, *HelperClasses::MuonInfoSwitch*, *HelperClasses::PhotonInfoSwitch*, *HelperClasses::TauInfoSwitch*, *HelperClasses::TruthInfoSwitch*

class **MuonInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::IParticleInfoSwitch` class for Muon Information.

Parameter	Pattern	Match
m_trigger	trigger	exact
m_isolation	isolation	exact
m_isolationKinematics	isolationKinematics	exact
m_quality	quality	exact
m_recoparams	recoparams	exact
m_trackparams	trackparams	exact
m_trackhitcont	trackhitcont	exact
m_effSF	effSF	exact
m_energyLoss	energyLoss	exact
m_recoWPs[XYZ]	RECO_XYZ	pattern
m_isolWPs[“”]	**ISOL_**	exact
m_isolWPs[“”]	ISOL_NONE	exact
m_isolWPs[XYZ]	ISOL_XYZ	pattern
m_trigWPs[XYZ]	TRIG_XYZ	pattern
m_passSel	passSel	exact
m_passOR	passOR	exact

Note: `quality`, `isolation` and `effSF` switches do not enable any additional output by themselves. They require additional working point pattern using `RECO_XYZ` for quality working points and scale factors, `ISOL_XYZ` for isolation working points and scale factors, and `TRIG_XYZ` for trigger scale factors. `XYZ` in the pattern should be replaced using the working point name, for example:

```
m_configStr = "... RECO_Medium ..."
```

will define the `Medium` quality working point and the accompanying scale factors.

Isolation supports NONE or empty option which will enable scale factors without additional isolation requirements, for example:

```
m_configStr = "... ISOL_NONE ISOL_Loose ..."
```

will define the Loose isolation working point status branch, and scale factors without isolation requirements and using the Loose WP.

class **ElectronInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

#include <HelperClasses.h>

The *HelperClasses::IParticleInfoSwitch* class for Electron Information.

Parameter	Pattern	Match
m_trigger	trigger	exact
m_isolation	isolation	exact
m_isolationKinematics	isolationKinematics	exact
m_PID	PID	exact
m_trackparams	trackparams	exact
m_trackhitcont	trackhitcont	exact
m_effSF	effSF	exact
m_PIDWPs[XYZ]	PID_XYZ	pattern
m_PIDSFWPs[XYZ]	PIDSF_XYZ	pattern
m_isolWPs[""]	**ISOL_**	exact
m_isolWPs[""]	ISOL_NONE	exact
m_isolWPs[XYZ]	ISOL_XYZ	pattern
m_trigWPs[XYZ]	TRIG_XYZ	pattern
m_passSel	passSel	exact
m_passOR	passOR	exact

Note: PID, isolation and effSF switches do not enable any additional output by themselves. They require additional working point pattern using PID_XYZ for PID working points, PIDSF_XYZ for PID scale factors, ISOL_XYZ for isolation working points and scale factors, and TRIG_XYZ for trigger scale factors. XYZ in the pattern should be replaced using the working point name, for example:

```
m_configStr = "... PID_LHMedium PIDSF_MediumLLH ..."
```

will define the LHMedium PID working point and the accompanying scale factors. Note that not all PID working points have scale factors available.

Isolation supports NONE or empty option which will enable scale factors without additional isolation requirements, for example:

```
m_configStr = "... ISOL_NONE ISOL_Loose ..."
```

will define the Loose isolation working point status branch, and scale factors without isolation requirements and using the Loose WP.

class **PhotonInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

#include <HelperClasses.h>

The *HelperClasses::IParticleInfoSwitch* class for Photon Information.

Parameter	Pattern	Match
m_isolation	isolation	exact
m_PID	PID	exact
m_purity	purity	exact
m_effSF	effSF	exact
m_trigger	trigger	exact
m_isoCones	isoCone	partial

Note: `isoCone` can be repeated but requires a number after it, for example:

```
m_configStr = "... isoCone20 isoCone40 ..."
```

which will define `std::vector<int> m_isoCones = {20,40}`.

class **ClusterInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

class **JetInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

#include <HelperClasses.h>

The *HelperClasses::IParticleInfoSwitch* class for Jet Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_trigger	trigger	exact
m_substructure	substructure	exact
m_ntrimsubjets	ntrimsubjets	exact
m_bosonCount	bosonCount	exact
m_VTags	VTags	exact
m_rapidity	rapidity	exact
m_clean	clean	exact
m_cleanLight	cleanLight	exact
m_cleanLightLLP	cleanLightLLP	exact
m_cleanTrig	cleanTrig	exact
m_timing	timing	exact
m_energy	energy	exact
m_energyLight	energyLight	exact
m_scales	scales	exact
m_constscaleEta	constscaleEta	exact
m_detectorEta	detectorEta	exact
m_resolution	resolution	exact
m_truth	truth	exact
m_truthDetails	truth_details	exact
m_layer	layer	exact
m_trackPV	trackPV	exact
m_trackAll	trackAll	exact
m_chargedPFOPV	chargedPFOPV	exact

continues on next page

Table 1 – continued from previous page

Parameter	Pattern	Match
m_jvt	JVT	exact
m_NNJvt	NNJvt	exact
m_sfJVTName	sfJVT	partial
m_sffJVTName	sffJVT	partial
m_allTrack	allTrack	exact
m_allTrackPVSel	allTrackPVSel	exact
m_allTrackDetail	allTrackDetail	exact
m_constituent	constituent	exact
m_constituentAll	constituentAll	exact
m_flavorTag	flavorTag	exact
m_flavorTagHLT	flavorTagHLT	exact
m_sfFTagFix	sfFTagFix	partial
m_sfFTagFlt	sfFTagFlt	partial
m_sfFTagHyb	sfFTagHyb	partial
m_jetBTag	jetBTag	partial
m_area	area	exact
m_JVC	JVC	exact
m_tracksInJet	tracksInJet	partial
m_trackJetName	trackJetName	partial
m_hltVtxComp	hltVtxComp	exact
m_onlineBS	onlineBS	exact
m_onlineBSTool	onlineBSTool	exact
m_charge	charge	exact
m_passSel	passSel	exact
m_passOR	passOR	exact
m_vsLumiBlock	vsLumiBlock	exact
m_vsActualMu	vsActualMu	exact
m_lumiB_runN	lumiB_runN	exact
m_byAverageMu	byAverageMu	exact
m_byEta	byEta	exact
m_etaPhiMap	etaPhiMap	exact
m_muonCorrection	muonCorrection	exact

`trackJetName` expects one or more track jet container names separated by an underscore. For example, the string `trackJetName_GhostAntiKt2TrackJet_GhostVR30Rmax4Rmin02TrackJet` will set the attriubte `m_trackJetNames` to `{"GhostAntiKt2TrackJet", "GhostVR30Rmax4Rmin02TrackJet"}`.

Note: `sfJVT` requires a working point after it, for example:

```
m_configStr = "... sfJVTMedium ..."
```

`jetBTag` expects the format `jetBTag_tagger_type_AABB..MM..YY.ZZ`. This will create a vector of working points (AA, BB, CC, …, ZZ) associated with that tagger. Several entries can be given. For example:

m_configStr = "… jetBTag_DL1r_FixedCutBEff_60707785 …"

class **TruthInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::InfoSwitch` struct for Truth Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_type	type	exact
m_bVtx	bVtx	exact
m_parents	parents	exact
m_children	children	exact
m_dressed	dressed	exact
m_origin	origin	exact
m_particleType	particleType	exact
m_pdgIdOnly	pdgIdOnly	exact

class **TrackInfoSwitch** : public *HelperClasses*::*InfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::InfoSwitch` struct for Track Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_fitpars	fitpars	exact
m_numbers	numbers	exact
m_vertex	vertex	exact
m_useTheS	useTheS	exact

class **TauInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::IParticleInfoSwitch` struct for Tau Information.

Note: `identification` and `effSF` switches do not enable any additional output by themselves. They require additional working point pattern using `TAUEFF_XYZ` for combined scale factors, and `TRIG_XYZ` for trigger scale factors. `XYZ` in the pattern should be replaced using the working point name, for example:

```
m_configStr = "... TAUEFF_EleOLRElectronEleRNNLoose_TauIDMedium ... TRIG_
↪EleOLRElectronEleRNNMedium_TauIDLoose_TrigMyTriggerMenu"
```

Notice that the working point for TAUEFF is a combination of two working points from EleOLRElectron and TauID.

class **METInfoSwitch** : public *HelperClasses*::*InfoSwitch*

#include <HelperClasses.h>

The `HelperClasses::InfoSwitch` struct for Missing E_T Information.

Parameter	Pattern	Match
m_metClus	metClus	exact
m_metTrk	metTrk	exact
m_sigClus	sigClus|all	exact
m_sigTrk	sigTrk|all	exact
m_sigResolutionClus	sigResolutionClus|all	exact
m_sigResolutionTrk	sigResolutionTrk|all	exact
m_refEle	refEle|all	exact
m_refGamma	refGamma|all	exact
m_refTau	refTau|all	exact
m_refMuons	refMuons|all	exact
m_refJet	refJet|all	exact
m_refJetTrk	refJetTrk	exact
m_softClus	softClus|all	exact
m_softTrk	softTrk|all	exact
m_noExtra	noExtra	exact

Note: For all except *m_refJetTrk*, you can pass in the string `"all"` to enable all information. You can force only calocluster- or track-based MET using *m_metClus* or *m_metTrk*.

Helper Functions

namespace **HelperFunctions**

Enums

enum **ShowerType**

The different supported shower types.

Values:

enumerator **Unknown**

enumerator **Pythia8**

enumerator **Herwig7**

enumerator **Sherpa21**

enumerator **Sherpa22**

enumerator **Sherpa2210**

Functions

MsgStream &**msg**(MSG::Level lvl = MSG::INFO)

Static object that provides athena-based message logging functionality

bool **passPrimaryVertexSelection**(const xAOD::VertexContainer *vertexContainer, int Ntracks = 2)

int **countPrimaryVertices**(const xAOD::VertexContainer *vertexContainer, int Ntracks = 2)

const xAOD::Vertex ***getPrimaryVertex**(const xAOD::VertexContainer *vertexContainer, MsgStream &msg)

inline const xAOD::Vertex ***getPrimaryVertex**(const xAOD::VertexContainer *vertexContainer)

float **getPrimaryVertexZ**(const xAOD::Vertex *pvx)

int **getPrimaryVertexLocation**(const xAOD::VertexContainer *vertexContainer, MsgStream &msg)

inline int **getPrimaryVertexLocation**(const xAOD::VertexContainer *vertexContainer)

bool **applyPrimaryVertexSelection**(const xAOD::JetContainer *jets, const xAOD::VertexContainer *vertices)

std::string **replaceString**(std::string subjet, const std::string &search, const std::string &replace)

std::vector<TString> **SplitString**(TString &orig, const char separator)

float **dPhi**(float phi1, float phi2)

bool **has_exact**(const std::string input, const std::string flag)

std::size_t **string_pos**(const std::string &haystack, const std::string &needle, unsigned int N)

Function which returns the position of the n-th occurence of a character in a string searching backwards. Returns -1 if no occurencies are found.

Source: http://stackoverflow.com/questions/18972258/index-of-nth-occurrence-of-the-string

StatusCode **isAvailableMetaData**(TTree *metaData)

bool **isFilePrimaryxAOD**(TFile *inputFile)

std::vector<TLorentzVector> **jetReclustering**(const xAOD::JetContainer *jets, double radius = 1.0, double fcut = 0.05, fastjet::JetAlgorithm rc_alg = fastjet::antikt_algorithm)

std::vector<TLorentzVector> **jetTrimming**(const xAOD::JetContainer *jets, double radius = 0.3, double fcut = 0.05, fastjet::JetAlgorithm s_alg = fastjet::kt_algorithm)

TLorentzVector **jetTrimming**(const xAOD::Jet *jet, double radius = 0.3, double fcut = 0.05, fastjet::JetAlgorithm s_alg = fastjet::kt_algorithm)

bool **sort_pt**(const xAOD::IParticle *partA, const xAOD::IParticle *partB)

std::vector<CP::SystematicSet> **getListofSystematics**(const CP::SystematicSet inSysts, std::string systNames, float systVal, MsgStream &msg)

Get a list of systematics.

Parameters

- **inSysts** – systematics set retrieved from the tool

- **systNames** – comma separated list of wanted systematics names, use "Nominal" for nominal and "All" for all systematics
- **systVal** – continuous systematics sigma value
- **msg** – the MsgStream object with appropriate level for debugging

void **writeSystematicsListHist**(const std::vector<CP::SystematicSet> &systs, std::string histName, TFile *file)

template<typename **T**>
std::string **type_name**(bool useXAOD = true)

template<typename **T1**, typename **T2**>
StatusCode **makeSubsetCont**(*T1* *&intCont, *T2* *&outCont, MsgStream &msg, const std::string &flagSelect = "", *HelperClasses*::*ToolName* tool_name = *HelperClasses*::*ToolName*::*DEFAULT*)

Function to copy a subset of a generic input xAOD container into a generic output xAOD container.

If the optional parameters aren't specified, the function will just make a full copy of the input container into the output one.

Author Marco Milesi (marco.milesi@cern.ch)

Parameters

- **intCont** – **[in]** input container
- **outCont** – **[inout]** output container
- **flagSelect** – **[in]** (optional) the name of the decoration for objects passing a certain selection (e.g. "passSel", "overlaps" …). When explicitly specified, it must not be empty.
- **tool_name** – **[in]** (optional) an enum specifying the tool type which is calling this function (definition in `HelperClasses::ToolName`)

template<typename **T1**, typename **T2**>
StatusCode **makeSubsetCont**(*T1* *&intCont, *T2* *&outCont, const std::string &flagSelect = "", *HelperClasses*::*ToolName* tool_name = *HelperClasses*::*ToolName*::*DEFAULT*)

template<typename **T**>
StatusCode **retrieve**(*T* *&cont, std::string name, xAOD::TEvent *event, xAOD::TStore *store, MsgStream &msg)

Retrieve an arbitrary object from TStore / TEvent.

This tries to make your life simple by providing a one-stop container retrieval shop for all types.

Example Usage:

```
const xAOD::JetContainer  jets(0);
// look for "AntiKt10LCTopoJets" in both TEvent and TStore
ANA_CHECK( HelperFunctions::retrieve(jets, "AntiKt10LCTopoJets", m_event, m_
↪store) );
// look for "AntiKt10LCTopoJets" in only TStore
```

(continues on next page)

(continued from previous page)

```
ANA_CHECK( HelperFunctions::retrieve(jets, "AntiKt10LCTopoJets", 0, m_store) );
// look for "AntiKt10LCTopoJets" in only TEvent, enable verbose output
ANA_CHECK( HelperFunctions::retrieve(jets, "AntiKt10LCTopoJets", m_event, 0,␣
↪msg()) );
```

Checking Order:

- start by checking TStore
  - check if store contains 'xAOD::JetContainer' named 'name'
    - attempt to retrieve from store
    - return if failure
- next check TEvent
  - check if event contains 'xAOD::JetContainer' named 'name'
    - attempt to retrieve from event
    - return if failure
  - return FAILURE
- return SUCCESS (should never reach this last line)

Parameters

- **cont** – pass in a pointer to the object to store the retrieved container in
- **name** – the name of the object to look up
- **event** – the TEvent, usually wk()->xaodEvent(). Set to 0 to not search TEvent.
- **store** – the TStore, usually wk()->xaodStore(). Set to 0 to not search TStore.
- **msg** – the MsgStream object with appropriate level for debugging

template<typename **T**>
StatusCode **retrieve**(*T* *&cont, std::string name, xAOD::TEvent *event, xAOD::TStore *store)

template<typename T> StatusCode __attribute__ ((deprecated("retrieve<T>(..., bool) is deprecated. See https://github.com/UCATLAS/xAODAnaHelpers/pull/882"))) retrieve(T *&cont

template<typename **T**>
bool **isAvailable**(std::string name, xAOD::TEvent *event, xAOD::TStore *store, MsgStream &msg)

Return true if an arbitrary object from TStore / TEvent is available.

This tries to make your life simple by providing a one-stop container check shop for all types

Example Usage:

```
const xAOD::JetContainer  jets(0);
// look for "AntiKt10LCTopoJets" in both TEvent and TStore
HelperFunctions::isAvailable<xAOD::JetContainer>("AntiKt10LCTopoJets", m_event,␣
↪m_store)
```

(continues on next page)

(continued from previous page)

```
// look for "AntiKt10LCTopoJets" in only TStore
HelperFunctions::isAvailable<xAOD::JetContainer>("AntiKt10LCTopoJets", 0, m_
↪store)
// look for "AntiKt10LCTopoJets" in only TEvent, enable verbose output
HelperFunctions::isAvailable<xAOD::JetContainer>("AntiKt10LCTopoJets", m_event,␣
↪0, MSG::VERBOSE)
```

Parameters

- **name** – the name of the object to look up
- **event** – the TEvent, usually wk()->xaodEvent(). Set to 0 to not search TEvent.
- **store** – the TStore, usually wk()->xaodStore(). Set to 0 to not search TStore.
- **msg** – the MsgStream object with appropriate level for debugging

template<typename **T**>
bool **isAvailable**(std::string name, xAOD::TEvent *event, xAOD::TStore *store)

template<class **T**>
const *T* ***getLink**(const xAOD::IParticle *particle, std::string name)

Access to element link to object of type T stored in auxdata.

template<typename **T**>
T **sort_container_pt**(*T* *inCont)

template<typename **T**>
const *T* **sort_container_pt**(const *T* *inCont)

inline bool **found_non_dummy_sys**(std::vector<std::string> *sys_list)

template<typename **T1**, typename **T2**, typename **T3**>
StatusCode **makeDeepCopy**(xAOD::TStore *m_store, std::string containerName, const *T1* *cont)

Make a deep copy of a container and put it in the TStore.

This is a very powerful templating function. The point is to remove the triviality of making deep copies by specifying all that is needed. The best way is to demonstrate via example:

```
const xAOD::JetContainer  selected_jets(nullptr);
ANA_CHECK( m_event->retrieve( selected_jets, "SelectedJets" ));
ANA_CHECK( (HelperFunctions::makeDeepCopy<xAOD::JetContainer,␣
↪xAOD::JetAuxContainer, xAOD::Jet>(m_store, "BaselineJets", selected_jets)));
```

Template Parameters

- **T1** – The type of the container you're going to deep copy into
- **T2** – The type of the aux container you're going to deep copy into
- **T3** – The type of the object inside the container you're going to deep copy

Parameters

- **m_store** – A pointer to the TStore object

- **containerName** – The name of the container to create as output in the TStore
- **cont** – The container to deep copy, it should be a container of pointers (IParticleContainer or ConstDataVector)

template<typename **T1**, typename **T2**>
StatusCode **recordOutput**(xAOD::TEvent *m_event, xAOD::TStore *m_store, std::string containerName)

Copy a container from the TStore to be recorded in the TEvent (eg: to an output)

If you have a container in the TStore, this function will record it into the output for you without an issue. As an example:

```
ANA_CHECK( HelperFunctions::recordOutput<xAOD::JetContainer,xAOD::JetAuxContainer>(m_event, m_store, "BaselineJets"));
```

where we build off the previous example of making a deep copy (see *HelperFunctions::makeDeepCopy()*).

Template Parameters

- **T1** – The type of the container you're going to record
- **T2** – The type of the aux container you're going to record

Parameters

- **m_event** – A pointer to the TEvent object
- **m_store** – A pointer to the TStore object
- **containerName** – The name of the container in the TStore to record to TEvent

template<typename **T_BR**>
void **connectBranch**(std::string name, TTree *tree, const std::string &branch, std::vector<*T_BR*> **variable)

template<typename **T**>
void **remove_duplicates**(std::vector<*T*> &vec)

ShowerType **getMCShowerType**(const std::string &sample_name)

Determines the type of generator used for the shower from the sample name.

The name of the generator is determined using some common definitions in the ATLAS MC dataset naming scheme. The case independent strings that are searched for are:

PYTHIA8EVTGEN or Py8EG or PYTHIA : Pythia8 HERWIG : Herwig7 SHERPA_CT : Sherpa21 SHERPA : Sherpa22 (if not Sherpa 21)

Parameters **sample_name** – The name of the sample, usualy the dataset name

Variables

StatusCode std::string name

StatusCode std::string xAOD::TEvent * event

StatusCode std::string xAOD::TStore * store

StatusCode std::string xAOD::TEvent xAOD::TStore bool debug = { return retrieve<T>(cont, name, event, store, msg())

struct **pt_sort**

Public Functions

inline bool **operator()**(const TLorentzVector &lhs, const TLorentzVector &rhs)

inline bool **operator()**(const TLorentzVector *lhs, const TLorentzVector *rhs)

inline bool **operator()**(const xAOD::IParticle &lhs, const xAOD::IParticle &rhs)

inline bool **operator()**(const xAOD::IParticle *lhs, const xAOD::IParticle *rhs)

MET Constructor

class **METConstructor** : public xAH::*Algorithm*

Public Functions

METConstructor()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_mapName** = "METAssoc_AntiKt4LCTopo"

std::string **m_coreName** = "MET_Core_AntiKt4LCTopo"

std::string **m_outputContainer** = "NewRefFinal"

std::string **m_systConfigPrefix** = "METUtilities/R22_PreRecs"

std::string **m_systConfigSoftTrkFile** = "TrackSoftTerms-pflow.config"

std::string **m_inputJets** = ""

std::string **m_inputElectrons** = ""

std::string **m_inputPhotons** = ""

std::string **m_inputTaus** = ""

std::string **m_inputMuons** = ""

bool **m_doElectronCuts** = false

bool **m_doPhotonCuts** = false

bool **m_doTauCuts** = false

bool **m_doMuonCuts** = false

bool **m_doMuonEloss** = false

bool **m_doIsolMuonEloss** = false

bool **m_doJVTCut** = false

bool **m_dofJVTCut** = false

std::string **m_fJVTdecorName** = "passFJVT"

Name of fJVT decoration.

bool **m_doPFlow** = true

To turn on p-flow MET calculation set m_doPFlow to true.

std::string **m_METWorkingPoint** = ""
Name of MET Working Point (defines the JetSelection applied in METMaker)

bool **m_rebuildUsingTracksInJets** = false
Rebuild MET using tracks in calo jets.

bool **m_addSoftClusterTerms** = false
Include soft cluster terms if rebuilding MET using jet terms (only considered if *m_rebuildUsingTracksInJets* is false)

bool **m_calculateSignificance** = false
Enable MET significance calculation.

bool **m_significanceTreatPUJets** = true
Introduce "resolution" for jets with low JVT, if the analysis is sensitive to pileup jets.

double **m_significanceSoftTermReso** = 10.0
Set soft term resolution.

bool **m_runNominal** = true
set to false if you want to run met systematics

std::string **m_systName** = "All"
do not change it, not useful

float **m_systVal** = 1.0

bool **m_writeSystToMetadata** = false
Write systematics names to metadata.

std::string **m_jetSystematics** = ""
Name of jet systematics vector from *JetCalibrator*.

std::string **m_eleSystematics** = ""
Name of electron systematics vector from *ElectronCalibrator*.

std::string **m_muonSystematics** = ""
Name of muon systematics vector from *MuonCalibrator*.

std::string **m_tauSystematics** = ""
Name of tau systematics vector from *TauCalibrator*.

std::string **m_phoSystematics** = ""
Name of photon systematics vector from *PhotonCalibrator*.

std::string **m_outputAlgoSystNames** = ""

Private Members

asg::AnaToolHandle<IMETMaker> **m_metmaker_handle** = {"met::METMaker/METMaker", this}

asg::AnaToolHandle<IMETSystematicsTool> **m_metSyst_handle** = {"met::METSystematicsTool/METSystematicsTool", this}

asg::AnaToolHandle<IMETSignificance> **m_metSignificance_handle** = {"met::METSignificance/METSignificance", this}

asg::AnaToolHandle<TauAnalysisTools::ITauSelectionTool> **m_tauSelTool_handle** = {"TauAnalysisTools::TauSelectionTool/TauSelectionTool", this}

std::vector<CP::SystematicSet> **m_sysList**

int **m_numEvent**

Particle PID Manager

Electron LH PID Manager

> **Warning:** doxygenclass: Cannot find class "ElectronLHPIDManager" in doxygen xml output for project "xAH" from directory: ./doxygen/xml

Electron Cut-Based PID Manager

class **ElectronCutBasedPIDManager**

Public Functions

ElectronCutBasedPIDManager()

ElectronCutBasedPIDManager(std::string WP, bool debug = false)

~ElectronCutBasedPIDManager()

StatusCode **setupWPs**(bool configTools, std::string selector_name = "")

StatusCode **setDecorations**(const xAOD::Electron *electron)

inline const std::string **getSelectedWP**()

inline std::multimap<std::string, AsgElectronIsEMSelector*> **getAllWPTools**()

inline std::multimap<std::string, AsgElectronIsEMSelector*> **getValidWPTools**()

inline const std::set<std::string> **getAllWPs**()

inline const std::set<std::string> **getValidWPs**()

Private Members

std::string **m_selectedWP**

bool **m_debug**

std::multimap<std::string, AsgElectronIsEMSelector*> **m_allWPTools**

std::multimap<std::string, AsgElectronIsEMSelector*> **m_validWPTools**

std::set<std::string> **m_allWPAuxDecors**

std::set<std::string> **m_validWPs**

AsgElectronIsEMSelector ***m_asgElectronIsEMSelector_Loose**

AsgElectronIsEMSelector ***m_asgElectronIsEMSelector_Medium**

AsgElectronIsEMSelector ***m_asgElectronIsEMSelector_Tight**

xAH::Algorithm

class **Algorithm** : public EL::Algorithm

This is used by all algorithms within xAODAnaHelpers.

The main goal of this algorithm class is to standardize how everyone defines an algorithm that plugs into xAODAnaHelpers. A series of common utilities are provided such as *m_className* which defines the class name so we can manage a registry *m_instanceRegistry* to keep xAODAnaHelpers as flexible as possible to our users.

We expect the user to create a new algorithm, such as a selector for jets:

```
class JetSelector : public xAH::Algorithm
{
    // ...
};
```

The above example is taken from our implementation in *JetSelector*. Just remember that when you write your initializer, you will be expected to do something like:

```
// this is needed to distribute the algorithm to the workers
ClassImp(JetSelector)


JetSelector :: JetSelector () :
    Algorithm("JetSelector"),
    ...
{
```

(continues on next page)

(continued from previous page)

```
    // ...
}
```

which this class will automatically register all instances of for you. Each instance can have a different algorithm name but will have the same *m_className* so we can track how many references have been made. This is useful for selectors to deal with cutflows, but can be useful for other algorithms that need to know how many times they've been instantiated in a single job.

Note: The expectation is that the user does not directly use this class but rather inherits from it.

Subclassed by *BJetEfficiencyCorrector*, *BasicEventSelection*, *ClusterHistsAlgo*, *DebugTool*, *ElectronCalibrator*, *ElectronEfficiencyCorrector*, *ElectronSelector*, *HLTJetGetter*, *IParticleHistsAlgo*, *IsoCloseByCorr*, *JetCalibrator*, *JetSelector*, *METConstructor*, *MessagePrinterAlgo*, *MetHistsAlgo*, *MinixAOD*, *MuonCalibrator*, *MuonEfficiencyCorrector*, *MuonInFatJetCorrector*, *MuonSelector*, *OverlapRemover*, *PhotonCalibrator*, *PhotonSelector*, *TauCalibrator*, *TauEfficiencyCorrector*, *TauJetMatching*, *TauSelector*, *TrackHistsAlgo*, *TrackSelector*, *TreeAlgo*, *TrigMatcher*, *TruthSelector*, *Writer*

Public Functions

Algorithm(std::string className = "Algorithm")

Initialization.

Parameters className – This is the name of the class that inherits from :cpp:namespace:~`xAH::Algorithm`

~Algorithm()

StatusCode **algInitialize**()

Run any initializations commmon to all xAH Algorithms (such as registerInstance). Call this inside `histInitialize` for best results.

StatusCode **algFinalize**()

Run any finalizations common to all xAH Algorithms (such as unregisterInstance). Call this inside `histFinalize` for best results.

StatusCode **parseSystValVector**()

Parse string of systematic sigma levels in m_systValVectorString into m_systValVector.

Public Members

std::string **m_name** = "UnnamedAlgorithm"

All algorithms initialized should have a unique name, to differentiate them at the TObject level.

Note, `GetName()` returns a `char*` while this returns a `std::string`.

bool **m_debug** = false

m_debug is being deprecated

bool **m_verbose** = false

m_verbose is being deprecated

MSG::Level **m_msgLevel** = MSG::INFO

debug level

std::string **m_cutFlowStreamName** = "cutflow"

std::string **m_systName** = ""

If running systematics, the name of the systematic

float **m_systVal** = 0.0

If running systematics, the value to set the systematic to

Note: This will set the systematic to the value $\pm x$.

std::string **m_systValVectorString** = ""

If running systematics, you can run multiple points and store them in here. A comma separated list of working points should be given to m_systValVectorString, and then parsed by calling parseSystValVector.

std::vector<float> **m_systValVector**

std::string **m_eventInfoContainerName** = "EventInfo"

If the xAOD has a different *EventInfo* container name, set it here

std::string **m_vertexContainerName** = "PrimaryVertices"

If the xAOD has a different PrimaryVertex container name, set it here

int **m_isMC** = -1

This stores the isMC decision, and can also be used to override at the algorithm level to force analyzing MC or not.

Value	Meaning
-1	Default, use eventInfo object to determine if data or mc
0	Treat the input as data
1	Treat the input as MC

int **m_isFastSim** = -1

This stores the isFastSim decision, and can also be used to override at the algorithm level to force analyzing FastSim or not.

Value	Meaning
-1	Default, use Metadata object to determine if FullSim or FastSim
0	Treat the input as FullSim
1	Treat the input as FastSim

int **m_isAF3** = -1

This stores the isAF3 decision, and can also be used to override at the algorithm level to force analyzing FastSim with AF3 or not.

Value	Meaning
-1	Default, use Metadata object to determine if AF3 FastSim or not
0	Treat the input as FullSim or AFII
1	Treat the input as FastSim with AF3

bool **m_useRun3navigation** = false

Flag to use Run 3 trigger navigation (true), or Run 2 navigation (false)

std::string **m_HLTSummary** = "HLTNav_Summary_DAODSlimmed"

String storing the type of HLT navigation info available for Run 3 samples. For AODs or unslimmed DAODs: HLTNav_Summary_AODSlimmed

bool **m_forceFastSim** = false

Flags to force a specific data-type, even if it disagrees with your input

bool **m_forceFullSim** = false

bool **m_forceData** = false

bool **m_setAFII** = false

Backwards compatibility, same as m_forceFastSim

bool **m_setAF3** = false

Protected Functions

bool **isMC**()

```
\verbatim embed:rst:leading-asterisk
   Try to determine if we are running over data or MC. The␣
↪:cpp:member:`xAH::Algorithm::m_isMC` can be used
```

to fix the return value. Otherwise the *EventInfo* object is queried.

An exception is thrown if the type cannot be determined.

```
    ============ =======
    Return Value Meaning
    ============ =======
    0            Data
    1            MC
    ============ =======
```

bool **isFastSim**()

```
\verbatim embed:rst:leading-asterisk
    Try to determine if we are running over data or MC. The␣
↪:cpp:member:`xAH::Algorithm::m_isFastSim` can be used
```

to fix the return value. Otherwise the metadata is queried.

An exception is thrown if the type cannot be determined.

```
    ============ =======
    Return Value Meaning
    ============ =======
    0            FullSim (or Data)
    1            FastSim
    ============ =======
```

bool **isAF3**()

If the name includes ATLFASTII or ATLFAST3 then set to AFII or AF3, if deemed fullSim then FS else leave as empty string and complain

bool **isPHYS**()

Determines if using DAOD_PHYS or not.

void **registerInstance**()

Register the given instance under the moniker *xAH::Algorithm::m_className*

This will increase the reference count by 1.

int **numInstances**()

Return number of instances registered under the moniker *xAH::Algorithm::m_className*

This will return the reference count.

Warning: If for some reason the instance wasn't registered, we spit out a warning.

void **unregisterInstance**()

Unregister the given instance under the moniker *xAH::Algorithm::m_className*

This will decrease the reference count by 1.

Warning: If for some reason the instance wasn't registered, we spit out a warning.

template<typename **T**>
inline StatusCode **checkToolStore**(const std::string &tool_name)

```
\verbatim embed:rst:leading-asterisk
    Check whether the input CP tool already exists with *this* name in the␣
↪asg::ToolStore
```

Depending on the outcome, the content of the map :cpp:member:`xAH::Algorithm::m_toolAlreadyUsed` wll be set accordingly.

inline bool **isToolAlreadyUsed**(const std::string &tool_name)

Check whether the input CP tool has been already used by any *xAH::Algorithm* in the current job by scanning *xAH::Algorithm::m_toolAlreadyUsed*.

template<typename **T**>
inline void **setToolName**(__attribute__((unused)) asg::AnaToolHandle<*T*> &handle, __attribute__((unused)) const std::string &name = "") const

Sets the name of a tool. If no name is needed, the tool will use the name of the algorithm plus a unique identifier (*xAH::Algorithm::getAddress()*) appended to ensure the tool is unique and effectively private.

The tool will not be guaranteed unique if two tools of the same type are created without a name passed in. But this is, at this point, up to the user and a more complex scenario than what this function tries to simplify on its own.

inline std::string **getAddress**() const

Return a `std::string` representation of `this`

Protected Attributes

std::string **m_className** = "Algorithm"

The moniker by which all instances are tracked in *xAH::Algorithm::m_instanceRegistry*

xAOD::TEvent ***m_event** = nullptr

The TEvent object

xAOD::TStore ***m_store** = nullptr

The TStore object

Private Members

bool **m_registered** = false

A boolean to keep track of whether this instance was registered or not.

Calling *xAH::Algorithm::registerInstance()* multiple times won't inflate the number of instances of a class made because of me.

std::map<std::string, bool> **m_toolAlreadyUsed**

Map containing info about whether a CP Tool of a given name has been already used or not by this *xAH::Algorithm*.

Its content gets set through *xAH::Algorithm::checkToolStore()*, depending on whether the tool it's created from scratch, or retrieved from `asg::ToolStore`

Private Static Attributes

static std::map<std::string, int> **m_instanceRegistry** = {}

Bookkeeps the number of times *xAH::Algorithm::m_className* has been used in a variable shared among all classes/instances that inherit from me

MessagePrinterAlgo

class **MessagePrinterAlgo** : public xAH::*Algorithm*

This algorithm changes the format of the `MsgStream` objects for all other algorithms. There should only be on instance of it, and it should probably be first.

Public Members

unsigned int **m_sourceWidth** = 25

Set the width of the name in the message.

4.9 Doxygen API

4.9.1 Page Hierarchy

4.9.2 Class Hierarchy

4.9.3 File Hierarchy

4.9.4 Full API

Namespaces

Namespace CP

Namespace EL

Namespace HelperClasses

Contents

Classes

- *Class ClusterInfoSwitch*
- *Class ElectronInfoSwitch*
- *Template Class EnumParser*
- *Class EventInfoSwitch*
- *Class InfoSwitch*
- *Class IParticleInfoSwitch*
- *Class JetInfoSwitch*
- *Class METInfoSwitch*
- *Class MuonInfoSwitch*
- *Class PhotonInfoSwitch*
- *Class TauInfoSwitch*
- *Class TrackInfoSwitch*
- *Class TriggerInfoSwitch*
- *Class TruthInfoSwitch*

Enums

- *Enum ContainerType*
- *Enum ToolName*

Namespace HelperFunctions

Contents

Classes

- *Struct pt_sort*

Enums

- *Enum ShowerType*

Functions

- *Template Function HelperFunctions::__attribute__*
- *Function HelperFunctions::applyPrimaryVertexSelection*
- *Template Function HelperFunctions::connectBranch*
- *Function HelperFunctions::countPrimaryVertices*
- *Function HelperFunctions::dPhi*
- *Function HelperFunctions::found_non_dummy_sys*
- *Template Function HelperFunctions::getLink*
- *Function HelperFunctions::getListofSystematics*
- *Function HelperFunctions::getMCShowerType*
- *Function HelperFunctions::getPrimaryVertex(const xAOD::VertexContainer *, MsgStream&)*
- *Function HelperFunctions::getPrimaryVertex(const xAOD::VertexContainer *)*
- *Function HelperFunctions::getPrimaryVertexLocation(const xAOD::VertexContainer *)*
- *Function HelperFunctions::getPrimaryVertexLocation(const xAOD::VertexContainer *, MsgStream&)*
- *Function HelperFunctions::getPrimaryVertexZ*
- *Function HelperFunctions::has_exact*
- *Template Function HelperFunctions::isAvailable(std::string, xAOD::TEvent *, xAOD::TStore *)*
- *Template Function HelperFunctions::isAvailable(std::string, xAOD::TEvent *, xAOD::TStore *, MsgStream&)*
- *Function HelperFunctions::isAvailableMetaData*
- *Function HelperFunctions::isFilePrimaryxAOD*
- *Function HelperFunctions::jetReclustering*
- *Function HelperFunctions::jetTrimming(const xAOD::Jet *, double, double, fastjet::JetAlgorithm)*
- *Function HelperFunctions::jetTrimming(const xAOD::JetContainer *, double, double, fastjet::JetAlgorithm)*
- *Template Function HelperFunctions::makeDeepCopy*
- *Template Function HelperFunctions::makeSubsetCont(T1 *&, T2 *&, const std::string&, HelperClasses::ToolName)*
- *Template Function HelperFunctions::makeSubsetCont(T1 *&, T2 *&, MsgStream&, const std::string&, HelperClasses::ToolName)*
- *Function HelperFunctions::msg*
- *Function HelperFunctions::passPrimaryVertexSelection*

Variables

Namespace Trig

Namespace TrigConf

Namespace xAH

Contents

Classes

- *Struct JetContainer::btagOpPoint*
- *Struct OnlineBeamSpotTool::LBData*
- *Class Algorithm*
- *Class Cluster*
- *Class ClusterContainer*
- *Class Electron*
- *Class ElectronContainer*
- *Class EventInfo*
- *Class FatJet*
- *Class FatJetContainer*
- *Class Jet*
- *Class JetContainer*
- *Class L1JetContainer*
- *Class MetContainer*
- *Class Muon*
- *Class MuonContainer*
- *Class OnlineBeamSpotTool*
- *Class Particle*
- *Template Class ParticleContainer*
- *Class Photon*
- *Class PhotonContainer*
- *Class Tau*
- *Class TauContainer*
- *Class TrackContainer*
- *Class TrackPart*
- *Class TruthContainer*
- *Class TruthPart*
- *Class VertexContainer*

Functions

- *Function xAH::addRucio*

Classes and Structs

Struct pt_sort

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Struct Documentation

struct **pt_sort**

Public Functions

inline bool **operator()**(const TLorentzVector &lhs, const TLorentzVector &rhs)

inline bool **operator()**(const TLorentzVector *lhs, const TLorentzVector *rhs)

inline bool **operator()**(const xAOD::IParticle &lhs, const xAOD::IParticle &rhs)

inline bool **operator()**(const xAOD::IParticle *lhs, const xAOD::IParticle *rhs)

Struct JetContainer::btagOpPoint

- Defined in file_xAODAnaHelpers_JetContainer.h

Nested Relationships

This struct is a nested type of *Class JetContainer*.

Struct Documentation

struct **btagOpPoint**

Public Functions

inline **btagOpPoint**(bool mc, const std::string &tagger, const std::string &wp)

inline **~btagOpPoint**()

inline void **setTree**(TTree *tree, const std::string &jetName)

inline void **setBranch**(TTree *tree, const std::string &jetName)

inline void **clear**()

inline void **Fill**(const xAOD::Jet *jet)

Public Members

bool **m_mc**

std::string **m_accessorName**

Jet::*BTaggerOP* **m_op** = *Jet*::*BTaggerOP*::*None*

bool **m_isContinuous**

std::vector<int> ***m_isTag**

std::vector<std::vector<float>> ***m_sf**

std::vector<std::vector<float>> ***m_ineffSf**

Struct OnlineBeamSpotTool::LBData

- Defined in file_xAODAnaHelpers_OnlineBeamSpotTool.h

Nested Relationships

This struct is a nested type of *Class OnlineBeamSpotTool*.

Struct Documentation

struct **LBData**

Public Functions

inline **LBData**(int LBStart, int LBEnd, float BSx, float BSy, float BSz)

Public Members

int **m_LBStart**

int **m_LBEnd**

float **m_BSx**

float **m_BSy**

float **m_BSz**

Class BasicEventSelection

- Defined in file_xAODAnaHelpers_BasicEventSelection.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **BasicEventSelection** : public xAH::*Algorithm*

This algorithm performs the very basic event selection. This should be the first algo in the algo chain. It can create weighted and unweighted cutflow objects to be picked up downstream by other xAH algos, and your own. The selection applied in data only is:

- GRL (can be turned off)
- LAr Error
- Tile Error
- Core Flag

In both data and simulation (MC), the following cuts are applied

- the highest sum p_T^2 primary vertex has 2 or more tracks (see *m_applyPrimaryVertexCut*)
- trigger requirements (see *m_applyTriggerCut*)

For derivations, the metadata can be accessed and added to the cutflow for normalization. The parameters to control the trigger are described in this header file. If one wants to write out some of the trigger information into a tree using *HelpTreeBase*, flags must be set here.

Note: For MC only, the pileup reweight can also be applied.

Public Functions

BasicEventSelection()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

bool **m_isTLAData** = false

Flag to determine when running on TLA data for different handling of TDT.

bool **m_truthLevelOnly** = false

Protection when running on truth xAOD.

bool **m_setAFII** = false

SimulationFlavour will be determined from the sample MetaData, unless AFII or FS is explicitely requested with the following flags.

bool **m_setAF3** = false

bool **m_setFS** = false

bool **m_applyGRLCut** = false

Apply GRL selection.

std::string **m_GRLxml** = ""

Path to GRL XML file.

std::string **m_GRLExcludeList** = ""

Run numbers to skip in GRL.

bool **m_cleanPowheg** = false

Clean Powheg huge weight.

bool **m_reweightSherpa22** = false

Reweight Sherpa 2.2 Samples.

bool **m_doPUreweighting** = false

Reweight pile-up profile μ

bool **m_doPUreweightingSys** = false

std::string **m_lumiCalcFileNames** = ""

Comma separated list of filenames.

std::string **m_PRWFileNames** = ""

Comma separated list of filenames.

bool **m_autoconfigPRW** = false

Automatically configure PRW using config files from SUSYTools instead of using m_PRWFileNames.

bool **m_useCommonPRWFiles** = false

Configure PRW using common files instead of DSID-specific files.

std::string **m_prwActualMu2016File** = ""

actualMu configuration file for the MC16a campaign (2015/2016). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2017File** = ""

actualMu configuration file for the MC16d campaign (2017). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2018File** = ""

actualMu configuration file for the MC16e campaign (2018). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2022File** = ""

actualMu configuration file for the MC23a campaign (2022). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_prwActualMu2023File** = ""

actualMu configuration file for the MC23d campaign (2023). Added to the PRW tool when using PRW autoconfiguration.

std::string **m_commonPRWFileMC20a** = "PileupReweighting/mc20_common/mc20a.284500.physlite.prw.v1.root"

Common PRW file for the MC20a campaign (2015/16). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC20d** =
"PileupReweighting/mc20_common/mc20d.300000.physlite.prw.v1.root"

Common PRW file for the MC20d campaign (2017). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC20e** =
"PileupReweighting/mc20_common/mc20e.310000.physlite.prw.v1.root"

Common PRW file for the MC20e campaign (2018). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC23a** =
"PileupReweighting/mc23_common/mc23a.410000.physlite.prw.v2.root"

Common PRW file for the MC23a campaign (2022). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC23c** =
"PileupReweighting/mc23_common/mc23c.450000.physlite.prw.v1.root"

Common PRW file for the MC23c campaign (2023). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_commonPRWFileMC23d** =
"PileupReweighting/mc23_common/mc23d.450000.physlite.prw.v1.root"

Common PRW file for the MC23d campaign (2023). Added to the PRW tool when using PRW autoconfiguration with common PRW files option.

std::string **m_mcCampaign**

mc16(acd) to bypass the automatic campaign determination from AMI, several campaigns can be separated by a comma. Only used when m_autoconfigPRW is true

std::string **m_periodConfig** = "auto"

Use Period Configuration or auto.

bool **m_checkStreams** = false

Print streamTags (only in debug mode)

int **m_actualMuMin** = -1

The minimum threshold for `EventInfo::actualInteractionsPerCrossing()`

int **m_actualMuMax** = -1

The maximum threshold for `EventInfo::actualInteractionsPerCrossing()`

bool **m_calcBCIDInfo** = false

Calculate distance to nearest empty and unpaired BCIDs.

bool **m_applyPrimaryVertexCut** = false

Enable to apply a primary vertex cut.

int **m_PVNTrack** = 2

Minimum number of tracks from **the** primary vertex (Harmonized Cut)

bool **m_applyEventCleaningCut** = false

bool **m_applyCoreFlagsCut** = false

bool **m_applyJetCleaningEventFlag** = false

recommended way to clean all jets, but especially collections other than EMTopo … equivalent to "loose" jet-by-jet cleaning!

bool **m_applyIsBadBatmanFlag** = false

should only ever be used in 2015 and 2016 data, for analyses which may be of interest for analyses where fake MET can be an issue

bool **m_printBranchList** = false

std::string **m_triggerSelection** = ""

RegEx expression to choose triggers to consider to be cut on with *m_applyTriggerCut*

std::string **m_extraTriggerSelection** = ""

Decisions of triggers which are saved but not cut on.

bool **m_applyTriggerCut** = false

Skip events in which the trigger string *m_triggerSelection* does not fire

bool **m_storeTrigDecisions** = false

Save string of fired triggers matching *m_triggerSelection*

bool **m_storePassL1** = false

Save if any L1 trigger fired, e.g. `"L1_.*"`

bool **m_storePassHLT** = false

Save if any HLT trigger fired, e.g. `"HLT_.*"`

bool **m_storeTrigKeys** = false

Save master, L1, and HLT key.

bool **m_storePrescaleWeight** = true

Save the trigger prescale weight.

std::string **m_derivationName** = ""

The name of the derivation (use this as an override) bool **m_useMetaData** = true

Retrieve and save information on DAOD selection.

std::string **m_metaDataStreamName** = "metadata"

std::string **m_duplicatesStreamName** = "duplicates_tree"

bool **m_checkDuplicatesData** = false

Check for duplicated events in data

bool **m_checkDuplicatesMC** = false

Check for duplicated events in MC

Class BJetEfficiencyCorrector

- Defined in file_xAODAnaHelpers_BJetEfficiencyCorrector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **BJetEfficiencyCorrector** : public xAH::*Algorithm*

Public Functions

BJetEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

EL::StatusCode **executeEfficiencyCorrection**(const xAOD::JetContainer *inJets, const xAOD::EventInfo *eventInfo, bool doNominal)

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

unsigned int **getMCIndex**(int dsid)

void **makeMCIndexMap**(std::string effCalib)

std::string **getFlavorLabel**(const xAOD::Jet &jet) const

Public Members

std::string **m_inContainerName** = ""

std::string **m_inputAlgo** = ""

The name of the vector containing the names of the systematically-varied jet-related containers from the upstream algorithm, which will be processed by this algorithm.

Only jet calibration systematics or any other that create shallow copies of jet containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

std::string **m_systName** = ""

std::string **m_outputSystName** = "BJetEfficiency_Algo"

bool **m_writeSystToMetadata** = false

std::string **m_corrFileName** = "xAODBTaggingEfficiency/13p6TeV/2023-22-13p6TeV-MC21-CDI_Test_2023-08-1_v1.root"

std::string **m_jetAuthor** = "AntiKt4EMPFlowJets"

float **m_minPt** = 20e3

Minimum pT in MeV for taggable jets.

std::string **m_taggerName** = "DL1r"

bool **m_useDevelopmentFile** = true

bool **m_coneFlavourLabel** = true

std::string **m_systematicsStrategy** = "SFEigen"

bool **m_errorOnTagWeightFailure** = true

BTaggingSelectionTool throws an error on missing tagging weights. If false, a warning is given instead.

bool **m_alwaysGetTagWeight** = false

Decorate tag weights even if we're not doing pseudocontinuous b-tagging.

std::string **m_operatingPt** = "FixedCutBEff_70"

Operating point.

std::string **m_operatingPtCDI** = ""

Operating point that CDI will understand.

bool **m_getScaleFactors** = false

will only get scale factors for calibrated working points

bool **m_useContinuous** = false

will get tagWeight, quantile, SF and InefficiencySF

std::string **m_decor** = "BTag"

The decoration key written to passing objects.

bool **m_tagDecisionOnly** = false

Only apply b-tag decision decoration; don't retrieve scale factors (Not recommended. For expert use.)

bool **m_setMapIndex** = false

Select an efficiency map for use in MC/MC and inefficiency scale factors, based on user specified selection of efficiency maps.

std::string **m_DSIDtoGenerator_filename** = "xAODAnaHelpers/DSIDtoGenerator.txt"

float **m_orBJetPtUpperThres** = -1

upper pt threshold of b-jet in OR in unit of GeV, negative value means no pt threshold

std::string **m_EfficiencyCalibration** = ""

Calibration to use for MC (EfficiencyB/C/T/LightCalibrations), "auto" to determine from sample name (multiple samples can be provided as long as they are separated by ';')

Example: "410470;410250;410558;410464" (Pythia8,Sherpa22,Herwig7,MG)

std::string **m_EigenvectorReductionB** = "Loose"

To change NP scheme for b-tagging systematics - Loose is the default value in athena.

std::string **m_EigenvectorReductionC** = "Loose"

std::string **m_EigenvectorReductionLight** = "Loose"

Class ClusterHistsAlgo

- Defined in file_xAODAnaHelpers_ClusterHistsAlgo.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **ClusterHistsAlgo** : public xAH::*Algorithm*

Public Functions

ClusterHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_detailStr** = ""

Class DebugTool

- Defined in file_xAODAnaHelpers_DebugTool.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **DebugTool** : public xAH::*Algorithm*

Public Functions

DebugTool()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

bool **m_printStore** = false

Class ElectronCalibrator

- Defined in file_xAODAnaHelpers_ElectronCalibrator.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **ElectronCalibrator** : public xAH::*Algorithm*

This is the algorithm class used to calibrate electrons.

In a nutshell, this algorithm performs the following actions:

- retrieves an `xAOD::ElectronContainer` from either `TEvent` or `TStore`
- makes a shallow copy container and fills it with energy-and-direction calibrated electrons using the `EgammaCalibrationAndSmearingTool` in Tools Used
- saves the shallow copy container to `TStore` from where it can be retrieved by algorithms downstream via name lookup

Public Functions

ElectronCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

The name of the input container for this algorithm to read from `TEvent` or `TStore`

std::string **m_outContainerName** = ""

The name of the nominal output container written by the algorithm to `TStore`

If the algorithm applies systematic variations, for each shallow copy saved to `TStore`, the systematic name will be appended to this.

bool **m_sort** = true

Sort the processed container elements by transverse momentum.

std::string **m_inputAlgoSystNames** = ""

The name of the vector containing the names of the systematically-varied containers from the upstream algorithm, which will be processed by this algorithm.

This vector is retrieved from the TStore. If left blank, it means there is no upstream algorithm which applies systematics. This is the case when processing straight from the original xAOD or DxAOD.

std::string **m_outputAlgoSystNames** = "ElectronCalibrator_Syst"

The name of the vector containing the names of the systematically-varied containers created by by this algorithm.

If *m_systName* is empty, the vector will contain only an empty string. When running on systematics, this is the string a downstream algorithm needs to process electrons.

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

std::string **m_esModel** = ""

std::string **m_decorrelationModel** = ""

bool **m_applyIsolationCorrection** = false

Apply isolation correction, not needed by default.

Class ElectronCutBasedPIDManager

- Defined in file_xAODAnaHelpers_ParticlePIDManager.h

Class Documentation

class **ElectronCutBasedPIDManager**

Public Functions

ElectronCutBasedPIDManager()

ElectronCutBasedPIDManager(std::string WP, bool debug = false)

~ElectronCutBasedPIDManager()

StatusCode **setupWPs**(bool configTools, std::string selector_name = "")

StatusCode **setDecorations**(const xAOD::Electron *electron)

inline const std::string **getSelectedWP**()

inline std::multimap<std::string, AsgElectronIsEMSelector*> **getAllWPTools**()

inline std::multimap<std::string, AsgElectronIsEMSelector*> **getValidWPTools**()

inline const std::set<std::string> **getAllWPs**()

inline const std::set<std::string> **getValidWPs**()

Class ElectronEfficiencyCorrector

- Defined in file_xAODAnaHelpers_ElectronEfficiencyCorrector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **ElectronEfficiencyCorrector** : public xAH::*Algorithm*

This is the algorithm class that applies generic corrections to electrons. At the moment, only data/MC efficiency correction is included (electron trigger SF and others will follow. . .).

In a nutshell, this algorithm performs the following actions:

- retrieves an `xAOD::ElectronContainer` from either `TEvent` or `TStore`
- adds a scale factor (SF) decoration for each electron in the input container calculated via the `AsgElectronEfficiencyCorrectionTool` in Tools Used
- the nominal SF and all the systematically-varied ones are saved as a `vector<double>` decoration for each electron

Note: Bear in mind that this algorithm must be called after *`ElectronSelector`*. In fact, the configuration file(s) being used must have the same working point as the one chosen in the selector.

Public Functions

ElectronEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **executeSF**(const xAOD::ElectronContainer *inputElectrons, bool nominal, bool writeSystNames)

Public Members

std::string **m_inContainerName** = ""

The name of the input container for this algorithm to read from `TEvent` or `TStore`

std::string **m_inputSystNamesElectrons**

The name of the vector containing the names of the systematically-varied electrons-related containers from the upstream algorithm, which will be processed by this algorithm.

Only electron calibration systematics or any other that create shallow copies of electron containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systValPID** = 0.0

float **m_systValIso** = 0.0

float **m_systValReco** = 0.0

float **m_systValTrig** = 0.0

std::string **m_systNamePID** = ""

std::string **m_systNameIso** = ""

std::string **m_systNameReco** = ""

std::string **m_systNameTrig** = ""

std::string **m_outputSystNamesPID** = "EleEffCorr_PIDSyst"

std::string **m_outputSystNamesIso** = "EleEffCorr_IsoSyst" std::string **m_outputSystNamesReco** = "EleEffCorr_RecoSyst"

std::string **m_outputSystNamesTrig** = "EleEffCorr_TrigSyst"

std::string **m_correlationModel** = "FULL"
Systematic correlation model.

std::string **m_WorkingPointPID** = ""
PID working point (LooseBLayer, Medium, Tight)

std::string **m_WorkingPointIso** = ""
Isolation working point.

std::string **m_WorkingPointReco** = ""
Reconstruction working point (Reconstruction only)

std::string **m_WorkingPointTrig** = ""
Trigger working point.

bool **m_usePerElectronTriggerSFs** = true

std::string **m_overrideMapFilePath** = ""
Override corrections map file (not recommended)

Class ElectronHistsAlgo

- Defined in file_xAODAnaHelpers_ElectronHistsAlgo.h

Inheritance Relationships

Base Type

- `public IParticleHistsAlgo` (*Class IParticleHistsAlgo*)

Class Documentation

class **ElectronHistsAlgo** : public *IParticleHistsAlgo*

Public Functions

ElectronHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **execute**()
Calls execute<IParticleContainer>

virtual EL::StatusCode **AddHists**(std::string name)
Calls AddHists<IParticleHists>

Parameters **name** – Name of the systematic

Class ElectronSelector

- Defined in file_xAODAnaHelpers_ElectronSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **ElectronSelector** : public xAH::*Algorithm*

This is the algorithm class that selects electrons according to user's choice.

In a nutshell, this algorithm performs the following actions:

- retrieves an `xAOD::ElectronContainer` from either `TEvent` or `TStore`
- iterates over the input container, and if electron passes selection, copies it in a `ConstDataVector(SG::VIEW_ELEMENTS)` container. Otherwise, the electron is skipped
- saves the view container to `TStore`, from where it can be retrieved by algorithms downstream via a name lookup

Public Functions

ElectronSelector()

~ElectronSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::ElectronContainer *inElectrons, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::ElectronContainer> *selectedElectrons)

virtual int **passCuts**(const xAOD::Electron *electron, const xAOD::Vertex *primaryVertex)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

The name of the input container for this algorithm read from `TEvent` or `TStore`

std::string **m_outContainerName** = ""

The name of the nominal output container written by the algorithm to `TStore`

std::string **m_inputAlgoSystNames** = ""

The name of the vector containing the names of the systematically-varied containers from the upstream algorithm, which will be processed by this algorithm.

This vector is retrieved from the `TStore`. If left blank, it means there is no upstream algorithm which applies systematics. This is the case when processing straight from the original `xAOD` or `DxAOD`.

std::string **m_outputAlgoSystNames** = "ElectronSelector_Syst"

The name of the vector containing the names of the systematically-varied containers created by by this algorithm.

If *m_systName* is empty, the vector will contain only an empty string. When running on systematics, this is the string a downstream algorithm needs to process electrons.

bool **m_decorateSelectedObjects** = true

Adds a `passSel` decoration for objects that pass selection.

bool **m_createSelectedContainer** = false

Fill using a read-only container (`SG::VIEW_ELEMENTS`) to `TStore`

int **m_nToProcess** = -1

Number of objects to process, set `n=-1` to look at all.

int **m_pass_min** = -1

Require event to have minimum number of objects passing selection.

int **m_pass_max** = -1

Require event to have maximum number of objects passing selection.

float **m_pT_max** = 1e8

[MeV] Require objects to have maximum transverse momentum threshold

float **m_pT_min** = 1e8

[MeV] Require objects to have minimum transverse momentum threshold

float **m_eta_max** = 1e8

Require objects to have maximum $|\eta|$ value

bool **m_vetoCrack** = true

Require objects to have $|\eta|$ outside the crack region using `caloCluster->eta()`

float **m_d0_max** = 1e8

Require objects to have a maximum d_0 [mm] (transverse impact parameter)

float **m_d0sig_max** = 1e8

Require objects to have a maximum d_0 significance at BL

float **m_z0sintheta_max** = 1e8

Require objects to have maximum $z_0 \sin(\theta)$ [mm] (longitudinal impact paramter) at BL - corrected with vertex info

bool **m_doAuthorCut** = true

Perform author kinematic cut.

bool **m_doOQCut** = true

Perform object quality cut.

bool **m_readIDFlagsFromDerivation** = false

To read electron PID decision from DAOD, rather than recalculate with tool.

bool **m_doModifiedEleId** = false

To correct egamma bug, see ATLSUSYSW-445.

bool **m_doLHPID** = true

Instantiate and perform the electron Likelihood PID.

bool **m_doLHPIDcut** = false

Cut on electron Likelihood PID (recommended)

std::string **m_LHOperatingPoint** = "Loose"

Loosest Likelihood PID operating point to save.

bool **m_doCutBasedPID** = false

Instantiate and perform the electron cut-based PID.

bool **m_doCutBasedPIDcut** = false

Cut on electron cut-based PID.

std::string **m_CutBasedOperatingPoint** = "Loose"

Loosest cut-based PID operating point to save.

std::string **m_MinIsoWPCut** = ""

reject objects which do not pass this isolation cut - default = "" (no cut)

std::string **m_IsoWPList** = "FCLoose,FCTight,Gradient,FCHighPtCaloOnly"

decorate objects with `isIsolated_*` flag for each WP in this input list - default = all current ASG WPs

std::string **m_CaloIsoEff** = "0.1*x+90"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_TrackIsoEff** = "98"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_CaloBasedIsoType** = "topoetcone20"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_TrackBasedIsoType** = "ptvarcone20"

to define a custom WP - make sure `"UserDefined"` is added in *m_IsoWPList*

std::string **m_singleElTrigChains** = ""

A comma-separated string w/ alll the HLT single electron trigger chains for which you want to perform the matching. This is passed by the user as input in configuration If left empty (as it is by default), no trigger matching will be attempted at all.

std::string **m_diElTrigChains** = ""

A comma-separated string w/ alll the HLT di-electron trigger chains for which you want to perform the matching. This is passed by the user as input in configuration If left empty (as it is by default), no trigger matching will be attempted at all.

double **m_minDeltaR** = 0.07

Recommended threshold for egamma triggers: see https://svnweb.cern.ch/trac/atlasoff/browser/Trigger/TrigAnalysis/TriggerMatchingTool/trunk/src/TestMatchingToolAlg.cxx.

bool **m_applyCrackVetoCleaning** = false

Apply fix to EGamma Crack-Electron topocluster association bug for MET (PFlow) / false by default.

bool **m_merged_electrons** = false

Element links need to be updated if merged electrons are used (LRT + std) / false by default.

std::string **m_trigInputPrefix** = ""
Input prefix of trigger decision tool.

std::string **m_isoDecSuffix** = ""

Class ClusterInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **ClusterInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

Public Functions

inline **ClusterInfoSwitch**(const std::string configStr)

inline virtual **~ClusterInfoSwitch**()

Protected Functions

virtual void **initialize**()

Class ElectronInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **ElectronInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

The *HelperClasses::IParticleInfoSwitch* class for Electron Information.

Parameter	Pattern	Match
m_trigger	trigger	exact
m_isolation	isolation	exact
m_isolationKinematics	isolationKinematics	exact
m_PID	PID	exact
m_trackparams	trackparams	exact
m_trackhitcont	trackhitcont	exact
m_effSF	effSF	exact
m_PIDWPs[XYZ]	PID_XYZ	pattern
m_PIDSFWPs[XYZ]	PIDSF_XYZ	pattern
m_isolWPs[""]	**ISOL_**	exact
m_isolWPs[""]	ISOL_NONE	exact
m_isolWPs[XYZ]	ISOL_XYZ	pattern
m_trigWPs[XYZ]	TRIG_XYZ	pattern
m_passSel	passSel	exact
m_passOR	passOR	exact

Note: `PID`, `isolation` and `effSF` switches do not enable any additional output by themselves. They require additional working point pattern using `PID_XYZ` for PID working points, `PIDSF_XYZ` for PID scale factors, `ISOL_XYZ` for isolation working points and scale factors, and `TRIG_XYZ` for trigger scale factors. `XYZ` in the pattern should be replaced using the working point name, for example:

```
m_configStr = "... PID_LHMedium PIDSF_MediumLLH ..."
```

will define the `LHMedium` PID working point and the accompanying scale factors. Note that not all PID working points have scale factors available.

Isolation supports `NONE` or empty option which will enable scale factors without additional isolation requirements, for example:

```
m_configStr = "... ISOL_NONE ISOL_Loose ..."
```

will define the `Loose` isolation working point status branch, and scale factors without isolation requirements and using the `Loose` WP.

Public Functions

inline **ElectronInfoSwitch**(const std::string configStr)

inline virtual **~ElectronInfoSwitch**()

Public Members

bool **m_trigger**

bool **m_isolation**

bool **m_isolationKinematics**

bool **m_quality**

bool **m_PID**

bool **m_recoparams**

bool **m_trackparams**

bool **m_trackhitcont**

bool **m_effSF**

bool **m_promptlepton**

std::vector<std::string> **m_PIDWPs**

std::vector<std::string> **m_PIDSFWPs**

std::vector<std::string> **m_isolWPs**

std::vector<std::string> **m_trigWPs**

bool **m_passSel**

bool **m_passOR**

bool **m_doLRT**

bool **m_closeByCorr**

Protected Functions

virtual void **initialize**()

Template Class EnumParser

- Defined in file_xAODAnaHelpers_HelperClasses.h

Class Documentation

template<typename **T**>

class **EnumParser**

template enum parser. Copied from: http://stackoverflow.com/a/726681

Public Functions

EnumParser()

inline *T* **parseEnum**(const std::string &value)

EnumParser()

EnumParser()

EnumParser()

EnumParser()

EnumParser()

EnumParser()

Class EventInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::InfoSwitch` (*Class InfoSwitch*)

Class Documentation

class **EventInfoSwitch** : public *HelperClasses*::*InfoSwitch*

The `HelperClasses::InfoSwitch` struct for Event Information.

Parameter	Pattern	Match
m_noDataInfo	noDataInfo	exact
m_eventCleaning	eventCleaning	exact
m_bcidInfo	bcidInfo	exact
m_pileup	pileup	exact
m_pileupsys	pileupsys	exact
m_shapeEM	shapeEM	exact
m_shapeEMPFLOW	shapeEMPFLOW	exact
m_shapeLC	shapeLC	exact
m_truth	truth	exact
m_caloClus	caloClusters	exact
m_weightsSys	weightsSys	exact
m_beamspotweight	beamspotweight	exact

Public Functions

inline **EventInfoSwitch**(const std::string configStr)

Public Members

bool **m_noDataInfo**

bool **m_eventCleaning**

bool **m_bcidInfo**

bool **m_pileup**

bool **m_pileupsys**

bool **m_shapeEM**

bool **m_shapeEMPFLOW**

bool **m_shapeLC**

bool **m_truth**

bool **m_caloClus**

bool **m_weightsSys**

bool **m_beamspotweight**

Protected Functions

void **initialize**()

Class InfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Derived Types

- `public HelperClasses::EventInfoSwitch` (*Class EventInfoSwitch*)
- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)
- `public HelperClasses::METInfoSwitch` (*Class METInfoSwitch*)
- `public HelperClasses::TrackInfoSwitch` (*Class TrackInfoSwitch*)
- `public HelperClasses::TriggerInfoSwitch` (*Class TriggerInfoSwitch*)

Class Documentation

class **InfoSwitch**

A struct that is used for parsing configuration strings and assigning booleans to various properties. Currently used in plotting code.

Strings are used to turn on and off histograms and branches in the tree The following structs hold the bools used to control the content and also have the string which is necessary to turn a set on. See the derived members for more information about what is supported. Each derived member should provide a table of parameters, patterns, and type of matching scheme used. The pattern will use standard PCRE-syntax when appropriate.

We support two major matching schemes:

Exact If a variable is matched exactly to a string, then a boolean is set to True or False based on whether an exact match exists or not.

Partial If a variable is partially matched to a string, then there is some specific pattern we are extracting that will succeed the partial match that determines what the variable will be set to (usually not a bool).

Subclassed by *HelperClasses::EventInfoSwitch*, *HelperClasses::IParticleInfoSwitch*, *HelperClasses::METInfoSwitch*, *HelperClasses::TrackInfoSwitch*, *HelperClasses::TriggerInfoSwitch*

Public Functions

inline **InfoSwitch**(const std::string configStr)

Constructor. Take in input string, create vector of tokens.

Parameters **configStr** – The configuration string to split up.

inline bool **has_exact**(const std::string flag)

Search for an exact match in *m_configDetails*.

Parameters **flag** – The string we search for.

inline bool **has_match**(const std::string flag)

Search for a partial match in *m_configStr*.

Parameters **flag** – The string we search for.

std::string **get_working_point**(const std::string flag)

Search for a single flag in *m_configDetails* and parse out the working point.

Parameters **flag** – The string we search for.

std::vector<std::string> **get_working_points**(const std::string flag)

Search for multiple flags in *m_configDetails* and parse out the working points.

Parameters **flag** – The string we search for.

Protected Attributes

const std::string **m_configStr**

The input configuration string from which we split up into tokens.

std::set<std::string> **m_configDetails**

The vector of tokens from which we search through for finding matches.

Class IParticleInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::InfoSwitch` (*Class InfoSwitch*)

Derived Types

- `public HelperClasses::ClusterInfoSwitch` (*Class ClusterInfoSwitch*)
- `public HelperClasses::ElectronInfoSwitch` (*Class ElectronInfoSwitch*)
- `public HelperClasses::JetInfoSwitch` (*Class JetInfoSwitch*)
- `public HelperClasses::MuonInfoSwitch` (*Class MuonInfoSwitch*)
- `public HelperClasses::PhotonInfoSwitch` (*Class PhotonInfoSwitch*)
- `public HelperClasses::TauInfoSwitch` (*Class TauInfoSwitch*)
- `public HelperClasses::TruthInfoSwitch` (*Class TruthInfoSwitch*)

Class Documentation

class **`IParticleInfoSwitch`** : public *HelperClasses*::*InfoSwitch*

The *`HelperClasses::InfoSwitch`* struct for IParticle Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_numLeading	NLeading	partial
m_useTheS	useTheS	exact

Note: `m_numLeading` requires a number `XX` to follow it, defining the number of leading partiles and associate it with that variable.

For example:

```
m_configStr = "... NLeading4 ..."
```

will define `int m_numLeading = 4`.

Subclassed by *HelperClasses::ClusterInfoSwitch*, *HelperClasses::ElectronInfoSwitch*, *HelperClasses::JetInfoSwitch*, *HelperClasses::MuonInfoSwitch*, *HelperClasses::PhotonInfoSwitch*, *HelperClasses::TauInfoSwitch*, *HelperClasses::TruthInfoSwitch*

Public Functions

inline **IParticleInfoSwitch**(const std::string configStr)

inline virtual **~IParticleInfoSwitch**()

Public Members

bool **m_noMultiplicity**

bool **m_kinematic**

int **m_numLeading**

bool **m_useTheS**

Protected Functions

virtual void **initialize**()

Class JetInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **JetInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

The `HelperClasses::IParticleInfoSwitch` class for Jet Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_trigger	trigger	exact
m_substructure	substructure	exact
m_ntrimsubjets	ntrimsubjets	exact
m_bosonCount	bosonCount	exact
m_VTags	VTags	exact

continues on next page

Table 2 – continued from previous page

Parameter	Pattern	Match
m_rapidity	rapidity	exact
m_clean	clean	exact
m_cleanLight	cleanLight	exact
m_cleanLightLLP	cleanLightLLP	exact
m_cleanTrig	cleanTrig	exact
m_timing	timing	exact
m_energy	energy	exact
m_energyLight	energyLight	exact
m_scales	scales	exact
m_constscaleEta	constscaleEta	exact
m_detectorEta	detectorEta	exact
m_resolution	resolution	exact
m_truth	truth	exact
m_truthDetails	truth_details	exact
m_layer	layer	exact
m_trackPV	trackPV	exact
m_trackAll	trackAll	exact
m_chargedPFOPV	chargedPFOPV	exact
m_jvt	JVT	exact
m_NNJvt	NNJvt	exact
m_sfJVTName	sfJVT	partial
m_sffJVTName	sffJVT	partial
m_allTrack	allTrack	exact
m_allTrackPVSel	allTrackPVSel	exact
m_allTrackDetail	allTrackDetail	exact
m_constituent	constituent	exact
m_constituentAll	constituentAll	exact
m_flavorTag	flavorTag	exact
m_flavorTagHLT	flavorTagHLT	exact
m_sfFTagFix	sfFTagFix	partial
m_sfFTagFlt	sfFTagFlt	partial
m_sfFTagHyb	sfFTagHyb	partial
m_jetBTag	jetBTag	partial
m_area	area	exact
m_JVC	JVC	exact
m_tracksInJet	tracksInJet	partial
m_trackJetName	trackJetName	partial
m_hltVtxComp	hltVtxComp	exact
m_onlineBS	onlineBS	exact
m_onlineBSTool	onlineBSTool	exact
m_charge	charge	exact
m_passSel	passSel	exact
m_passOR	passOR	exact
m_vsLumiBlock	vsLumiBlock	exact
m_vsActualMu	vsActualMu	exact
m_lumiB_runN	lumiB_runN	exact
m_byAverageMu	byAverageMu	exact
m_byEta	byEta	exact
m_etaPhiMap	etaPhiMap	exact

continues on next page

Table 2 – continued from previous page

Parameter	Pattern	Match
m_muonCorrection	muonCorrection	exact

`trackJetName` expects one or more track jet container names separated by an underscore. For example, the string `trackJetName_GhostAntiKt2TrackJet_GhostVR30Rmax4Rmin02TrackJet` will set the attriubte `m_trackJetNames` to `{"GhostAntiKt2TrackJet", "GhostVR30Rmax4Rmin02TrackJet"}`.

Note: `sfJVT` requires a working point after it, for example:

```
m_configStr = "... sfJVTMedium ..."
```

`jetBTag` expects the format `jetBTag_tagger_type_AABB..MM..YY.ZZ`. This will create a vector of working points (AA, BB, CC, …, ZZ) associated with that tagger. Several entries can be given. For example:

m_configStr = "… jetBTag_DL1r_FixedCutBEff_60707785 …"

Public Functions

inline **JetInfoSwitch**(const std::string configStr)

inline virtual **~JetInfoSwitch**()

Public Members

bool **m_trigger**

bool **m_substructure**

bool **m_ntrimsubjets**

bool **m_bosonCount**

bool **m_VTags**

bool **m_rapidity**

bool **m_clean**

bool **m_cleanLight**

bool **m_cleanLLP**

bool **m_cleanTrig**

bool m_timing

bool m_energy

bool m_energyLight

bool m_scales

bool m_constscaleEta

bool m_detectorEta

bool m_resolution

bool m_truth

bool m_truthDetails

bool m_layer

bool m_trackPV

bool m_trackAll

bool m_fJvt

bool m_chargedPFOPV

bool m_jvt

bool m_NNJvt

bool m_allTrack

bool m_allTrackDetail

bool m_allTrackPVSel

bool m_constituent

bool m_constituentAll

bool **m_flavorTag**

bool **m_flavorTagHLT**

bool **m_btag_jettrk**

bool **m_jetFitterDetails**

bool **m_svDetails**

bool **m_ipDetails**

bool **m_tracksInJet**

bool **m_hltVtxComp**

bool **m_onlineBS**

bool **m_onlineBSTool**

bool **m_charge**

bool **m_passSel**

bool **m_passOR**

bool **m_etaPhiMap**

bool **m_vsLumiBlock**

bool **m_vsActualMu**

bool **m_lumiB_runN**

bool **m_byEta**

bool **m_byAverageMu**

bool **m_area**

bool **m_JVC**

bool **m_muonCorrection**

std::string **m_trackName**

std::vector<std::string> **m_trackJetNames**

std::string **m_sfJVTName**

std::string **m_sffJVTName**

std::map<std::string, std::vector<std::pair<std::string, uint>>> **m_jetBTag**

std::vector<std::string> **m_jetBTagCts**

Protected Functions

virtual void **initialize**()

Class METInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::InfoSwitch` (*Class InfoSwitch*)

Class Documentation

class **METInfoSwitch** : public *HelperClasses*::*InfoSwitch*

The *HelperClasses::InfoSwitch* struct for Missing E_T Information.

Parameter	Pattern	Match
m_metClus	metClus	exact
m_metTrk	metTrk	exact
m_sigClus	sigClus|all	exact
m_sigTrk	sigTrk|all	exact
m_sigResolutionClus	sigResolutionClus|all	exact
m_sigResolutionTrk	sigResolutionTrk|all	exact
m_refEle	refEle|all	exact
m_refGamma	refGamma|all	exact
m_refTau	refTau|all	exact
m_refMuons	refMuons|all	exact
m_refJet	refJet|all	exact
m_refJetTrk	refJetTrk	exact
m_softClus	softClus|all	exact
m_softTrk	softTrk|all	exact
m_noExtra	noExtra	exact

Note: For all except *m_refJetTrk*, you can pass in the string `"all"` to enable all information. You can force only calocluster- or track-based MET using *m_metClus* or *m_metTrk*.

Public Functions

inline **METInfoSwitch**(const std::string configStr)

Public Members

bool **m_metClus**

bool **m_metTrk**

bool **m_sigClus**

bool **m_sigTrk**

bool **m_sigResolutionClus**

bool **m_sigResolutionTrk**

bool **m_refEle**

bool **m_refGamma**

bool **m_refTau**

bool **m_refMuons**

bool **m_refJet**

bool **m_refJetTrk**

bool **m_softClus**

bool **m_softTrk**

bool **m_noExtra**

Protected Functions

void **initialize**()

Class MuonInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **MuonInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

The *`HelperClasses::IParticleInfoSwitch`* class for Muon Information.

Parameter	Pattern	Match
m_trigger	trigger	exact
m_isolation	isolation	exact
m_isolationKinematics	isolationKinematics	exact
m_quality	quality	exact
m_recoparams	recoparams	exact
m_trackparams	trackparams	exact
m_trackhitcont	trackhitcont	exact
m_effSF	effSF	exact
m_energyLoss	energyLoss	exact
m_recoWPs[XYZ]	RECO_XYZ	pattern
m_isolWPs[""]	**ISOL_**	exact
m_isolWPs[""]	ISOL_NONE	exact
m_isolWPs[XYZ]	ISOL_XYZ	pattern
m_trigWPs[XYZ]	TRIG_XYZ	pattern
m_passSel	passSel	exact
m_passOR	passOR	exact

Note: `quality`, `isolation` and `effSF` switches do not enable any additional output by themselves. They require additional working point pattern using `RECO_XYZ` for quality working points and scale factors, `ISOL_XYZ` for isolation working points and scale factors, and `TRIG_XYZ` for trigger scale factors. `XYZ` in the pattern should be replaced using the working point name, for example:

```
m_configStr = "... RECO_Medium ..."
```

will define the `Medium` quality working point and the accompanying scale factors.

Isolation supports `NONE` or empty option which will enable scale factors without additional isolation requirements, for example:

```
m_configStr = "... ISOL_NONE ISOL_Loose ..."
```

will define the `Loose` isolation working point status branch, and scale factors without isolation requirements and using the `Loose` WP.

Public Functions

inline **MuonInfoSwitch**(const std::string configStr)

inline virtual **~MuonInfoSwitch**()

Public Members

bool **m_trigger**

bool **m_isolation**

bool **m_isolationKinematics**

bool **m_quality**

bool **m_trackparams**

bool **m_trackhitcont**

bool **m_effSF**

bool **m_energyLoss**

bool **m_promptlepton**

std::vector<std::string> **m_recoWPs**

std::vector<std::string> **m_isolWPs**

std::vector<std::string> **m_trigWPs**

bool **m_passSel**

bool **m_passOR**

bool **m_doLRT**

bool **m_closeByCorr**

bool **m_recoEff_sysNames**

bool **m_isoEff_sysNames**

bool **m_trigEff_sysNames**

bool **m_ttvaEff_sysNames**

Protected Functions

virtual void **initialize**()

Class PhotonInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **PhotonInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

The `HelperClasses::IParticleInfoSwitch` class for Photon Information.

Parameter	Pattern	Match
m_isolation	isolation	exact
m_PID	PID	exact
m_purity	purity	exact
m_effSF	effSF	exact
m_trigger	trigger	exact
m_isoCones	isoCone	partial

Note: `isoCone` can be repeated but requires a number after it, for example:

```
m_configStr = "... isoCone20 isoCone40 ..."
```

which will define `std::vector<int> m_isoCones = {20,40}`.

Public Functions

inline **PhotonInfoSwitch**(const std::string configStr)

inline virtual **~PhotonInfoSwitch**()

Public Members

bool **m_isolation**

bool **m_PID**

bool **m_purity**

bool **m_effSF**

bool **m_trigger**

std::vector<std::string> **m_isoCones**

Protected Functions

virtual void **initialize**()

Class TauInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **TauInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

The `HelperClasses::IParticleInfoSwitch` struct for Tau Information.

Note: `identification` and `effSF` switches do not enable any additional output by themselves. They require additional working point pattern using `TAUEFF_XYZ` for combined scale factors, and `TRIG_XYZ` for trigger scale factors. `XYZ` in the pattern should be replaced using the working point name, for example:

```
m_configStr = "... TAUEFF_EleOLRElectronEleRNNLoose_TauIDMedium ... TRIG_
↪EleOLRElectronEleRNNMedium_TauIDLoose_TrigMyTriggerMenu"
```

Notice that the working point for TAUEFF is a combination of two working points from EleOLRElectron and TauID.

Public Functions

inline **TauInfoSwitch**(const std::string configStr)

inline virtual **~TauInfoSwitch**()

Public Members

bool **m_trigger**

bool **m_JetID**

bool **m_EleVeto**

bool **m_xahTauJetMatching**

bool **m_trackAll**

bool **m_trackparams**

bool **m_trackhitcont**

bool **m_effSF**

std::vector<std::string> **m_tauEffWPs**

std::vector<std::string> **m_trigWPs**

Protected Functions

virtual void **initialize**()

Class TrackInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::InfoSwitch` (*Class InfoSwitch*)

Class Documentation

class **TrackInfoSwitch** : public *HelperClasses*::*InfoSwitch*

The *HelperClasses::InfoSwitch* struct for Track Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_fitpars	fitpars	exact
m_numbers	numbers	exact
m_vertex	vertex	exact
m_useTheS	useTheS	exact

Public Functions

inline **TrackInfoSwitch**(const std::string configStr)

Public Members

bool **m_noMultiplicity**

bool **m_kinematic**

bool **m_fitpars**

bool **m_numbers**

bool **m_vertex**

bool **m_useTheS**

Protected Functions

void **initialize**()

Class TriggerInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::InfoSwitch` (*Class InfoSwitch*)

Class Documentation

class **TriggerInfoSwitch** : public *HelperClasses*::*InfoSwitch*

The *HelperClasses::InfoSwitch* struct for Trigger Information.

Parameter	Pattern	Match
m_basic	basic	exact
m_menuKeys	menuKeys	exact
m_passTriggers	passTriggers	exact
m_passTrigBits	passTrigBits	exact
m_prescales	prescales	exact
m_prescalesLumi	prescalesLumi	exact

Note: `m_prescales` contains information from the `TrigDecisionTool` for every trigger used in event selection and event trigger-matching. `m_prescalesLumi` contains information retrieved from the pile-up reweighting tool based on the actual luminosities of triggers.

Public Functions

inline **TriggerInfoSwitch**(const std::string configStr)

Public Members

bool **m_basic**

bool **m_menuKeys**

bool **m_passTriggers**

bool **m_passTrigBits**

bool **m_prescales**

bool **m_prescalesLumi**

Protected Functions

void **initialize**()

Class TruthInfoSwitch

- Defined in file_xAODAnaHelpers_HelperClasses.h

Inheritance Relationships

Base Type

- `public HelperClasses::IParticleInfoSwitch` (*Class IParticleInfoSwitch*)

Class Documentation

class **TruthInfoSwitch** : public *HelperClasses*::*IParticleInfoSwitch*

The *`HelperClasses::InfoSwitch`* struct for Truth Information.

Parameter	Pattern	Match
m_noMultiplicity	noMultiplicity	exact
m_kinematic	kinematic	exact
m_type	type	exact
m_bVtx	bVtx	exact
m_parents	parents	exact
m_children	children	exact
m_dressed	dressed	exact
m_origin	origin	exact
m_particleType	particleType	exact
m_pdgIdOnly	pdgIdOnly	exact

Public Functions

inline **TruthInfoSwitch**(const std::string configStr)

Public Members

bool **m_type**

bool **m_bVtx**

bool **m_parents**

bool **m_children**

bool **m_dressed**

bool **m_origin**

bool **m_particleType**

bool **m_pdgIdOnly**

Protected Functions

virtual void **initialize**()

Class HelpTreeBase

- Defined in file_xAODAnaHelpers_HelpTreeBase.h

Class Documentation

class **HelpTreeBase**

Public Functions

HelpTreeBase(xAOD::TEvent *event, TTree *tree, TFile *file, const float units = 1e3, bool debug = false, xAOD::TStore *store = nullptr, std::string nominalTreeName = "nominal")

HelpTreeBase(TTree *tree, TFile *file, xAOD::TEvent *event = nullptr, xAOD::TStore *store = nullptr, const float units = 1e3, bool debug = false, std::string nominalTreeName = "nominal")

virtual **~HelpTreeBase**()

void `AddEvent`(const std::string &detailStr = "")

void `AddTrigger`(const std::string &detailStr = "")

void `AddJetTrigger`(const std::string &detailStr = "")

void `AddMuons`(const std::string &detailStr = "", const std::string &muonName = "muon")

void `AddElectrons`(const std::string &detailStr = "", const std::string &elecName = "el")

void `AddPhotons`(const std::string &detailStr = "", const std::string &photonName = "ph")

void `AddClusters`(const std::string &detailStr = "", const std::string &clusterName = "cl")

void `AddJets`(const std::string &detailStr = "", const std::string &jetName = "jet")

void `AddL1Jets`(const std::string &jetName = "")

void `AddTruthParts`(const std::string &detailStr = "", const std::string &truthName = "xAH_truth")

void `AddTrackParts`(const std::string &detailStr = "", const std::string &trackName = "trk")

void `AddVertices`(const std::string &detailStr = "", const std::string &vertexName = "vertex")

void `AddTruthVertices`(const std::string &detailStr = "", const std::string &vertexName = "truth_vertex")

void `AddFatJets`(const std::string &detailStr = "", const std::string &fatjetName = "fatjet", const std::string &subjetDetailStr = "", const std::string &suffix = "")

Declare a new collection of fatjets to be written to the output tree.

Parameters

- `detailStr` – A (space-separated) list of detail options. These keywords specify exactly which information about each jet is written out. Current influential options are: `kinematic` `substructure` `constituent` `constituentAll`
- `fatjetName` – The (prefix) name of the container. Default: `fatjet`.
- `subjetDetailStr` – List of detail options to pass to the subjet container. See :cpp:member:`HelpTreeBase::AddJets` for list of supported values.

void `AddTruthFatJets`(const std::string &detailStr = "", const std::string &truthFatJetName = "truth_fatjet")

void `AddTaus`(const std::string &detailStr = "", const std::string &tauName = "tau")

void `AddMET`(const std::string &detailStr = "", const std::string &metName = "met")

void `FillEvent`(const xAOD::EventInfo *eventInfo, xAOD::TEvent *event = nullptr, const xAOD::VertexContainer *vertices = nullptr)

void `FillTrigger`(const xAOD::EventInfo *eventInfo)

void `FillJetTrigger`()

void `FillMuons`(const xAOD::MuonContainer *muons, const xAOD::Vertex *primaryVertex, const std::string &muonName = "muon")

void `FillMuon`(const xAOD::Muon *muon, const xAOD::Vertex *primaryVertex, const std::string &muonName = "muon") void **FillElectrons**(const xAOD::ElectronContainer *electrons, const xAOD::Vertex *primaryVertex, const std::string &elecName = "el")

void **FillElectron**(const xAOD::Electron *elec, const xAOD::Vertex *primaryVertex, const std::string &elecName = "el")

void **FillPhotons**(const xAOD::PhotonContainer *photons, const std::string &photonName = "ph")

void **FillPhoton**(const xAOD::Photon *photon, const std::string &photonName = "ph")

void **FillClusters**(const xAOD::CaloClusterContainer *clusters, const std::string &clusterName = "cl")

void **FillCluster**(const xAOD::CaloCluster *cluster, const std::string &clusterName = "cl")

void **FillJets**(const xAOD::JetContainer *jets, int pvLocation = -1, const std::string &jetName = "jet")

void **FillJet**(const xAOD::Jet *jet_itr, const xAOD::Vertex *pv, int pvLocation, const std::string &jetName = "jet")

void **FillLegacyL1Jets**(const xAOD::JetRoIContainer *jets, const std::string &jetName = "L1Jet", bool sortL1Jets = false)

template<typename **T**>
inline void **FillPhase1L1Jets**(*T* *&jets, const std::string &jetName = "L1Jet", bool sortL1Jets = false)

void **FillTruth**(const xAOD::TruthParticleContainer *truth, const std::string &truthName = "xAH_truth")

void **FillTruth**(const xAOD::TruthParticle *truthPart, const std::string &truthName)

void **FillTracks**(const xAOD::TrackParticleContainer *tracks, const std::string &trackName = "trk")

void **FillTrack**(const xAOD::TrackParticle *trackPart, const std::string &trackName)

void **FillVertices**(const xAOD::VertexContainer *vertices, const std::string &vertexName = "vertex")

void **FillTruthVertices**(const xAOD::TruthVertexContainer *truthVertices, const std::string &truthVertexName = "truth_vertex")

void **FillFatJets**(const xAOD::JetContainer *fatJets, int pvLocation = 0, const std::string &fatjetName = "fatjet", const std::string &suffix = "")

Write a container of jets to the specified container name (and optionally suffix). The container name and suffix should be declared beforehand using *AddFatJets()*. This clears the current branch state for the collection so it only makes sense to call once per call to `Fill()`.

Parameters

- **fatJets** – A container of jets to be written out.
- **fatjetName** – The name of the output collection to write to.
- **suffix** – The suffix of the output collection to write to.

void **FillFatJet**(const xAOD::Jet *fatjet_itr, int pvLocation = 0, const std::string &fatjetName = "fatjet", const std::string &suffix = "")

void **FillTruthFatJets**(const xAOD::JetContainer *truthFatJets, int pvLocation = 0, const std::string &truthFatJetName = "truth_fatjet") void **FillTruthFatJet**(const xAOD::Jet *truth_fatjet_itr, int pvLocation = 0, const std::string &truthFatJetName = "truth_fatjet")

void **FillTaus**(const xAOD::TauJetContainer *taus, const std::string &tauName = "tau")

void **FillTau**(const xAOD::TauJet *tau, const std::string &tauName = "tau")

void **FillMET**(const xAOD::MissingETContainer *met, const std::string &metName = "met")

void **Fill**()

void **ClearEvent**()

void **ClearTrigger**()

void **ClearJetTrigger**()

void **ClearMuons**(const std::string &jetName = "muon")

void **ClearElectrons**(const std::string &elecName = "el")

void **ClearPhotons**(const std::string &photonName = "ph")

void **ClearClusters**(const std::string &clusterName = "cl")

void **ClearJets**(const std::string &jetName = "jet")

void **ClearL1Jets**(const std::string &jetName = "L1Jet")

void **ClearTruth**(const std::string &truthName)

void **ClearTracks**(const std::string &trackName)

void **ClearFatJets**(const std::string &fatjetName, const std::string &suffix = "")

void **ClearTruthFatJets**(const std::string &truthFatJetName = "truth_fatjet")

void **ClearTaus**(const std::string &tauName = "tau")

void **ClearMET**(const std::string &metName = "met")

void **ClearVertices**(const std::string &vertexName = "vertex")

void **ClearTruthVertices**(const std::string &vertexName = "truth_vertex")

bool **writeTo**(TFile *file)

inline virtual void **AddEventUser**(const std::string &detailStr = "")

inline virtual void **AddTriggerUser**(const std::string &detailStr = "")

inline virtual void **AddJetTriggerUser**(const std::string &detailStr = "")

inline virtual void **AddMuonsUser**(const std::string &detailStr = "", const std::string &muonName = "muon")

inline virtual void **AddElectronsUser**(const std::string &detailStr = "", const std::string &elecName = "el")

inline virtual void **AddPhotonsUser**(const std::string &detailStr = "", const std::string &photonName = "ph")

inline virtual void **AddClustersUser**(const std::string &detailStr = "", const std::string &clusterName = "cl") inline virtual void **AddJetsUser**(const std::string &detailStr = "", const std::string &jetName = "jet")

inline virtual void **AddTruthUser**(const std::string &truthName = "", const std::string &detailStr = "xAH_truth")

inline virtual void **AddTracksUser**(const std::string &trackName = "", const std::string &detailStr = "trk")

inline virtual void **AddFatJetsUser**(const std::string &detailStr = "", const std::string &fatjetName = "", const std::string &suffix = "")

Declare a new fat jet collection. Automatically called once per call to `AddFatJets()`; override this if you want to provide your own additional branches for fatjets.

Parameters

- **detailStr** – The space-separated list of detail requested by the called.
- **fatjetName** – The (prefix) name of the output collection.
- **suffix** – A suffix to be appeneded to the end of the output branch name(s).

inline virtual void **AddTruthFatJetsUser**(const std::string &detailStr = "", const std::string &truthFatJetName = "truth_fatjet")

inline virtual void **AddTausUser**(const std::string &detailStr = "", const std::string &tauName = "tau")

inline virtual void **AddMETUser**(const std::string &detailStr = "", const std::string &metName = "met")

inline virtual void **ClearEventUser**()

inline virtual void **ClearTriggerUser**()

inline virtual void **ClearMuonsUser**(const std::string&)

inline virtual void **ClearElectronsUser**(const std::string&)

inline virtual void **ClearPhotonsUser**(const std::string&)

inline virtual void **ClearClustersUser**(const std::string&)

inline virtual void **ClearTruthUser**(const std::string&)

inline virtual void **ClearTracksUser**(const std::string&)

inline virtual void **ClearJetsUser**(const std::string&)

inline virtual void **ClearFatJetsUser**(const std::string&, const std::string&)

inline virtual void **ClearTruthFatJetsUser**(const std::string&)

inline virtual void **ClearTausUser**(const std::string&)

inline virtual void **ClearMETUser**(const std::string&)

inline virtual void **FillEventUser**(const xAOD::EventInfo*)

inline virtual void **FillMuonsUser**(const xAOD::Muon*, const std::string&, const xAOD::Vertex*)

inline virtual void **FillElectronsUser**(const xAOD::Electron*, const std::string&, const xAOD::Vertex*)

inline virtual void **FillPhotonsUser**(const xAOD::Photon*, const std::string&)

inline virtual void **FillClustersUser**(const xAOD::CaloCluster*, const std::string&)

inline virtual void **FillJetsUser**(const xAOD::Jet*, const std::string&)

inline virtual void **FillTruthUser**(const xAOD::TruthParticle*, const std::string&)

inline virtual void **FillTracksUser**(const xAOD::TrackParticle*, const std::string&)

inline virtual void **FillFatJetsUser**(const xAOD::Jet*, int, const std::string&, const std::string&)

Called once per call to *FillFatJets()*.Ooverride this if you want to any additional information to your jet collection.

Parameters

- **jet** – a pointer to the current xAOD::Jet object that should be written to the output branch(s).
- **fatjetName** – the (prefix) name of the output collection
- **suffix** – the suffix to append to output branches.

inline virtual void **FillTruthFatJetsUser**(const xAOD::Jet*, int, const std::string&)

inline virtual void **FillTausUser**(const xAOD::TauJet*, const std::string&)

inline virtual void **FillMETUser**(const xAOD::MissingETContainer*, const std::string&)

inline virtual void **FillTriggerUser**(const xAOD::EventInfo*)

inline virtual void **FillJetTriggerUser**()

Public Members

xAOD::TEvent ***m_event**

xAOD::TStore ***m_store**

std::string **m_vertexContainerName** = "PrimaryVertices"

Name of vertex container.

std::string **m_truthVertexContainerName** = "TruthVertices"

HelperClasses::*TriggerInfoSwitch* ***m_trigInfoSwitch**

std::string **m_triggerSelection**

TrigConf::xAODConfigTool ***m_trigConfTool**

Trig::TrigDecisionTool ***m_trigDecTool**

Public Static Functions

static std::string **FatJetCollectionName**(const std::string &fatjetName = "fatjet", const std::string &suffix = "")

Helper function to lookup each fatjet container name/suffix combo in the internal map of vectors for vectors. You probably don't need this but it might be useful if you're implementing `[Add/Fill/Clear]FatJetsUser()`.

Parameters

- **fatjetName** – The (prefix) name of the container.
- **suffix** – The container branch suffix.

Returns a string that uniquely identifies the collection name/suffix in the lookup map.

Protected Functions

template<typename **T**, typename **U**, typename **V**>
void **safeFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<*T*> &accessor, std::vector<*U*> &destination, *U* defaultValue, int m_units = 1)

template<typename **T**, typename **U**, typename **V**>
void **safeVecFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<std::vector<*T*>> &accessor, std::vector<std::vector<*U*>> &destination, int m_units = 1)

template<typename **T**>
void **setBranch**(std::string prefix, std::string varName, std::vector<*T*> *localVectorPtr)

Protected Attributes

TTree ***m_tree**

int **m_units**

bool **m_debug**

bool **m_isMC**

std::string **m_nominalTreeName**

bool **m_nominalTree**

xAH::*EventInfo* ***m_eventInfo**

int **m_passL1**

int **m_passHLT**

unsigned int **m_masterKey**

unsigned int **m_L1PSKey**

unsigned int **m_HLTPSKey**

std::vector<std::string> **m_elTrigForMatching**

std::vector<std::string> **m_passedTriggers**

std::vector<std::string> **m_disabledTriggers**

std::vector<float> **m_triggerPrescales**

std::vector<float> **m_triggerPrescalesLumi**

std::vector<std::string> **m_isPassBitsNames**

std::vector<unsigned int> **m_isPassBits**

std::map<std::string, xAH::*JetContainer**> **m_jets**

std::map<std::string, xAH::*L1JetContainer**> **m_l1Jets**

std::map<std::string, xAH::*TruthContainer**> **m_truth**

std::map<std::string, xAH::*TrackContainer**> **m_tracks**

std::map<std::string, xAH::*FatJetContainer**> **m_fatjets**

std::map<std::string, xAH::*FatJetContainer**> **m_truth_fatjets**

std::map<std::string, xAH::*MuonContainer**> **m_muons**

std::map<std::string, std::vector<std::string>> **m_MuonRecoEff_SF_sysNames**

std::map<std::string, std::vector<std::string>> **m_MuonIsoEff_SF_sysNames**

std::map<std::string, std::map<std::string, std::vector<std::string>>> **m_MuonTrigEff_SF_sysNames**

std::vector<std::string> **m_MuonTTVAEff_SF_sysNames** std::map<std::string, xAH::*ElectronContainer**> **m_elecs**

std::map<std::string, xAH::*PhotonContainer**> **m_photons**

std::map<std::string, xAH::*ClusterContainer**> **m_clusters**

std::map<std::string, xAH::*TauContainer**> **m_taus**

std::map<std::string, xAH::*MetContainer**> **m_met**

std::map<std::string, xAH::*VertexContainer**> **m_vertices**

std::map<std::string, xAH::*VertexContainer**> **m_truth_vertices**

Class HistogramManager

- Defined in file_xAODAnaHelpers_HistogramManager.h

Inheritance Relationships

Derived Type

- `public MetHists` (*Class MetHists*)

Class Documentation

class **HistogramManager**

This is used by any class extending to pre-define a set of histograms to book by default.

We expect the user to create a new group of histograms, such as for jets:

```
class JetHists : public HistogramManager
{
  public:
    JetHists(std::string name, std::string detailStr);
    virtual ~JetHists() ;

    bool m_debug;
    StatusCode initialize();
    StatusCode execute( const xAOD::JetContainer  jets, float eventWeight, int pvLoc = -1);
    StatusCode execute( const xAOD::Jet  jet, float eventWeight, int pvLoc = -1 );
    using HistogramManager::book; // make other overloaded version of book() to show up in subclass
```

(continues on next page)

(continued from previous page)

```
    using HistogramManager::execute; // overload
};
```

The above example is taken from our implementation in `JetHists`.

Note: The expectation is that the user does not directly use this class but rather inherits from it.

Subclassed by *MetHists*

Public Types

typedef std::unordered_map<std::string, TH1*> **HistMap_t**

Typedef for convenience.

Public Functions

HistogramManager(std::string name, std::string detailStr)

Initialization.

Parameters

- **name** – The top-level path in which all histograms are stored under (think of `TDirectory`)
- **detailStr** – Specify the various details of which to plot. For example, jets might want `"kinematic substructure"`.

virtual **~HistogramManager**()

Destructor, allows the user to delete histograms that are not being recorded.

inline virtual StatusCode **initialize**()

Initialize and book all histograms.

Example implementation:

```
StatusCode JetHists::initialize() {
  m_jetPt          = book(m_name, "jetPt",  "jet p_{T} [GeV]", 120, 0, 3000.);
  return StatusCode::SUCCESS;
}
```

Note: This should call the overloaded functions *HistogramManager::book()* to create the histograms so that the user can call *hists->record(wk())* to record all histograms to the EventLoop worker.

---

inline virtual StatusCode **execute**()

Execute by filling in the histograms.

Example implementation:

```
StatusCode JetHists::execute( const xAOD::JetContainer  jets, float eventWeight
↪){
  for(const auto& jet:  jets)
    m_jetPt->Fill( jet->pt()/1.e3, eventWeight );
  return StatusCode::SUCCESS;
}
```

inline virtual StatusCode **finalize**()

Finalize anything that needs to be finalized.

> **Warning:** This should rarely be used. There is not a good use case for this functionality but it needs to exist in the off-chance that a user comes along and needs it for their histogram class.

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh)

record a histogram and call various functions

> **Note:** This is an overloaded function. It will build the right histogram given the correct number of input arguments.

Parameters

- **name** – name of histogram, access it in ROOT file like `h_jetPt->Draw()`
- **title** – usually pointless,put a description of the histogram in here
- **xlabel** – label to put on the x-axis
- **xbins** – number of xbins to use
- **xlow** – lower bound on xbins
- **xhigh** – upper bound on xbins
- **xbinsArr** – variable xbins, test math (x_1, y_1) and (x_2, y_2)
- **ylabel** – label to put on the y-axis
- **ylow** – lower bound on ybins
- **yhigh** – upper bound on ybins
- **ybinsArr** – variable ybins
- **zlabel** – label to put on the z-axix
- **zlow** – lower bound on zbins
- **zhigh** – upper bound on zbins
- **zbinsArr** – variable zbins

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string xyabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, double ylow, double yhigh, std::string zlabel, int zbins, double zlow, double zhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr, std::string zlabel, int zbins, const Double_t *zbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, double ylow, double yhigh, std::string option = "")

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, const Double_t *xbinsArr, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, double xlow, double xhigh, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

void **record**(EL::IWorker *wk)

record all histograms from *HistogramManager::m_allHists* to the worker

MsgStream &**msg**() const

the standard message stream for this algorithm

MsgStream &**msg**(int level) const

allow ANA_MSG_XXXX macros to be used within algorithms for a given level

TH1 ***findHist**(const std::string &histName)

Return the pointer to the histogram.

void **fillHist**(const std::string &histName, double value)

Fill a histogram by name. Can be overloaded with weight.

Parameters

- **histName** – The name of the histogram to be filled
- **value** – The value to fill the histogram with

void **fillHist**(const std::string &histName, double value, double weight)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

void **fillHist**(const std::string &histName, double valueX, double valueY, double weight)

void **fillHist**(const std::string &histName, double valueX, double valueY, double valueZ, double weight)

void **fillProfile**(const std::string &histName, double valueX, double valueY, double weight)

Public Members

HistMap_t **m_histMap**

The map of histogram names to their pointers.

Protected Attributes

std::string **m_name**

generically the main name assigned to all histograms

std::string **m_detailStr**

a detail level in the form of a string

std::vector<TH1*> **m_allHists**

a container holding all generated histograms

mutable MsgStream **m_msg**

hold the MsgStream object

Class HLTJetGetter

- Defined in file_xAODAnaHelpers_HLTJetGetter.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **HLTJetGetter** : public xAH::*Algorithm*

Public Functions

HLTJetGetter()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_triggerList** = ".*"

List of triggers whose features will be extracted from TDT.

std::string **m_inContainerName** = ""

input container name, WITHOUT the HLT_xAOD__JetContainer_ prefix

std::string **m_outContainerName** = ""

output container name

Class IParticleHistsAlgo

- Defined in file_xAODAnaHelpers_IParticleHistsAlgo.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Derived Types

- `public ElectronHistsAlgo` (*Class ElectronHistsAlgo*)
- `public JetHistsAlgo` (*Class JetHistsAlgo*)
- `public MuonHistsAlgo` (*Class MuonHistsAlgo*)
- `public PhotonHistsAlgo` (*Class PhotonHistsAlgo*)

Class Documentation

class **IParticleHistsAlgo** : public xAH::*Algorithm*

Subclassed by *ElectronHistsAlgo*, *JetHistsAlgo*, *MuonHistsAlgo*, *PhotonHistsAlgo*

Public Functions

IParticleHistsAlgo(std::string className = "IParticleHistsAlgo")

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

Calls execute<IParticleContainer>

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

template<class **HIST_T**, class **CONT_T**>

inline EL::StatusCode **execute**()

Fill histograms with particles in a container.

Templated (container type) function that loops over all systematics (or nominal only)

and fills the corresponding histogram objects.

The event weight, in case of Monte Carlo samples, is mcEventWeight crossSection*filterEfficiency*kfactor

where the sample-weights are taken from SampleHandler and set to 1 by default.

virtual EL::StatusCode **AddHists**(std::string name)

Calls AddHists<IParticleHists>

Parameters **name** – Name of the systematic

template<class **HIST_T**>
inline EL::StatusCode **AddHists**(std::string name)

Create histograms.

Tempalated (histogram colllection class) function that creates all necessary histogram

objects for a given systematic. The class chosen for HIST_T template must inherit from IParticleHists.

Parameters **name** – Name of the systematic

Public Members

std::string **m_inContainerName** = ""

input container

std::string **m_detailStr** = ""

which plots will be turned on

std::string **m_inputAlgo** = ""

name of algo input container comes from - only if

std::string **m_histPrefix**

Histogram name prefix when using *IParticleHistsAlgo* directly

std::string **m_histTitle**

Histogram xaxis title when using *IParticleHistsAlgo* directly

Class IsoCloseByCorr

- Defined in file_xAODAnaHelpers_IsoCloseByCorr.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **IsoCloseByCorr** : public xAH::*Algorithm*

Public Functions

IsoCloseByCorr()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

ClassDef (IsoCloseByCorr, 1)

Public Members

bool **m_decorateSelectedObjects**

Decorate selected objects (the default decoration string is `passOR`)

std::string **m_decor** = "passOR"

std::string **m_inContainerName_Electrons** = ""

std::string **m_inputAlgoElectrons** = ""

std::string **m_el_iso_WP** = ""

std::string **m_inContainerName_Muons** = ""

std::string **m_outContainerName_Muons** = ""

std::string **m_mu_iso_WP** = ""

bool **m_doPhotons** = false

std::string **m_inContainerName_Photons** = ""

std::string **m_outContainerName_Photons** = ""

Protected Attributes

int **dummyVaraibleToKeepExample**

Class JetCalibrator

- Defined in file_xAODAnaHelpers_JetCalibrator.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **JetCalibrator** : public xAH::*Algorithm*

A wrapper to a few JetETMiss packages. By setting the configuration parameters detailed in the header documentation, one can:

- calibrate a given jet collection
- apply systematic variations for JES
- apply systematic variations for JER
- decorate the jet with the decision of the Jet Cleaning tool

When considering systematics, a new `xAOD::JetCollection` is created for each systematic variation. The names are then saved in a vector for downstream algorithms to use.

Public Functions

JetCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

The name of the input container for this algorithm to read from `TEvent` or `TStore`

std::string **m_outContainerName** = ""

The name of the nominal output container written by the algorithm to `TStore`

If the algorithm applies systematic variations, for each shallow copy saved to `TStore`, the systematic name will be appended to this.

std::string **m_jetAlgo** = ""

set to `AntiKt4EMTopo` for `AntiKt4EMTopoJets`

std::string **m_outputAlgo** = ""

name of vector holding names of jet systematics given by the JetEtmiss Tools

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

bool **m_recalibrateHLTJets** = false

whether to run HLT jet re-calibration

std::string **m_HLTVertexContainerName** = "HLT_IDVertex_FS"

vertex container name to use for HLT jet re-calibration

std::string **m_HLTAvgMuDecor** = "EventInfo.AvgMu"

HLT average mu decoration on EventInfo after formatting.

std::string **m_EvtInfoHLTNPVDecor** = ""

location of the HLT NPV on EventInfo object (e.g. EventInfo.NPV) this defaults to an empty string and is only configured in JetCalibrationTool when a non-empty string is provided

std::string **m_calibGSCDepth** = ""

GSCDepth property to override GSCDepth in config file when set to a non-empty string and GSC is in the calibration sequence.

std::string **m_calibConfigDir** = ""

config for JetCalibrationTool ConfigDir, set it to override tool defaults

std::string **m_calibConfigData** = "JES_data2017_2016_2015_Recommendation_Aug2018_rel21.config"

config for JetCalibrationTool for Data

std::string **m_calibConfigFullSim** = "JES_data2017_2016_2015_Recommendation_Aug2018_rel21.config"

config for JetCalibrationTool for Full Sim MC

std::string **m_calibConfigAFII** = "JES_MC16Recommendation_AFII_EMTopo_April2018_rel21.config"

config for JetCalibrationTool for AFII MC

std::string **m_calibSequence** = ""

List of calibration steps. Auto-configured to the Jet/Etmiss recommendation if left blank.

std::string **m_uncertConfig** = ""

config for Jet Uncertainty Tool

std::string **m_uncertMCType** = ""

MC type for Jet Uncertainty Tool (need to be set for FullSim)

std::string **m_overrideCalibArea** = ""

Override CalibArea tag (default recommended)

std::string **m_overrideUncertCalibArea** = ""

Override uncertainties CalibArea tag (default recommended)

std::string **m_overrideAnalysisFile** = ""

Set analysis-specific jet flavour composition file for JetUncertainties (default: unknown comp.)

std::string **m_overrideUncertPath** = ""

Override uncertainties path (not recommended)

bool **m_forceInsitu** = false

when running data "_Insitu" is appended to calibration sequence

bool **m_forceSmear** = false

when running FullSim "_Smear" is appended to calibration sequence

bool **m_jetCalibToolsDEV** = false
 when using DEV mode of JetCalibTools

bool **m_addGhostMuonsToJets** = false
 Run muon-to-jet ghost association (recommended for MET)

bool **m_doCleaning** = true
 enable to apply jet cleaning decoration

std::string **m_jetCleanCutLevel** = "LooseBad"
 Cut Level.

bool **m_saveAllCleanDecisions** = false
 Save all cleaning decisions as decorators.

bool **m_jetCleanUgly** = false
 Do Ugly cleaning (i.e. TileGap 3)

bool **m_sort** = true
 Sort the processed container elements by transverse momentum.

bool **m_cleanParent** = false
 Apply jet cleaning to parent jet.

bool **m_applyFatJetPreSel** = false

bool **m_useLargeRTruthLabelingTool** = true
 Use large-R jet truth labeling tool (needed for systematics)

std::string **m_truthLabelName** = "R10TruthLabel_R21Consolidated"
 Name of the large-R jet truth labeling definition.

bool **m_isTruthJetCol** = false
 Flag to indicate if using a truth jet collection.

bool **m_useTRUTH3** = true
 Flag to indicate if input xAOD uses TRUTH3 style containers.

std::string **m_truthParticleContainerName** = "TruthParticles"
 Name of the truth particle container if not using TRUTH3 containers.

std::string **m_truthBosonContainerName** = "TruthBosonsWithDecayParticles"
 Name of the truth boson container if using TRUTH3 containers.

std::string **m_truthTopQuarkContainerName** = "TruthTopQuarkWithDecayParticles"

Name of the truth top quark container if using TRUTH3 containers.

bool **m_doJetTileCorr** = false

jet tile correction

bool **m_pseudoData** = false

needed in case want to treat MC as pseudoData for JER uncertainty propagation

bool **m_mcAndPseudoData** = false

Treat MC as usual, then run the JER uncertainties on it a second time treating it as pseudodata. Overrides m_pseudodata if true.

Class JetHistsAlgo

- Defined in file_xAODAnaHelpers_JetHistsAlgo.h

Inheritance Relationships

Base Type

- `public IParticleHistsAlgo` (*Class IParticleHistsAlgo*)

Class Documentation

class **JetHistsAlgo** : public *IParticleHistsAlgo*

Public Functions

JetHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **execute**()

Calls execute<IParticleContainer>

virtual EL::StatusCode **AddHists**(std::string name)

Calls AddHists<IParticleHists>

Parameters **name** – Name of the systematic

Class JetSelector

- Defined in file_xAODAnaHelpers_JetSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **JetSelector** : public xAH::*Algorithm*

Public Functions

JetSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual bool **executeSelection**(const xAOD::JetContainer *inJets, float mcEvtWeight, bool count, std::string outContainerName, bool isNominal)

virtual int **PassCuts**(const xAOD::Jet *jet)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_truthJetContainer** = "AntiKt4TruthJets"

truth jet container name (used for JVT SF)

std::string **m_inputAlgo** = ""

input type - from xAOD or from xAODAnaHelper Algo output

std::string **m_outputAlgo** = ""

output type - this is how the vector<string> w/ syst names will be saved in TStore

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

std::string **m_jetScaleType** = ""

Type of Scale Momementum.

std::string **m_decor** = "passSel"

The decoration key written to passing objects.

bool **m_decorateSelectedObjects** = true

decorate selected objects? defaul passSel

bool **m_createSelectedContainer** = false

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1

look at n objects

bool **m_cleanJets** = true

require cleanJet decoration to not be set and false

int **m_cleanEvtLeadJets** = -1

kill event if any of the N leading jets are not clean

bool **m_cleanEvent** = false

Kill event if any passing jets are not clean.

Note: The jets need the *cleanJet* decoration which is set when you enable `JetCalibrator::m_doCleaning`

bool **m_markCleanEvent** = false

Mark event with decorator if any passing jets are not clean.

std::string **m_jetScale4Selection** = "Final"

Choose the scale at which the selection is performed (default "Final", i.e. default 4vector)

bool **m_doMCCleaning** = false

(MC-only) Kill pileup overlay event if reconstructed jets avg(pT1,pT2) > 1.4*(truth jet pT1)

float **m_mcCleaningCut** = 1.4

Change the default 1.4 cut to x > 1.0.

int **m_pass_min** = -1

minimum number of objects passing cuts

int **m_pass_max** = -1

maximum number of objects passing cuts

float **m_pT_max** = 1e8

require pT < pt_max

float **m_pT_min** = 1e8

require pT > pt_min

float **m_ET_max** = 1e8

require ET < ET_max

float **m_ET_min** = 1e8

require ET > ET_min

float **m_eta_max** = 1e8

require eta < eta_max

float **m_eta_min** = 1e8

require eta > eta_min

float **m_detEta_max** = 1e8

require detEta < detEta_max

float **m_detEta_min** = 1e8

require detEta > detEta_min

float **m_mass_max** = 1e8

require mass < mass_max

float **m_mass_min** = 1e8

require mass > mass_min

float **m_rapidity_max** = 1e8

require rapidity < rapidity_max

float **m_rapidity_min** = 1e8

require rapidity > rapidity_min

int **m_truthLabel** = -1

require truth level on truth jets

bool **m_useHadronConeExcl** = true

use HadronConeExclTruthLabelID for truth match (default)

bool **m_doJVF** = false

check JVF

float **m_pt_max_JVF** = 50e3

max pT [GeV] (JVF is a pileup cut)

float **m_eta_max_JVF** = 2.4

detector eta cut

float **m_JVFCut** = 0.5

cut value

bool **m_doJVT** = false

check JVT

bool **m_noJVTVeto** = false

keep JVT-rejected jets and decorate passing status

bool **m_dofJVT** = false

check forward JVT

bool **m_dofJVTVeto** = true

Remove jets that fail fJVT. Like JVT, the default is to clean the collection.

float **m_pt_max_JVT** = 60e3

max pT [GeV] (JVT is a pileup cut)

float **m_eta_max_JVT** = 2.4

detector eta cut

bool **m_jvtUsedBefore** = false

was JVT already run in an earlier instance of *JetSelector*?

bool **m_haveTruthJets** = true

Does the input have truth jets? If not, cannot decorate with true hard scatter / pileup info.

bool **m_getJVTSF** = true

Retrieve JVT SFs (true by default, when false: allows to get JVT decision w/o needing truth jets)

float **m_JVTCut** = -1.0

Minimum value of JVT for selecting jets.

> **Warning:** If set to a non-negative value (default is -1.0), it will override any set value for *JetSelector::m_WorkingPointJVT*

std::string **m_WorkingPointJVT** = "FixedEffPt"

Available working points for JVT cut from the `CP::IJetJvtEfficiency` tool.

The corresponding data/MC SF will be saved as a `std::vector<float>` decoration (for MC only), for nominal WP and the available systematics.

Value	JVT Cut	Efficiency
"Medium"	(Default) 0.59	92%
"Loose"	0.11	97%
"Tight"	0.91	85%

std::string **m_SFFileJVT** = "DummySFs.root"

Configuration containting JVT scale factors.

The configuration file with the scale factors calculated by the `CP::IJetJvtEfficiency`.

See :https://twiki.cern.ch/twiki/bin/view/AtlasProtected/JVTCalibration for latest recommendation.

std::string **m_outputSystNamesJVT** = "JetJvtEfficiency_JVTSyst"

int **m_JvtTaggingAlg** = CP::JvtTagger::NNJvt

Tagging algorithm to be used to veto PU jets in central region - default in R22 is NNJvt. If another algorithm is needed, use corresponding index for the enum here: https://acode-browser1.usatlas.bnl.gov/lxr/source/athena/PhysicsAnalysis/Interfaces/JetAnalysisInterfaces/JetAnalysisInterfaces/IJetJvtEfficiency.h#0022 (note: this link points to the latest r22 version, i.e. master, if a release is used, please check the corresponding enum for the given release: https://gitlab.cern.ch/atlas/athena/-/tags?search=release%2F22.2&sort=updated_desc)

bool **m_recalculateJvtScores** = true

Do re-calculation of NNJvt - scores need to be re-evaluated in case jet pt changed w.r.t. derivation.

float **m_systValJVT** = 0.0

std::string **m_systNameJVT** = ""

std::string **m_WorkingPointfJVT** = "Loose"

Available working points for fJVT cut from the `CP::IJetJvtEfficiency` tool.

The corresponding data/MC SF will be saved as a `std::vector<float>` decoration (for MC only), for nominal WP and the available systematics.

Value	HS Efficiency	PU Fake Rate
"Medium"	87.1-97.0%	53.4-60.9%
"Tight"	79.9-95.6%	45.4-50.3%

See :https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/FJVTCalibration for more information.

std::string **m_SFFilefJVT** = ""

Configuration containting fJVT scale factors.

The configuration file with the scale factors calculated by the `CP::IJetJvtEfficiency`.

See :https://twiki.cern.ch/twiki/bin/view/AtlasProtected/FJVTCalibration for latest recommendation.

std::string **m_outputSystNamesfJVT** = "JetJvtEfficiency_fJVTSyst"

float **m_systValfJVT** = 0.0

std::string **m_systNamefJVT** = ""

bool **m_fjvtUsedBefore** = false

was fJVT already run in an earlier instance of *JetSelector*?

bool **m_doJetTimingCut** = false

Timing cut.

float **m_jetTiming_max** = -1

bool **m_doBTagCut** = false

Flag to apply btagging cut, if false just decorate decisions.

std::string **m_corrFileName** = "xAODBTaggingEfficiency/cutprofiles_22072015.root"

std::string **m_jetAuthor** = "AntiKt4EMPFlowJets"

std::string **m_taggerName** = "DL1r"

std::string **m_operatingPt** = "FixedCutBEff_70" double **m_b_eta_max** = 2.5

double **m_b_pt_min** = 20e3

bool **m_doHLTBTagCut** = false

std::string **m_HLTBTagTaggerName** = "DL1r"

float **m_HLTBTagCutValue** = -0.4434

bool **m_requireHLTVtx** = false

bool **m_requireNoHLTVtx** = false

std::string **m_passAuxDecorKeys** = ""

std::string **m_failAuxDecorKeys** = ""

std::string **m_singleJetTrigChains** = ""

A comma-separated string w/ alll the HLT single jet trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

std::string **m_diJetTrigChains** = ""

A comma-separated string w/ all the HLT dijet trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

bool **m_removeDuplicates** = false

remove duplicate jets (exactly the same eta)

int **m_count_events_with_duplicates** = 0

number of events with duplicates

bool **m_sort** = false

sort jets (normally done by *JetCalibrator*, but HLT jets need sorting and don't get calibrated here)

Class MessagePrinterAlgo

- Defined in file_xAODAnaHelpers_MessagePrinterAlgo.h

Inheritance Relationships

Base Type

- public xAH::Algorithm (*Class Algorithm*)

Class Documentation

class **MessagePrinterAlgo** : public xAH::*Algorithm*

This algorithm changes the format of the MsgStream objects for all other algorithms. There should only be on instance of it, and it should probably be first.

Public Functions

MessagePrinterAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

unsigned int **m_sourceWidth** = 25

Set the width of the name in the message.

Class METConstructor

- Defined in file_xAODAnaHelpers_METConstructor.h

Inheritance Relationships

Base Type

- public xAH::Algorithm (*Class Algorithm*)

Class Documentation

class **METConstructor** : public xAH::*Algorithm*

Public Functions

METConstructor()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_mapName** = "METAssoc_AntiKt4LCTopo"

std::string **m_coreName** = "MET_Core_AntiKt4LCTopo"

std::string **m_outputContainer** = "NewRefFinal"

std::string **m_systConfigPrefix** = "METUtilities/R22_PreRecs"

std::string **m_systConfigSoftTrkFile** = "TrackSoftTerms-pflow.config"

std::string **m_inputJets** = ""

std::string **m_inputElectrons** = ""

std::string **m_inputPhotons** = ""

std::string **m_inputTaus** = ""

std::string **m_inputMuons** = ""

bool **m_doElectronCuts** = false

bool **m_doPhotonCuts** = false

bool **m_doTauCuts** = false

bool **m_doMuonCuts** = false

bool **m_doMuonEloss** = false

bool **m_doIsolMuonEloss** = false

bool **m_doJVTCut** = false

bool **m_dofJVTCut** = false

std::string **m_fJVTdecorName** = "passFJVT"

Name of fJVT decoration.

bool **m_doPFlow** = true

To turn on p-flow MET calculation set m_doPFlow to true.

std::string **m_METWorkingPoint** = ""

Name of MET Working Point (defines the JetSelection applied in METMaker)

bool **m_rebuildUsingTracksInJets** = false

Rebuild MET using tracks in calo jets.

bool **m_addSoftClusterTerms** = false

Include soft cluster terms if rebuilding MET using jet terms (only considered if *m_rebuildUsingTracksInJets* is false)

bool **m_calculateSignificance** = false

Enable MET significance calculation.

bool **m_significanceTreatPUJets** = true

Introduce "resolution" for jets with low JVT, if the analysis is sensitive to pileup jets.

double **m_significanceSoftTermReso** = 10.0

Set soft term resolution.

bool **m_runNominal** = true

set to false if you want to run met systematics

std::string **m_systName** = "All"

do not change it, not useful

float **m_systVal** = 1.0

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

std::string **m_jetSystematics** = ""

Name of jet systematics vector from *JetCalibrator*.

std::string **m_eleSystematics** = ""

Name of electron systematics vector from *ElectronCalibrator*.

std::string **m_muonSystematics** = ""

Name of muon systematics vector from *MuonCalibrator*.

std::string **m_tauSystematics** = ""

Name of tau systematics vector from *TauCalibrator*.

std::string **m_phoSystematics** = ""

Name of photon systematics vector from *PhotonCalibrator*.

std::string **m_outputAlgoSystNames** = ""

Class MetHists

- Defined in file_xAODAnaHelpers_MetHists.h

Inheritance Relationships

Base Type

- `public HistogramManager` (*Class HistogramManager*)

Class Documentation

class **MetHists** : public *HistogramManager*

Public Functions

MetHists(std::string name, std::string detailStr)

virtual **~MetHists**()

virtual StatusCode **initialize**()

Initialize and book all histograms.

Example implementation:

```
StatusCode JetHists::initialize() {
  m_jetPt          = book(m_name, "jetPt",  "jet p_{T} [GeV]", 120, 0, 3000.);
  return StatusCode::SUCCESS;
}
```

Note: This should call the overloaded functions `HistogramManager::book()` to create the histograms so that the user can call *hists->record(wk())* to record all histograms to the EventLoop worker.

StatusCode **execute**(const xAOD::MissingETContainer *met, float eventWeight)

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh)

record a histogram and call various functions

Note: This is an overloaded function. It will build the right histogram given the correct number of input arguments.

Parameters

- **name** – name of histogram, access it in ROOT file like `h_jetPt->Draw()`
- **title** – usually pointless,put a description of the histogram in here
- **xlabel** – label to put on the x-axis
- **xbins** – number of xbins to use
- **xlow** – lower bound on xbins
- **xhigh** – upper bound on xbins
- **xbinsArr** – variable xbins, test math (x_1, y_1) and (x_2, y_2)
- **ylabel** – label to put on the y-axis
- **ylow** – lower bound on ybins
- **yhigh** – upper bound on ybins
- **ybinsArr** – variable ybins

- **`zlabel`** – label to put on the z-axix
- **`zlow`** – lower bound on zbins
- **`zhigh`** – upper bound on zbins
- **`zbinsArr`** – variable zbins

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string xyabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, double ylow, double yhigh, std::string zlabel, int zbins, double zlow, double zhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH1F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, double xlow, double xhigh, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH2F ***book**(std::string name, std::string title, std::string xyabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TH3F ***book**(std::string name, std::string title, std::string xlabel, int xbins, const Double_t *xbinsArr, std::string ylabel, int ybins, const Double_t *ybinsArr, std::string zlabel, int zbins, const Double_t *zbinsArr)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, std::string xlabel, int xbins, double xlow, double xhigh, std::string ylabel, double ylow, double yhigh, std::string option = "")

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, const Double_t *xbinsArr, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

TProfile ***book**(std::string name, std::string title, int xbins, double xlow, double xhigh, double ylow, double yhigh)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

inline virtual StatusCode **execute**()

Execute by filling in the histograms.

Example implementation:

```
StatusCode JetHists::execute( const xAOD::JetContainer  jets, float eventWeight␣
↪){
  for(const auto& jet:  jets)
    m_jetPt->Fill( jet->pt()/1.e3, eventWeight );
  return StatusCode::SUCCESS;
}
```

Public Members

bool **m_debug**

Protected Attributes

HelperClasses::*METInfoSwitch* ***m_infoSwitch**

Class MetHistsAlgo

- Defined in file_xAODAnaHelpers_MetHistsAlgo.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **MetHistsAlgo** : public xAH::*Algorithm*

Public Functions

MetHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_detailStr** = ""

Class MinixAOD

- Defined in file_xAODAnaHelpers_MinixAOD.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **MinixAOD** : public xAH::*Algorithm*

Produce xAOD outputs.

I can think up the following cases when a user is doing an EL Algorithm:

> input containers in TEvent (simple) deep-copied containers in TStore (deep-copy) shallow-copied containers in TStore (shallow) CDV containers in TStore (cdv)

For the above use-cases, we might produce outputs like so:

write the input container to the output. This uses `TEvent::copy()`. write the deep-copied containers to the output. This calls `TStore::retrieve()` and then `TEvent::record()`. two options when we have shallow-copies:

1. `shallowIO=false`: write to the output as a deep-copy like in the previous option
2. `shallowIO=true`: write to the output as a shallow-copy, but make sure the original container is also written to the output

make a deep-copy of the ConstDataVector and then move from `TStore` to `TEvent`. The problem is that we point to local memory that will not persist when making the CDV.

The trickiest case is with shallow copies because those could be our systematics – and you might want to copy the original container, and only copy over systematics via true shallow copies to conserve memory and space.

> **Warning:** Care must be taken when managing memory and using copies. You need to think about how copies point to each other and whether you can use shallow copies or deep copies or both.

Public Functions

MinixAOD()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_outputFileName** = "out_miniXAOD"

name of the output file to use for xAOD dumping

bool **m_createOutputFile** = true

enable to create the output file for xAOD dumping

bool **m_copyFileMetaData** = false

copy the file metadata over

bool **m_copyTriggerInfo** = false

copy the trigger containers and meta data over

bool **m_copyCutBookkeeper** = false

copy the cutbookkeeper data over

std::string **m_simpleCopyKeys** = ""

names of containers to copy from the input file

Container names should be space-delimited:

```
"m_simpleCopyKeys": "EventInfo AntiKt4EMTopoJets"
```

std::string **m_storeCopyKeys** = ""

names of containers in the TStore to copy over

Container names should be space-delimited:

```
"m_storeCopyKeys": "BrandNewJetContainer ReclusteredJets"
```

Note: This option is appropriate for deep-copied containers.

std::string **m_shallowCopyKeys** = ""

names of containers that have been shallow-copied

This option is a little different because shallow-copied containers have parent containers. However, there are two options depending on the `setShallowIO` option

True If this is set to true, you will want to specify the parent container so that we copy it over as well (it is assumed that the parent container is in TStore or TEvent):

```
"m_shallowCopyKeys": "SCAntiKt4EMTopoJets|AntiKt4EMTopoJets␣
↪SCMuons|Muons_Presel"
```

False If this is set to false, you will not want to specify the parent container

”m_shallowCopyKeys”: “SCAntiKt4EMTopoJets| SCMuons|”

Always specify your string in a space-delimited format where pairs are split up by `shallow container name|parent container name`.

Note: This option is appropriate for shallow-copied containers.

Warning: Please note that the `shallowIO` option is what determines how the memory is managed. If you run into issues with shallow-copied containers here, make sure you know whether this option was enabled or not before asking for help.

std::string **`m_deepCopyKeys`** = ""

names of containers that have been shallow-copied

Here, we will do the deep-copying for you, so that the containers can be correctly recorded into the output. Due to the way view-only containers work, we can't figure out whether the memory points to a specific parent container we can copy, or to a non-persistable, local (stack) memory. The best option is to just deep-copy and allocate new memory instead:

```
"m_deepCopyKeys": "AntiKt4EMTopoJets|DeepCopyAntiKt4Jets Muons|DeepCopyMuons"
```

Always specify your string in a space-delimited format where pairs are split up by `input container name|output container name`.

Note: This option is appropriate for view-only containers such as `ConstDataVector`.

std::string **`m_vectorCopyKeys`** = ""

names of vectors that have container names for its contents

Here, we will do the copying for you by retrieving the vector of container names and copy each one over. See how *`MinixAOD::m_shallowCopyKeys`* works.

Always specify your string in a space-delimited format where pairs are split up by `vector name|parent container name`.

Note: This option is appropriate for groups shallow-copied containers such as when you are dealing with systematics.

Class MuonCalibrator

- Defined in file_xAODAnaHelpers_MuonCalibrator.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **MuonCalibrator** : public xAH::*Algorithm*

Public Functions

MuonCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName** = ""

std::string **m_calibrationMode** = "noOption"

Set calibrationMode property if different than noOption.

bool **m_isRun3Geo** = false

Switch on Run3 geometry for muon selector tool.

bool **m_do2StationsHighPt** = false

bool **m_sort** = true

std::string **m_inputAlgoSystNames** = ""

this is the name of the vector of names of the systematically varied containers produced by the upstream algo (e.g., the SC containers with calibration systematics)

std::string **m_outputAlgoSystNames** = "MuonCalibrator_Syst"

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systVal** = 0.0

std::string **m_systName** = ""

bool **m_forceDataCalib** = false

Force `MuonCalibrationPeriodTool.h` to calibrate data.

`MuonSelectorTool` depends on a specific decoration existing on Muons, namely `MuonSpectrometerPt`. This is decorated by the `MuonCalibrationAndSmearingTool`. However, you do not calibrate data by default so this tool would not be run on data.

In the case where you need the tool to be forced to run on data in order to have this decoration on your muons, you need to flip this boolean. See the Muon Combined Performance Working Group twiki for more information.

Note: This should not modify the momentum of muons in data (according to the tool as of `MuonMomentumCorrections-01-00-37`).

Class MuonEfficiencyCorrector

- Defined in file_xAODAnaHelpers_MuonEfficiencyCorrector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **MuonEfficiencyCorrector** : public xAH::*Algorithm*

Public Functions

MuonEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **executeSF**(const xAOD::EventInfo *eventInfo, const xAOD::MuonContainer *inputMuons, bool nominal, bool writeSystNames)

Public Members

std::string **m_inContainerName** = ""

std::string **m_overrideCalibRelease** = ""

Recommendations release (not recommended to change)

std::string **m_WorkingPointReco** = "Loose"

std::string **m_WorkingPointIso** = "LooseTrackOnly"

bool **m_AllowZeroSF** = false

Use with caution!!!

std::string **m_MuTrigLegs** = "HLT_mu26_imedium"

list of comma-separated single-mu trigger corrections. Individual legs of di-mu menus can be parsed

bool **m_usePerMuonTriggerSFs** = true

Get per-muon trigger SF (default: true) [if false it will take into account combinatorics using all muons from the input muon container].

std::string **m_WorkingPointTTVA** = "TTVA"

std::string **m_inputSystNamesMuons** = ""

The name of the vector containing the names of the systematically-varied muons-related containers from the upstream algorithm, which will be processed by this algorithm.

Only muon calibration systematics or any other that create shallow copies of electron containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systValReco** = 0.0

float **m_systValIso** = 0.0

float **m_systValTrig** = 0.0

float **m_systValTTVA** = 0.0

std::string **m_systNameReco** = ""

std::string **m_systNameIso** = ""

std::string **m_systNameTrig** = ""

std::string **m_systNameTTVA** = ""

std::string **m_outputSystNamesReco** = "MuonEfficiencyCorrector_RecoSyst"

std::string **m_outputSystNamesIso** = "MuonEfficiencyCorrector_IsoSyst"

std::string **m_outputSystNamesTrig** = "MuonEfficiencyCorrector_TrigSyst"

std::string **m_outputSystNamesTTVA** = "MuonEfficiencyCorrector_TTVASyst"

Class MuonHistsAlgo

- Defined in file_xAODAnaHelpers_MuonHistsAlgo.h

Inheritance Relationships

Base Type

- `public IParticleHistsAlgo` (*Class IParticleHistsAlgo*)

Class Documentation

class **MuonHistsAlgo** : public *IParticleHistsAlgo*

Public Functions

MuonHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **execute**()

Calls execute<IParticleContainer>

virtual EL::StatusCode **AddHists**(std::string name)

Calls AddHists<IParticleHists>

Parameters `name` – Name of the systematic

Class MuonInFatJetCorrector

- Defined in file_xAODAnaHelpers_MuonInFatJetCorrector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **MuonInFatJetCorrector** : public xAH::*Algorithm*

Algorithm for correcting the momentum of largeR jets containing muon decays.

Only muons associated to track jets are used. Quality and kinematic cuts for the muons and track jets can be adjusted by the user.

There are currently four correction schemes; Calorimeter, TrackAssisted, Combined, and SimpleMuon. At present, Combined is used, which takes a weighted sum of corrections from both the TrackAssisted and Calorimeter Schemes.

The corrected large-R are saved as a TLorentzVector in a decorator named "correctedFatJets_tlv".

Public Types

enum **Scheme**

Different schemes for the muon in jet correction.

Values:

enumerator **Calorimeter**

enumerator **TrackAssisted**

enumerator **Combined**

enumerator **SimpleMuon**

Public Functions

MuonInFatJetCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

EL::StatusCode **matchTrackJetsToMuons**() const

TLorentzVector **getHbbCorrectedVector**(const xAOD::Jet &jet)

const xAOD::JetFourMom_t **getMuonCorrectedJetFourMom**(const xAOD::Jet &jet, std::vector<const xAOD::Muon*> muons, Scheme scheme, bool useJMSScale = false) const

Public Members

std::string **m_fatJetContainerName** = ""

The name of the container with fat jets to be corrected.

std::string **m_trackJetContainerName** = "AntiKtVR30Rmax4Rmin02TrackJets"

The name of the container with track jets used for matching.

std::string **m_muonContainerName** = ""

The name of the container with muons to be used for the correction.

std::string **m_trackJetLinkName** = "GhostVR30Rmax4Rmin02TrackJet"

The name of the link to matched track jets.

std::string **m_calibratedMassDecoratorData** = "JetInsituScaleMomentum"

Name of calibrated jet mass decorator, without the TA/Calo suffix, for data.

std::string **m_calibratedMassDecoratorFullSim** = "JetJMSScaleMomentum"

Name of calibrated jet mass decorator, without the TA/Calo suffix, for full sim.

std::string **m_inputAlgo**

Algortihm systematics loop.

float **m_trackJetPtMin** = 10000.0

Minimum pt of track jets to use for correction.

float **m_trackJetEtaMax** = 2.5

Maximum eta of track jets to use for correction.

float **m_trackJetNConst** = 2.0

Minimum number of constituents (tracks) of track jets to use for correction.

float **m_muonPtMin** = 10000.0

Minimum pt of muons to use for correction.

float **m_muonEtaMax** = 2.7

Maximum eta of muons to use for correction.

float **m_muonDrMax** = 0.4

DR cut to use when matching muons to track jets.

Class MuonSelector

- Defined in file_xAODAnaHelpers_MuonSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **MuonSelector** : public xAH::*Algorithm*

Public Functions

MuonSelector()

~MuonSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::MuonContainer *inMuons, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::MuonContainer> *selectedMuons)

virtual int **passCuts**(const xAOD::Muon *muon, const xAOD::Vertex *primaryVertex)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""
  input container name

std::string **m_outContainerName** = ""
  output container name

std::string **m_outAuxContainerName**
  output auxiliary container name

std::string **m_inputAlgoSystNames** = ""

std::string **m_outputAlgoSystNames** = "MuonSelector_Syst"

bool **m_decorateSelectedObjects** = true
  decorate selected objects - default "passSel"

bool **m_createSelectedContainer** = false
  fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1

look at n objects

int **m_pass_min** = -1

minimum number of objects passing cuts

int **m_pass_max** = -1

maximum number of objects passing cuts

float **m_pT_max** = 1e8

require pT < pt_max

float **m_pT_min** = 1e8

require pT > pt_min

bool **m_pT_NaNcheck** = false

check if pT is NaN

std::string **m_muonQualityStr** = "Medium"

require quality

bool **m_isRun3Geo** = false

Switch on Run3 geometry for muon selector tool.

float **m_eta_max** = 1e8

require type require |eta| < eta_max

float **m_d0_max** = 1e8

require d0 < m_d0_max

float **m_d0sig_max** = 1e8

require d0 significance (at BL) < m_d0sig_max

float **m_z0sintheta_max** = 1e8

require z0*sin(theta) (at BL - corrected with vertex info) < m_z0sintheta_max

bool **m_removeCosmicMuon** = false

Remove cosmic muons that fail absolute z0 and d0 selections.

bool **m_removeEventBadMuon** = true

Remove events with a bad muon, defined by poor q/p.

bool **m_doIsolation** = true

enable or disable isolation

std::string **m_MinIsoWPCut** = ""

reject objects which do not pass this isolation cut - default = “” (no cut)

std::string **m_IsoWPList** = "FCTightTrackOnly_FixedRad,FCLoose_FixedRad,FCTight_FixedRad,FixedCutPflowTight,FixedCutPflowLoose"

decorate objects with ‘isIsolated_*’ flag for each WP in this input list - default = all current ASG WPs

std::string **m_CaloIsoEff** = "0.1*x+90"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_TrackIsoEff** = "98"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_CaloBasedIsoType** = "topoetcone20"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_TrackBasedIsoType** = "ptvarcone30"

to define a custom WP - make sure “UserDefined” is added in the above input list!

std::string **m_singleMuTrigChains** = ""

A comma-separated string w/ alll the HLT single muon trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

std::string **m_diMuTrigChains** = ""

A comma-separated string w/ all the HLT dimuon trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

double **m_minDeltaR** = 0.1

Recommended threshold for muon triggers: see https://svnweb.cern.ch/trac/atlasoff/browser/Trigger/TrigAnalysis/TriggerMatchingTool/trunk/src/TestMatchingToolAlg.cxx.

bool **m_merged_muons** = false

Element links need to be updated if merged muons are used (LRT + std) / false by default.

std::string **m_trigInputPrefix** = ""

Input prefix of trigger decision tool.

bool **m_doLRT** = false

add LRT muon information

std::string **m_isoDecSuffix** = ""

Class OverlapRemover

- Defined in file_xAODAnaHelpers_OverlapRemover.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **OverlapRemover** : public xAH::*Algorithm*

A wrapper of the overlap removal tool in the ASG AssociationUtils package.

The logic of the OLR belongs to the ASG tool itself, and is described extensively in the Analysis Harmonisation Task Force note.

If you wish to apply a custom OLR scheme, please contact the author marco.milesi@cern.ch for detailed instructions.

The idea behind this algorithm is to consistently thread together the inputs from upstream xAODAnaHelpers algorithms based on user's configuration, handling also the case where systematics on the input physics objects are taken into account. Here follows a usage example.

Consider the simplified scenario where we care only about jets** and **electrons**. Assuming the typical xAODAnaHelpers analysis configuration through `xAH_config`, the analysis workflow could look like the following:

```
c = xAH_config()
# ...
c.algorithm("JetSelector", JetSelectorDict)
c.algorithm("ElectronSelector", ElectronSelectorDict)
# ...
c.algorithm("OverlapRemover", OverlapRemoverDict)
# ...
```

where each algorithm has the following I/O systematics configuration (via python dictionaries):

```
JetSelectorDict = {
    # ...
    "m_inputAlgo" : "JetCalibrator_Syst",
    "m_outputAlgo" : "JetSelector_Syst",
    # ...
}

ElectronSelectorDict = {
    # ...
    "m_inputAlgo" : "ElectronCalibrator_Syst",
    "m_outputAlgo" : "ElectronSelector_Syst",
    # ...
```

(continues on next page)

(continued from previous page)

```
}

OverlapRemoverDict = {
     # ...
     "m_inputAlgoJets" : "JetSelector_Syst", # leave empty when not considering jet␣
↪systematics
     "m_inputAlgoElectrons" : "ElectronSelector_Syst", # leave empty when not␣
↪considering electron systematics
     # ...
}
```

In this way the overlap removal algorithm will be able to correctly work out all the combinatorics, generating output `xAOD` containers for jets and electrons for each input systematics combination to be subsequently used downstream according to the user's needs. The overlap removal algorithm creates an output systematic list that is a combination of systematics from all input containers.

Public Functions

OverlapRemover()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **fillObjectCutflow**(const xAOD::IParticleContainer *objCont, const std::string &overlapFlag = "passOR", const std::string &selectFlag = "passSel")

Fill the cutflow histograms.

Parameters

- **objCont** – The `xAOD` container to be considered
- **overlapFlag** – The string identifying objects not overlapping with another object, to be kept (default is `"passOR"`)
- **selectFlag** – The string identifying selected objects (default is `"passSel"`)

virtual EL::StatusCode **executeOR**(const xAOD::ElectronContainer *inElectrons, const xAOD::MuonContainer *inMuons, const xAOD::JetContainer *inJets, const xAOD::PhotonContainer *inPhotons, const xAOD::TauJetContainer *inTaus, *SystType* syst_type = NOMINAL, std::vector<std::string> *sysVec = nullptr, std::vector<std::string> *sysVecOut = nullptr)

Function that internally calls the OLR tool for the input containers (and systematics)

Parameters

- **inElectrons** – Input xAOD container for electrons
- **inMuons** – Input xAOD container for muons
- **inJets** – Input xAOD container for jets
- **inPhotons** – Input xAOD container for photons
- **inTaus** – Input xAOD container for taus
- **syst_type** – The type of object for which input systematics should be considered. Default is NOMINAL
- **sysVec** – The list of the input systematics for a given object. Must match with the choice of syst_type. Default is nullptr

EL::StatusCode **setCutFlowHist**()

Setup cutflow histograms.

EL::StatusCode **setCounters**()

Initialise counters for events/objects.

Public Members

bool **m_useCutFlow** = true

Fill the cutflow histogram(s) for object counting.

bool **m_decorateSelectedObjects**

Decorate selected objects (the default decoration string is passOR)

std::string **m_decor** = "passOR"

bool **m_createSelectedContainers**

Make a copy of input container(s) with selected objects (using SG::VIEW_ELEMENTS to be light weight)

bool **m_useSelected** = false

In the OLR, consider only objects passing a (pre)selection.

std::string **m_bTagWP** = ""

Use b-tagging decision, set previously with the given decoration name, to remove electrons and muons.

Note: This is automatically set by *BJetEfficiencyCorrector*

bool **m_linkOverlapObjects** = true

Create a link between overlapped objects.

bool **m_useBoostedLeptons** = false

Use boosted object working point.

bool **m_doEleEleOR** = false

Do overlap removal between electrons (HSG2 prescription)

bool **m_applyRelPt** = false

Turn ON ApplyRelPt in MuJetOverlapTool (default is false)

bool **m_lepFavWP** = false

Turn ON Lepton favored working point (HSG2 prescription)

std::string **m_outputAlgoSystNames** = "ORAlgo_Syst"

Output systematics list container name.

std::string **m_inContainerName_Electrons** = ""

Input container name.

std::string **m_outContainerName_Electrons** = ""

Output container name.

std::string **m_inputAlgoElectrons** = ""

Name of the `std::vector` of systematics coming from the upstream algorithm

std::string **m_inContainerName_Muons** = ""

std::string **m_outContainerName_Muons** = ""

std::string **m_inputAlgoMuons** = ""

std::string **m_inContainerName_Jets** = ""

std::string **m_outContainerName_Jets** = ""

std::string **m_inputAlgoJets** = ""

std::string **m_inContainerName_Photons** = ""

std::string **m_outContainerName_Photons** = ""

std::string **m_inputAlgoPhotons** = ""

std::string **m_inContainerName_Taus** = ""

std::string **m_outContainerName_Taus** = ""

std::string **m_inputAlgoTaus** = ""

Protected Types

enum **SystType**

An enum encoding systematics according to the various objects.

Values:

enumerator **NOMINAL**

enumerator **ELSYST**

enumerator **MUSYST**

enumerator **JETSYST**

enumerator **PHSYST**

enumerator **TAUSYST**

Protected Attributes

int **m_numEvent**

A counter for the number of processed events.

int **m_numObject**

A counter for the number of processed objects.

int **m_numEventPass**

A counter for the number of passed events.

int **m_weightNumEventPass**

A counter for the number of passed *weighted* events.

int **m_numObjectPass**

A counter for the number of passed objects.

bool **m_useElectrons** = false

Consider electrons in the OLR.

This is set to `false` if *m_inContainerName_Electrons* is set as an empty string. Electrons (unlike jets) are considered "optional" objetcs in the OLR.

bool **m_useMuons** = false

Consider muons in the OLR.

This is set to `false` if *m_inContainerName_Muons* is set as an empty string. Muons (unlike jets) are considered "optional" objects in the OLR.

bool **m_usePhotons** = false

Consider photons in the OLR.

This is set to `false` if *m_inContainerName_Photons* is set as an empty string. Photons (unlike jets) are considered "optional" objects in the OLR.

bool **m_useTaus** = false

Consider taus in the OLR.

This is set to `false` if *m_inContainerName_Taus* is set as an empty string. Taus (unlike jets) are considered "optional" objects in the OLR.

std::string **m_outAuxContainerName_Electrons**

Output auxiliary container name.

std::string **m_outAuxContainerName_Muons**

Output auxiliary container name.

std::string **m_outAuxContainerName_Jets**

Output auxiliary container name.

std::string **m_outAuxContainerName_Photons**

Output auxiliary container name.

std::string **m_outAuxContainerName_Taus**

Output auxiliary container name.

ORUtils::ToolBox **m_ORToolbox**

Pointer to the CP Tool which performs the actual OLR.

TH1D ***m_el_cutflowHist_1** = nullptr

Pointer to the histogram for the electron cutflow.

TH1D ***m_mu_cutflowHist_1** = nullptr

Pointer to the histogram for the muon cutflow.

TH1D ***m_jet_cutflowHist_1** = nullptr

Pointer to the histogram for the jet cutflow.

TH1D ***m_ph_cutflowHist_1** = nullptr

Pointer to the histogram for the photon cutflow.

TH1D ***m_tau_cutflowHist_1** = nullptr

Pointer to the histogram for the tau cutflow.

int **m_el_cutflow_OR_cut**

int **m_mu_cutflow_OR_cut**

int **m_jet_cutflow_OR_cut**

int **m_ph_cutflow_OR_cut**

int **m_tau_cutflow_OR_cut**

Class PhotonCalibrator

- Defined in file_xAODAnaHelpers_PhotonCalibrator.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **PhotonCalibrator** : public xAH::*Algorithm*

Public Functions

PhotonCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName** = ""

std::string **m_overridePhotonCalibMap** = ""

std::string **m_tightIDConfigPath** = "ElectronPhotonSelectorTools/offline/20180825/PhotonIsEMTightSelectorCutDefs.conf"

std::string **m_mediumIDConfigPath** = "ElectronPhotonSelectorTools/offline/mc15_20150712/PhotonIsEMMediumSelectorCutDefs.conf"

std::string **m_looseIDConfigPath** = "ElectronPhotonSelectorTools/offline/mc15_20150712/PhotonIsEMLooseSelectorCutDefs.conf"

bool **m_sort** = true

std::string **m_inputAlgoSystNames** = ""

this is the name of the vector of names of the systematically varied containers produced by the upstream algo (e.g., the SC containers with calibration systematics)

std::string **m_outputAlgoSystNames** = "PhotonCalibrator_Syst"

this is the name of the vector of names of the systematically varied containers produced by THIS algo (these will be the m_inputAlgoSystNames of the algo downstream

bool **m_useAFII** = false

bool **m_useAF3** = false

float **m_systVal** = 0.0

std::string **m_systName** = ""

std::string **m_esModel** = "es2017_R21_v1"

std::string **m_decorrelationModel** = ""

int **m_randomRunNumber** = -1

bool **m_readIDFlagsFromDerivation** = false

To read PID decision from DAOD, rather than recalculate with tool.

Class PhotonHistsAlgo

- Defined in file_xAODAnaHelpers_PhotonHistsAlgo.h

Inheritance Relationships

Base Type

- `public IParticleHistsAlgo` (*Class IParticleHistsAlgo*)

Class Documentation

class **PhotonHistsAlgo** : public *IParticleHistsAlgo*

Public Functions

PhotonHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **execute**()

Calls execute<IParticleContainer>

virtual EL::StatusCode **AddHists**(std::string name)

Calls AddHists<IParticleHists>

Parameters **name** – Name of the systematic

Class PhotonSelector

- Defined in file_xAODAnaHelpers_PhotonSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **PhotonSelector** : public xAH::*Algorithm*

Public Functions

PhotonSelector()

~PhotonSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::PhotonContainer *inPhotons, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::PhotonContainer> *selectedPhotons)

virtual bool **passCuts**(const xAOD::Photon *photon)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

configuration variables input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_inputAlgoSystNames** = ""

output auxiliary container name

std::string **m_outputAlgoSystNames** = "PhotonSelector_Syst"

bool **m_decorateSelectedObjects** = true

decorate selected objects - default "passSel"

bool **m_createSelectedContainer** = true

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1
: look at n objects

int **m_pass_min** = -1
: minimum number of objects passing cuts

int **m_pass_max** = -1
: maximum number of objects passing cuts

float **m_pT_max** = 1e8
: require pT < pt_max

float **m_pT_min** = 1e8
: require pT > pt_min

float **m_eta_max** = 1e8
: require |eta| < eta_max

bool **m_vetoCrack** = true
: require |eta| outside crack region

bool **m_doAuthorCut** = true

bool **m_doOQCut** = true

bool **m_readOQFromDerivation** = false
: read object quality from derivation, rather than calculating it on the fly

std::string **m_photonIdCut** = "None"
: Name of ID variable to cut

std::string **m_MinIsoWPCut** = ""
: reject objects which do not pass this isolation cut - default = "" (no cut)

std::string **m_IsoWPList** = "FixedCutTightCaloOnly,FixedCutTight,FixedCutLoose"
: decorate objects with 'isIsolated_*' flag for each WP in this input list - default = all current ASG WPs

Class TauCalibrator

- Defined in file_xAODAnaHelpers_TauCalibrator.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TauCalibrator** : public xAH::*Algorithm*

Public Functions

TauCalibrator()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName** = ""

std::string **m_RecommendationTag** = ""

bool **m_applyMVATESQualityCheck** = false

std::string **m_generator** = ""

std::string **m_campaign** = ""

bool **m_setAFII** = false

bool **m_setAF3** = false

bool **m_skipTruthMatchCheck** = false

bool **m_sort** = true

std::string **m_inputAlgoSystNames** = ""

this is the name of the vector of names of the systematically varied containers produced by the upstream algo (e.g., the SC containers with calibration systematics)

std::string **m_outputAlgoSystNames** = "TauCalibrator_Syst"

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

Class TauEfficiencyCorrector

- Defined in file_xAODAnaHelpers_TauEfficiencyCorrector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TauEfficiencyCorrector** : public xAH::*Algorithm*

Public Functions

TauEfficiencyCorrector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual EL::StatusCode **executeSF**(const xAOD::EventInfo *eventInfo, const xAOD::TauJetContainer *inputTaus, bool nominal, bool writeSystNames)

Public Members

std::string **m_RecommendationTag** = ""

std::string **m_inContainerName** = ""

std::string **m_WorkingPointReco** = ""

std::string **m_WorkingPointEleOLRHadTau** = ""

std::string **m_WorkingPointTauEleID** = ""

std::string **m_WorkingPointTauJetID** = ""

std::string **m_TriggerName** = ""

std::string **m_inputSystNamesTaus** = ""

The name of the vector containing the names of the systematically-varied taus-related containers from the upstream algorithm, which will be processed by this algorithm.

Only tau systematics or any other that create shallow copies of tau containers should be passed to this tool. It is advised to run this algorithm before running algorithms combining multiple calibration systematics (e.g. overlap removal).

bool **m_writeSystToMetadata** = false

Write systematics names to metadata.

float **m_systVal** = 0.0

std::string **m_systName** = ""

std::string **m_outputSystNames** = "TauEfficiencyCorrector_Syst"

Class TauJetMatching

- Defined in file_xAODAnaHelpers_TauJetMatching.h

Inheritance Relationships

Base Type

- public xAH::Algorithm (*Class Algorithm*)

Class Documentation

class **TauJetMatching** : public xAH::*Algorithm*

Public Functions

TauJetMatching()

~TauJetMatching()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeDecoration**(std::unordered_map<int, std::pair<const xAOD::TauJet*, const xAOD::Jet*>>, const xAOD::TauJetContainer *tauCont)

float **getDR**(float eta1, float eta2, float phi1, float phi2)

std::unordered_map<int, std::pair<const xAOD::TauJet*, const xAOD::Jet*>> **findBestMatchDR**(const xAOD::JetContainer *jetCont, const xAOD::TauJetContainer *tauCont, float best_DR)

Public Members

std::string **m_inContainerName** = ""

std::string **m_outContainerName**

std::string **m_outAuxContainerName**

std::string **m_inputAlgoSystNames** = ""

std::string **m_outputAlgoSystNames** = "TauJetMatching_Syst"

std::string **m_inJetContainerName** = ""

float **m_DeltaR** = 0.2

Class TauSelector

- Defined in file_xAODAnaHelpers_TauSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TauSelector** : public xAH::*Algorithm*

Public Functions

TauSelector()

~TauSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

bool **executeSelection**(const xAOD::TauJetContainer *inTaus, float mcEvtWeight, bool countPass, ConstDataVector<xAOD::TauJetContainer> *selectedTaus)

virtual int **passCuts**(const xAOD::TauJet *tau)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

std::string **m_outContainerName**

std::string **m_outAuxContainerName**

std::string **m_inputAlgoSystNames** = ""

std::string **m_outputAlgoSystNames** = "TauSelector_Syst"

bool **m_decorateWithTracks** = false

bool **m_decorateSelectedObjects** = true

std::string **m_decorationName** = "passSel"

bool **m_createSelectedContainer** = false

int **m_nToProcess** = -1

int **m_pass_min** = -1

int **m_pass_max** = -1

std::string **m_ConfigPath** = "xAODAnaHelpers/TauConf/00-01-19/Selection/recommended_selection_mc15.conf"

float **m_minPtDAOD** = 15e3

std::string **m_JetIDWP** = ""

std::string **m_EleRNNWP** = ""

bool **m_EleID** = true

std::string **m_singleTauTrigChains** = ""

std::string **m_diTauTrigChains** = ""

Class TrackHistsAlgo

- Defined in file_xAODAnaHelpers_TrackHistsAlgo.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TrackHistsAlgo** : public xAH::*Algorithm*

Public Functions

TrackHistsAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

Public Members

std::string **m_inContainerName** = ""

std::string **m_detailStr** = ""

Class TrackSelector

- Defined in file_xAODAnaHelpers_TrackSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TrackSelector** : public xAH::*Algorithm*

Public Functions

TrackSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

EL::StatusCode **executeTrackCollection**(float mcEvtWeight)

EL::StatusCode **executeTracksInJets**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual int **PassCuts**(const xAOD::TrackParticle *jet, const xAOD::Vertex *pvx)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""
: input container name

std::string **m_outContainerName** = ""
: output container name

std::string **m_inJetContainerName** = ""
: input jet container name

bool **m_decorateSelectedObjects** = true
: decorate selected objects? defaul passSel

bool **m_createSelectedContainer** = false
: fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1
: look at n objects

int **m_pass_min** = -1
: minimum number of objects passing cuts

int **m_pass_max** = -1
: maximum number of objects passing cuts

std::string **m_cutLevelString** = ""
: available: Loose LoosePrimary TightPrimary LooseMuon LooseElectron MinBias HILoose HITight

float **m_pT_max** = 1e8
: require pT < pt_max

float **m_pT_min** = 1e8
: require pT > pt_max

float **m_p_min** = 1e8
: require |p| > p_min

float **m_eta_max** = 1e8
: require |eta| < eta_max

float **m_eta_min** = 1e8
: require |eta| > eta_min

float **m_etaSigned_min** = 1e8
: require eta > eta_min

float **m_etaSigned_max** = 1e8
: require eta < eta_max

float **m_d0_max** = 1e8
: require |d0| < d0_max

float **m_z0_max** = 1e8
: require |z0| < z0_max

float **m_sigmad0_max** = 1e8
: maximum error on d0

float **m_d0oversigmad0_max** = 1e8
: maximum significance of |d0|

float **m_z0sinT_max** = 1e8
: require |z0xsin(theta)| < z0sintheta_max

float **m_sigmaz0_max** = 1e8
: maximum error on z0

float **m_sigmaz0sintheta_max** = 1e8
: maximum error on z0*sin(theta)

float **m_z0oversigmaz0_max** = 1e8
: max |z0| significance

float **m_z0sinthetaoversigmaz0sintheta_max** = 1e8
: max |z0sin(theta)| significance

int **m_nPixelHits_min** = 1e8
: minimum pixel hits (counting dead sensors)

int **m_nPixelHitsPhysical_min** = 1e8
: minimum pixel hits (no dead sensors)

int **m_nSctHits_min** = 1e8
: minimum SCT hits (counting dead sensors)

int **m_nSctHitsPhysical_min** = 1e8
: minimum SCT hits (no dead sensors)

int **m_nSi_min** = 1e8

require nSi >= nSi_min (nSi = nPix + nSct)

int **m_nSiPhysical_min** = 1e8

require nSi >= nSi_min (nSi = nPix + nSct, no dead sensors)

int **m_nPixHoles_max** = 1e8

require nPixHoles <= nPixHoles_max

int **m_nSctHoles_max** = 1e8

require nSCTHoles <= nSCTHoles_max

int **m_nSiHoles_max** = 1e8

maximum silicon holes

int **m_nInnermostPixel_min** = 1e8

minimum nIBL (if expected)

int **m_nNextToInnermostPixel_min** = 1e8

minimum nBL (if expected)

int **m_nBothInnermostLayersHits_min** = 1e8

minimum nIBL + nBL (if every hit that is not expected, we require one less)

int **m_nPixelSharedHits_max** = 1e8

maximum pixel hits shared with other tracks

int **m_nSctSharedHits_max** = 1e8

maximum SCT hits shared with other tracks

int **m_nSiSharedHits_max** = 1e8

maximum silicon hits shared with other tracks

int **m_nSiSharedModules_max** = 1e8

maximum (pixel + SCT/2) shared hits

float **m_chi2NdofCut_max** = 1e8

require chi2/ndof < chi2NdofCut_max

float **m_chi2Prob_max** = 1e8

require TMath::Prob(chi2,ndof) < chi2ProbMax

float **m_chi2Prob_min** = 1e8

require TMath::Prob(chi2,ndof) > chi2ProbMax

int **m_nBL_min** = 1e8
require nIBL >= nBL_min (not recommended; for downward compatibility)

std::string **m_passAuxDecorKeys** = ""

std::string **m_failAuxDecorKeys** = ""

bool **m_doTracksInJets** = false
do track selection on track within jets

Class TreeAlgo

- Defined in file_xAODAnaHelpers_TreeAlgo.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TreeAlgo** : public xAH::*Algorithm*

Public Functions

TreeAlgo()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual *HelpTreeBase* ***createTree**(xAOD::TEvent *event, TTree *tree, TFile *file, const float units, bool debug, xAOD::TStore *store)

Public Members

bool **m_outHistDir** = false

std::string **m_evtDetailStr** = ""

std::string **m_trigDetailStr** = ""

std::string **m_muDetailStr** = ""

std::string **m_elDetailStr** = ""

std::string **m_jetDetailStr** = ""

std::string **m_trigJetDetailStr** = ""

std::string **m_truthJetDetailStr** = ""

std::string **m_fatJetDetailStr** = ""

std::string **m_truthFatJetDetailStr** = ""

std::string **m_tauDetailStr** = ""

std::string **m_METDetailStr** = ""

std::string **m_METReferenceDetailStr** = ""

std::string **m_photonDetailStr** = ""

std::string **m_clusterDetailStr** = ""

std::string **m_truthParticlesDetailStr** = ""

std::string **m_trackParticlesDetailStr** = ""

std::string **m_vertexDetailStr** = ""

std::string **m_evtContainerName** = ""

std::string **m_muContainerName** = ""

std::string **m_elContainerName** = ""

std::string **m_jetContainerName** = ""

std::string **m_jetBranchName** = "jet"

std::string **m_truthJetContainerName** = ""

std::string **m_truthJetBranchName** = "truthJet"

std::string **m_trigJetContainerName** = ""

std::string **m_trigJetBranchName** = "trigJet"

std::string **m_fatJetContainerName** = ""

std::string **m_fatJetBranchName** = ""

std::string **m_truthFatJetContainerName** = ""

std::string **m_truthFatJetBranchName** = "truth_fatjet"

std::string **m_tauContainerName** = ""

std::string **m_METContainerName** = ""

std::string **m_METReferenceContainerName** = ""

std::string **m_photonContainerName** = ""

std::string **m_clusterContainerName** = ""

std::string **m_clusterBranchName** = "CaloCalTopoClusters"

std::string **m_truthParticlesContainerName** = ""

std::string **m_truthParticlesBranchName** = "xAH_truth"

std::string **m_trackParticlesContainerName** = ""

std::string **m_l1JetContainerName** = ""

std::string **m_l1JetBranchName** = "L1Jet"

std::string **m_vertexBranchName** = "vertex"

bool **m_sortL1Jets** = false

bool **m_retrievePV** = true

std::string **m_muSystsVec** = ""

std::string **m_elSystsVec** = ""

std::string **m_tauSystsVec** = ""

std::string **m_jetSystsVec** = ""

std::string **m_photonSystsVec** = ""

std::string **m_fatJetSystsVec** = ""

std::string **m_metSystsVec** = ""

float **m_units** = 1e3

unit conversion from MeV, default is GeV

int **m_autoFlush** = 0

Set to a large negative number, such as -1000000, to ensure that the tree flushes memory after a reasonable amount of time. Otherwise, jobs with a lot of systematics use too much memory.

Protected Attributes

std::vector<std::string> **m_jetDetails**

std::vector<std::string> **m_trigJetDetails**

std::vector<std::string> **m_fatJetDetails**

std::vector<std::string> **m_jetContainers**

std::vector<std::string> **m_truthJetContainers**

std::vector<std::string> **m_trigJetContainers** std::vector<std::string> **m_fatJetContainers**

std::vector<std::string> **m_l1JetContainers**

std::vector<std::string> **m_vertexContainers**

std::vector<std::string> **m_truthParticlesContainers**

std::vector<std::string> **m_jetBranches**

std::vector<std::string> **m_truthJetBranches**

std::vector<std::string> **m_trigJetBranches**

std::vector<std::string> **m_fatJetBranches**

std::vector<std::string> **m_l1JetBranches**

std::vector<std::string> **m_vertexBranches**

std::vector<std::string> **m_truthParticlesBranches**

std::vector<std::string> **m_clusterDetails**

std::vector<std::string> **m_clusterContainers**

std::vector<std::string> **m_clusterBranches**

std::vector<std::string> **m_vertexDetails**

std::map<std::string, *HelpTreeBase**> **m_trees**

Class TrigMatcher

- Defined in file_xAODAnaHelpers_TrigMatcher.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TrigMatcher** : public xAH::*Algorithm*

A wrapper of the trigger matching tool in the ASG TriggerMatchingTool package.

The idea behind this algorithm is to decorate inputs from upstream xAODAnaHelpers algorithms based on user's configuration, handling also the case where systematics on the input physics objects are taken into account. The list of trigger chains that contain a trigger object matched to a reconstructed object are saved in the `trigMatched` decoration as :cpp:any:`std::vector<std::string>.

Here follows a usage example for photon matching.

Assuming the typical xAODAnaHelpers analysis configuration through `xAH_config`, the analysis workflow could look like the following:

```
c = xAH_config()
# ...
c.algorithm("PhotonSelector", PhotonSelectorDict)
# ...
c.algorithm("TrigMatcher", TrigMatcherDict)
# ...
```

where each algorithm has the following I/O systematics configuration (via python dictionaries):

```
 PhotonSelectorDict = {
     # ...
     "m_inputAlgo" : "PhotonCalibrator_Syst",
 "m_outContainerName" : "PhotonSelector_Out",
     "m_outputAlgo" : "PhotonSelector_Syst",
     # ...
}

TrigMatcherDict = {
     # ...
 "m_inContainerName" : "PhotonSelector_Out",
 "m_systNames" : "PhotonSelector_Syst",  # leave empty when not considering systematics
 "m_trigChains" : "HLT_g120_loose,HLT_g140_loose"
     # ...
}
```

In this way the trigger matching algorithm will be able to correctly work out all the combinatorics, assigning decoration to input objects in each input systematics combination to be subsequently used downstream according to the user's needs.

Public Functions

TrigMatcher()

/* contains all the HLT trigger chains tokens extracted from m_trigChains */

~TrigMatcher()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

EL::StatusCode **executeMatching**(const xAOD::IParticleContainer *inParticles)

Public Members

std::string **m_inContainerName** = ""

Input container name.

std::string **m_systNames** = ""

Input systematics list container name.

std::string **m_trigChains** = ""

Comma-separated list of trigger chains.

A comma-separated string w/ all the HLT trigger chains for which you want to perform the matching. If left empty (as it is by default), no trigger matching will be attempted at all

Class TruthSelector

- Defined in file_xAODAnaHelpers_TruthSelector.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **TruthSelector** : public xAH::*Algorithm*

Public Functions

TruthSelector()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

virtual bool **executeSelection**(const xAOD::TruthParticleContainer *inTruthParts, float mcEvtWeight, bool count, std::string outContainerName)

virtual int **PassCuts**(const xAOD::TruthParticle *truthPart)

Public Members

bool **m_useCutFlow** = true

std::string **m_inContainerName** = ""

input container name

std::string **m_outContainerName** = ""

output container name

std::string **m_decor** = "passSel"

The decoration key written to passing objects.

bool **m_decorateSelectedObjects** = true

decorate selected objects? defaul passSel

bool **m_createSelectedContainer** = false

fill using SG::VIEW_ELEMENTS to be light weight

int **m_nToProcess** = -1

look at n objects

int **m_pass_min** = -1

minimum number of objects passing cuts

int **m_pass_max** = -1
maximum number of objects passing cuts

float **m_pT_max** = 1e8
require pT < pt_max

float **m_pT_min** = 1e8
require pT > pt_min

float **m_eta_max** = 1e8
require eta < eta_max

float **m_eta_min** = 1e8
require eta > eta_max

float **m_mass_max** = 1e8
require mass < mass_max

float **m_mass_min** = 1e8
require mass > mass_max

float **m_rapidity_max** = 1e8
require rapidity < rapidity_max

float **m_rapidity_min** = 1e8
require rapidity > rapidity_min

unsigned int **m_type** = 1000
require classifierParticleType == type (defined by TruthClassifier: https://gitlab.cern.ch/atlas/athena/blob/21.2/PhysicsAnalysis/MCTruthClassifier/MCTruthClassifier/MCTruthClassifierDefs.h)

std::string **m_typeOptions**
require classifierParticleType to match any of the "|" separated type values (e.g. "1|2|3|4")

unsigned int **m_origin** = 1000
require classifierParticleOrigin == origin (defined by TruthClassifier: https://gitlab.cern.ch/atlas/athena/blob/21.2/PhysicsAnalysis/MCTruthClassifier/MCTruthClassifier/MCTruthClassifierDefs.h)

std::string **m_originOptions**
require classifierParticleOrigin to match any of the "|" separated origin values (e.g. "10|12|13")

float **m_pT_dressed_min** = 1e8
require pt_dressed > pt_dressed_min

float **m_eta_dressed_min** = 1e8
require eta_dressed > eta_dressed_min

float **m_eta_dressed_max** = 1e8

require eta_dressed > eta_dressed_max

Class Writer

- Defined in file_xAODAnaHelpers_Writer.h

Inheritance Relationships

Base Type

- `public xAH::Algorithm` (*Class Algorithm*)

Class Documentation

class **Writer** : public xAH::*Algorithm*

Public Functions

Writer()

virtual EL::StatusCode **setupJob**(EL::Job &job)

virtual EL::StatusCode **fileExecute**()

virtual EL::StatusCode **histInitialize**()

virtual EL::StatusCode **changeInput**(bool firstFile)

virtual EL::StatusCode **initialize**()

virtual EL::StatusCode **execute**()

virtual EL::StatusCode **postExecute**()

virtual EL::StatusCode **finalize**()

virtual EL::StatusCode **histFinalize**()

ClassDef (Writer, 1)

Public Members

TString **m_outputLabel** = ""

TString **m_jetContainerNamesStr** = ""

TString **m_electronContainerNamesStr** = ""

TString **m_muonContainerNamesStr** = ""

Class Algorithm

- Defined in file_xAODAnaHelpers_Algorithm.h

Inheritance Relationships

Base Type

- `public EL::Algorithm`

Derived Types

- `public BJetEfficiencyCorrector` (*Class BJetEfficiencyCorrector*)
- `public BasicEventSelection` (*Class BasicEventSelection*)
- `public ClusterHistsAlgo` (*Class ClusterHistsAlgo*)
- `public DebugTool` (*Class DebugTool*)
- `public ElectronCalibrator` (*Class ElectronCalibrator*)
- `public ElectronEfficiencyCorrector` (*Class ElectronEfficiencyCorrector*)
- `public ElectronSelector` (*Class ElectronSelector*)
- `public HLTJetGetter` (*Class HLTJetGetter*)
- `public IParticleHistsAlgo` (*Class IParticleHistsAlgo*)
- `public IsoCloseByCorr` (*Class IsoCloseByCorr*)
- `public JetCalibrator` (*Class JetCalibrator*)
- `public JetSelector` (*Class JetSelector*)
- `public METConstructor` (*Class METConstructor*)
- `public MessagePrinterAlgo` (*Class MessagePrinterAlgo*)
- `public MetHistsAlgo` (*Class MetHistsAlgo*)
- `public MinixAOD` (*Class MinixAOD*)
- `public MuonCalibrator` (*Class MuonCalibrator*)
- `public MuonEfficiencyCorrector` (*Class MuonEfficiencyCorrector*)

- public MuonInFatJetCorrector (*Class MuonInFatJetCorrector*)
- public MuonSelector (*Class MuonSelector*)
- public OverlapRemover (*Class OverlapRemover*)
- public PhotonCalibrator (*Class PhotonCalibrator*)
- public PhotonSelector (*Class PhotonSelector*)
- public TauCalibrator (*Class TauCalibrator*)
- public TauEfficiencyCorrector (*Class TauEfficiencyCorrector*)
- public TauJetMatching (*Class TauJetMatching*)
- public TauSelector (*Class TauSelector*)
- public TrackHistsAlgo (*Class TrackHistsAlgo*)
- public TrackSelector (*Class TrackSelector*)
- public TreeAlgo (*Class TreeAlgo*)
- public TrigMatcher (*Class TrigMatcher*)
- public TruthSelector (*Class TruthSelector*)
- public Writer (*Class Writer*)

Class Documentation

class **Algorithm** : public EL::Algorithm

This is used by all algorithms within xAODAnaHelpers.

The main goal of this algorithm class is to standardize how everyone defines an algorithm that plugs into xAODAnaHelpers. A series of common utilities are provided such as *m_className* which defines the class name so we can manage a registry *m_instanceRegistry* to keep xAODAnaHelpers as flexible as possible to our users.

We expect the user to create a new algorithm, such as a selector for jets:

```
class JetSelector : public xAH::Algorithm
{
    // ...
};
```

The above example is taken from our implementation in *JetSelector*. Just remember that when you write your initializer, you will be expected to do something like:

```
// this is needed to distribute the algorithm to the workers
ClassImp(JetSelector)


JetSelector :: JetSelector () :
    Algorithm("JetSelector"),
    ...
{
    // ...
}
```

which this class will automatically register all instances of for you. Each instance can have a different algorithm name but will have the same *m_className* so we can track how many references have been made. This is useful for selectors to deal with cutflows, but can be useful for other algorithms that need to know how many times they've been instantiated in a single job.

Note: The expectation is that the user does not directly use this class but rather inherits from it.

Subclassed by *BJetEfficiencyCorrector*, *BasicEventSelection*, *ClusterHistsAlgo*, *DebugTool*, *ElectronCalibrator*, *ElectronEfficiencyCorrector*, *ElectronSelector*, *HLTJetGetter*, *IParticleHistsAlgo*, *IsoCloseByCorr*, *JetCalibrator*, *JetSelector*, *METConstructor*, *MessagePrinterAlgo*, *MetHistsAlgo*, *MinixAOD*, *MuonCalibrator*, *MuonEfficiencyCorrector*, *MuonInFatJetCorrector*, *MuonSelector*, *OverlapRemover*, *PhotonCalibrator*, *PhotonSelector*, *TauCalibrator*, *TauEfficiencyCorrector*, *TauJetMatching*, *TauSelector*, *TrackHistsAlgo*, *TrackSelector*, *TreeAlgo*, *TrigMatcher*, *TruthSelector*, *Writer*

Public Functions

`Algorithm`(std::string className = "Algorithm")

Initialization.

Parameters **`className`** – This is the name of the class that inherits from :cpp:namespace:~`xAH::Algorithm`

`~Algorithm`()

StatusCode **`algInitialize`**()

Run any initializations commmon to all xAH Algorithms (such as registerInstance). Call this inside `histInitialize` for best results.

StatusCode **`algFinalize`**()

Run any finalizations common to all xAH Algorithms (such as unregisterInstance). Call this inside `histFinalize` for best results.

StatusCode **`parseSystValVector`**()

Parse string of systematic sigma levels in m_systValVectorString into m_systValVector.

Public Members

std::string **`m_name`** = "UnnamedAlgorithm"

All algorithms initialized should have a unique name, to differentiate them at the TObject level.

Note, `GetName()` returns a `char*` while this returns a `std::string`.

bool **`m_debug`** = false

m_debug is being deprecated

bool **`m_verbose`** = false

m_verbose is being deprecated

MSG::Level **`m_msgLevel`** = MSG::INFO

debug level

std::string **m_cutFlowStreamName** = "cutflow"

std::string **m_systName** = ""

If running systematics, the name of the systematic

float **m_systVal** = 0.0

If running systematics, the value to set the systematic to

Note: This will set the systematic to the value $\pm x$.

std::string **m_systValVectorString** = ""

If running systematics, you can run multiple points and store them in here. A comma separated list of working points should be given to m_systValVectorString, and then parsed by calling parseSystValVector.

std::vector<float> **m_systValVector**

std::string **m_eventInfoContainerName** = "EventInfo"

If the xAOD has a different *EventInfo* container name, set it here

std::string **m_vertexContainerName** = "PrimaryVertices"

If the xAOD has a different PrimaryVertex container name, set it here

int **m_isMC** = -1

This stores the isMC decision, and can also be used to override at the algorithm level to force analyzing MC or not.

Value	Meaning
-1	Default, use eventInfo object to determine if data or mc
0	Treat the input as data
1	Treat the input as MC

int **m_isFastSim** = -1

This stores the isFastSim decision, and can also be used to override at the algorithm level to force analyzing FastSim or not.

Value	Meaning
-1	Default, use Metadata object to determine if FullSim or FastSim
0	Treat the input as FullSim
1	Treat the input as FastSim

int **m_isAF3** = -1

This stores the isAF3 decision, and can also be used to override at the algorithm level to force analyzing FastSim with AF3 or not.

Value	Meaning
-1	Default, use Metadata object to determine if AF3 FastSim or not
0	Treat the input as FullSim or AFII
1	Treat the input as FastSim with AF3

bool **m_useRun3navigation** = false

Flag to use Run 3 trigger navigation (true), or Run 2 navigation (false)

std::string **m_HLTSummary** = "HLTNav_Summary_DAODSlimmed"

String storing the type of HLT navigation info available for Run 3 samples. For AODs or unslimmed DAODs: HLTNav_Summary_AODSlimmed

bool **m_forceFastSim** = false

Flags to force a specific data-type, even if it disagrees with your input

bool **m_forceFullSim** = false

bool **m_forceData** = false

bool **m_setAFII** = false

Backwards compatibility, same as m_forceFastSim

bool **m_setAF3** = false

Protected Functions

bool **isMC**()

```
\verbatim embed:rst:leading-asterisk
    Try to determine if we are running over data or MC. The␣
↪:cpp:member:`xAH::Algorithm::m_isMC` can be used
```

to fix the return value. Otherwise the *EventInfo* object is queried.

An exception is thrown if the type cannot be determined.

```
    ============ =======
    Return Value Meaning
    ============ =======
    0            Data
    1            MC
    ============ =======
```

bool **isFastSim**()

```
\verbatim embed:rst:leading-asterisk
    Try to determine if we are running over data or MC. The␣
↪:cpp:member:`xAH::Algorithm::m_isFastSim` can be used
```

to fix the return value. Otherwise the metadata is queried.

An exception is thrown if the type cannot be determined.

```
============ =======
Return Value Meaning
============ =======
0            FullSim (or Data)
1            FastSim
============ =======
```

bool **isAF3**()

If the name includes ATLFASTII or ATLFAST3 then set to AFII or AF3, if deemed fullSim then FS else leave as empty string and complain

bool **isPHYS**()

Determines if using DAOD_PHYS or not.

void **registerInstance**()

Register the given instance under the moniker *xAH::Algorithm::m_className*

This will increase the reference count by 1.

int **numInstances**()

Return number of instances registered under the moniker *xAH::Algorithm::m_className*

This will return the reference count.

> **Warning:** If for some reason the instance wasn't registered, we spit out a warning.

void **unregisterInstance**()

Unregister the given instance under the moniker *xAH::Algorithm::m_className*

This will decrease the reference count by 1.

> **Warning:** If for some reason the instance wasn't registered, we spit out a warning.

template<typename **T**>
inline StatusCode **checkToolStore**(const std::string &tool_name)

```
\verbatim embed:rst:leading-asterisk
    Check whether the input CP tool already exists with *this* name in the
↪asg::ToolStore
```

Depending on the outcome, the content of the map :cpp:member:`xAH::Algorithm::m_toolAlreadyUsed` wll be set accordingly.

```

```

inline bool **isToolAlreadyUsed**(const std::string &tool_name)

Check whether the input CP tool has been already used by any *xAH::Algorithm* in the current job by scanning *xAH::Algorithm::m_toolAlreadyUsed*.

template<typename **T**>
inline void **setToolName**(__attribute__((unused)) asg::AnaToolHandle<*T*> &handle, __attribute__((unused)) const std::string &name = "") const

Sets the name of a tool. If no name is needed, the tool will use the name of the algorithm plus a unique identifier (*xAH::Algorithm::getAddress()*) appended to ensure the tool is unique and effectively private.

The tool will not be guaranteed unique if two tools of the same type are created without a name passed in. But this is, at this point, up to the user and a more complex scenario than what this function tries to simplify on its own.

inline std::string **getAddress**() const

Return a `std::string` representation of `this`

Protected Attributes

std::string **m_className** = "Algorithm"

The moniker by which all instances are tracked in *xAH::Algorithm::m_instanceRegistry*

xAOD::TEvent ***m_event** = nullptr

The TEvent object

xAOD::TStore ***m_store** = nullptr

The TStore object

Class Cluster

- Defined in file_xAODAnaHelpers_Cluster.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **Cluster** : public xAH::*Particle*

Class ClusterContainer

- Defined in file_xAODAnaHelpers_ClusterContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Cluster, HelperClasses::ClusterInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **ClusterContainer** : public xAH::*ParticleContainer*<*Cluster*, *HelperClasses*::*ClusterInfoSwitch*>

Public Functions

ClusterContainer(const std::string &name = "clus", const std::string &detailStr = "", float units = 1e3, bool mc = false)

virtual **~ClusterContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillCluster**(const xAOD::CaloCluster *cluster)

virtual void **FillCluster**(const xAOD::IParticle *particle)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *Cluster* &cluster)

Class Electron

- Defined in file_xAODAnaHelpers_Electron.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **Electron** : public xAH::*Particle*

Public Members

float **caloCluster_eta**

float **charge**

int **isTrigMatched**

std::vector<int> **isTrigMatchedToChain**

std::vector<std::string> **listTrigChains**

std::map<std::string, int> **isIsolated**

float **etcone20**

float **ptcone20**

float **ptcone30**

float **ptcone40**

float **ptvarcone20**

float **ptvarcone30**

float **ptvarcone40**

float **topoetcone20**

float **ptcone20_Nonprompt_All_MaxWeightTTVALooseCone_pt500** float **ptcone20_Nonprompt_All_MaxWeightTTVALooseCone_pt1000**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVALooseCone_pt500**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVALooseCone_pt1000**

float **topoetcone30**

float **topoetcone40**

float **neflowisol20**

float **topoetcone20_CloseByCorr**

float **ptcone20_Nonprompt_All_MaxWeightTTVALooseCone_pt1000_CloseByCorr**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVALooseCone_pt1000_CloseByCorr**

std::map<std::string, int> **PID**

std::vector<float> **RecoEff_SF**

std::map<std::string, std::vector<float>> **PIDEff_SF**

std::map<std::string, std::vector<float>> **IsoEff_SF**

std::map<std::string, std::vector<float>> **TrigEff_SF**

std::map<std::string, std::vector<float>> **TrigMCEff**

int **author**

int **OQ**

float **trkd0**

float **trkd0sig**

float **trkz0**

float **trkz0sintheta**

float **trkphi0**

float **trktheta**

float **trkcharge**

float **trkqOverP**

int **trknSiHits**

int **trknPixHits**

int **trknPixHoles**

int **trknSCTHits**

int **trknSCTHoles**

int **trknTRTHits**

int **trknTRTHoles**

int **trknBLayerHits**

int **trknInnermostPixLayHits**

float **trkPixdEdX**

float **PromptLeptonInput_DL1mu**

float **PromptLeptonInput_DRlj**

float **PromptLeptonInput_LepJetPtFrac**

float **PromptLeptonInput_PtFrac**

float **PromptLeptonInput_PtRel**

int **PromptLeptonInput_TrackJetNTrack**

float **PromptLeptonInput_ip2**

float **PromptLeptonInput_ip3**

float **PromptLeptonInput_rnnip**

int **PromptLeptonInput_sv1_jf_ntrkv**

float **PromptLeptonIso**

float **PromptLeptonVeto**

char **passSel**

char **passOR**

char **isLRT**

Class ElectronContainer

- Defined in file_xAODAnaHelpers_ElectronContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Electron, HelperClasses::ElectronInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **ElectronContainer** : public xAH::*ParticleContainer*<*Electron*, *HelperClasses*::*ElectronInfoSwitch*>

Public Functions

ElectronContainer(const std::string &name = "el", const std::string &detailStr = "", float units = 1e3, bool mc = false, bool storeSystSFs = true)

virtual **~ElectronContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillElectron**(const xAOD::Electron *elec, const xAOD::Vertex *primaryVertex) virtual void **FillElectron**(const xAOD::IParticle *particle, const xAOD::Vertex *primaryVertex)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *Electron* &elec)

Class EventInfo

- Defined in file_xAODAnaHelpers_EventInfo.h

Class Documentation

class **EventInfo**

Public Functions

EventInfo(const std::string &detailStr = "", float units = 1e3, bool mc = false, bool storeSyst = true)

~EventInfo()

void **setTree**(TTree *tree)

void **setBranches**(TTree *tree)

void **clear**()

void **FillEvent**(const xAOD::EventInfo *eventInfo, xAOD::TEvent *event = nullptr, const xAOD::VertexContainer *vertices = nullptr)

template<typename **T_BR**>
void **connectBranch**(TTree *tree, std::string name, *T_BR* *variable)

Public Members

HelperClasses::*EventInfoSwitch* **m_infoSwitch**

bool **m_mc**

bool **m_debug**

bool **m_storeSyst**

float **m_units**

int **m_runNumber**

Long64_t **m_eventNumber**

int **m_lumiBlock**

uint32_t **m_coreFlags**

uint32_t **m_timeStamp**

uint32_t **m_timeStampNSOffset**

bool **m_TileError**

bool **m_LArError**

bool **m_SCTError**

uint32_t **m_TileFlags**

uint32_t **m_LArFlags**

uint32_t **m_SCTFlags**

bool **m_eventClean_LooseBad**

bool **m_eventClean_TightBad**

int **m_mcEventNumber**

int **m_mcChannelNumber**

float **m_mcEventWeight**

std::vector<float> **m_mcEventWeights**

float **m_weight_pileup**

float **m_weight_pileup_up**

float **m_weight_pileup_down**

float **m_correctedAvgMu**

float **m_correctedAndScaledAvgMu**

float **m_correctedMu**

float **m_correctedAndScaledMu**

int **m_rand_run_nr**

int **m_rand_lumiblock_nr**

int **m_bcid**

int **m_DistEmptyBCID**

int **m_DistLastUnpairedBCID**

int **m_DistNextUnpairedBCID**

int **m_npv**

float **m_actualMu**

float **m_averageMu**

double **m_rhoEM**

double **m_rhoEMPFLOW**

double **m_rhoLC**

float **m_beamspotweight**

int **m_pdgId1**

int **m_pdgId2**

int **m_pdfId1**

int **m_pdfId2**

float **m_x1**

float **m_x2**

float **m_q**

float **m_xf1**

float **m_xf2**

std::vector<float> **m_caloCluster_pt**

std::vector<float> **m_caloCluster_eta**

std::vector<float> **m_caloCluster_phi**

std::vector<float> **m_caloCluster_e**

Class FatJet

- Defined in file_xAODAnaHelpers_FatJet.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **FatJet** : public xAH::*Particle*

Public Members

float **JetConstitScaleMomentum_eta**

float **JetConstitScaleMomentum_phi**

float **JetConstitScaleMomentum_m**

float **JetConstitScaleMomentum_pt**

float `JetEMScaleMomentum_eta`

float `JetEMScaleMomentum_phi`

float `JetEMScaleMomentum_m`

float `JetEMScaleMomentum_pt`

float `GhostArea`

float `ActiveArea`

float `VoronoiArea`

float `ActiveArea4vec_pt`

float `ActiveArea4vec_eta`

float `ActiveArea4vec_phi`

float `ActiveArea4vec_m`

float `Split12`

float `Split23`

float `Split34`

float `tau1_wta`

float `tau2_wta`

float `tau3_wta`

float `tau21_wta`

float `tau32_wta`

float `ECF1`

float `ECF2`

float **ECF3**

float **C2**

float **D2**

float **NTrimSubjets**

int **NClusters**

int **nTracks**

int **ungrtrk500**

float **EMFrac**

int **nChargedParticles**

int **numConstituents**

std::vector<float> **constituentWeights**

std::vector<float> **constituent_pt**

std::vector<float> **constituent_eta**

std::vector<float> **constituent_phi**

std::vector<float> **constituent_e**

TLorentzVector **truth_p4**

int **nTQuarks**

int **nHBosons**

int **nWBosons**

int **nZBosons**

int **Wtag_medium**

int **Ztag_medium**

int **Wtag_tight**

int **Ztag_tight**

std::unordered_map<std::string, std::vector<xAH::*Jet*>> **trkJets**

float **muonCorrected_pt**

float **muonCorrected_eta**

float **muonCorrected_phi**

float **muonCorrected_m**

Class FatJetContainer

- Defined in file_xAODAnaHelpers_FatJetContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< FatJet, HelperClasses::JetInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **FatJetContainer** : public xAH::*ParticleContainer*<*FatJet*, *HelperClasses*::*JetInfoSwitch*>

Public Functions

FatJetContainer(const std::string &name = "fatjet", const std::string &detailStr = "", const std::string &subjetDetailStr = "kinematic", const std::string &suffix = "", float units = 1e3, bool mc = false)

virtual **~FatJetContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillFatJet**(const xAOD::Jet *jet, int pvLocation = 0)

virtual void **FillFatJet**(const xAOD::IParticle *particle, int pvLocation = 0)

inline void **setTree**(TTree *tree)

Public Members

float **m_trackJetPtCut** = 10e3

float **m_trackJetEtaCut** = 2.5

Protected Functions

virtual void **updateParticle**(uint idx, *FatJet* &jet)

Class Jet

- Defined in file_xAODAnaHelpers_Jet.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **Jet** : public xAH::*Particle*

Public Types

enum **BTaggerOP**

Values:

enumerator **None**

enumerator **DL1r_FixedCutBEff_60**

enumerator **DL1r_FixedCutBEff_70**

enumerator **DL1r_FixedCutBEff_77**

enumerator DL1r_FixedCutBEff_85

enumerator DL1dv00_FixedCutBEff_60

enumerator DL1dv00_FixedCutBEff_70

enumerator DL1dv00_FixedCutBEff_77

enumerator DL1dv00_FixedCutBEff_85

enumerator DL1dv01_FixedCutBEff_60

enumerator DL1dv01_FixedCutBEff_70

enumerator DL1dv01_FixedCutBEff_77

enumerator DL1dv01_FixedCutBEff_85

enumerator GN120220509_FixedCutBEff_60

enumerator GN120220509_FixedCutBEff_70

enumerator GN120220509_FixedCutBEff_77

enumerator GN120220509_FixedCutBEff_85

enumerator DL1dv00_Continuous

enumerator DL1r_Continuous

enumerator DL1dv01_Continuous

enumerator GN120220509_Continuous

enumerator GN2v00LegacyWP_FixedCutBEff_60

enumerator GN2v00LegacyWP_FixedCutBEff_70

enumerator GN2v00LegacyWP_FixedCutBEff_77

enumerator GN2v00LegacyWP_FixedCutBEff_85 enumerator **GN2v00NewAliasWP_FixedCutBEff_60**

enumerator **GN2v00NewAliasWP_FixedCutBEff_70**

enumerator **GN2v00NewAliasWP_FixedCutBEff_77**

enumerator **GN2v00NewAliasWP_FixedCutBEff_85**

Public Functions

int **is_btag**(*BTaggerOP* op) const

const std::vector<float> &**SF_btag**(*BTaggerOP* op) const

void **muonInJetCorrection**(const xAH::*MuonContainer* *muons)

Public Members

float **rapidity**

int **isTrigMatched**

std::vector<int> **isTrigMatchedToChain**

std::string **listTrigChains**

float **Timing**

float **LArQuality**

float **HECQuality**

float **NegativeE**

float **AverageLArQF**

float **BchCorrCell**

float **N90Constituents**

float **LArBadHVEFrac**

int LArBadHVNCell

float ChargedFraction

float OotFracClusters5

float OotFracClusters10

float LeadingClusterPt

float LeadingClusterSecondLambda

float LeadingClusterCenterLambda

float LeadingClusterSecondR

int clean_passLooseBad

int clean_passLooseBadLLP

int clean_passLooseBadTrigger

int clean_passLooseBadTriggerUgly

int clean_passLooseBadUgly

int clean_passTightBad

int clean_passTightBadUgly

float HECFrac

float EMFrac

float CentroidR

float FracSamplingMax

float FracSamplingMaxIndex

float LowEtConstituentsFrac

float **GhostMuonSegmentCount**

float **Width**

float **NumTrkPt1000PV**

float **SumPtTrkPt1000PV**

float **TrackWidthPt1000PV**

float **NumTrkPt500PV**

float **SumPtTrkPt500PV**

float **TrackWidthPt500PV**

float **JVFPV**

float **Jvt**

float **JvtJvfcorr**

float **JvtRpt**

float **SumPtChargedPFOPt500PV**

float **fCharged**

float **JVC**

float **SV0**

float **SV1**

float **IP3D**

float **SV1IP3D**

float **COMBx**

float **DL1r**

float DL1r_pu

float DL1r_pc

float DL1r_pb

float DL1dv00

float DL1dv00_pu

float DL1dv00_pc

float DL1dv00_pb

float DL1dv01

float DL1dv01_pu

float DL1dv01_pc

float DL1dv01_pb

float GN1

float GN1_pu

float GN1_pc

float GN1_pb

float GN2v00LegacyWP

float GN2v00LegacyWP_pu

float GN2v00LegacyWP_pc

float GN2v00LegacyWP_pb

float GN2v00NewAliasWP

float GN2v00NewAliasWP_pu

float GN2v00NewAliasWP_pc

float GN2v00NewAliasWP_pb

int HadronConeExclTruthLabelID

int HadronConeExclExtendedTruthLabelID

float vtxOnlineValid

float vtxHadDummy

float bs_online_vx

float bs_online_vy

float bs_online_vz

float vtx_offline_x0

float vtx_offline_y0

float vtx_offline_z0

float vtx_online_x0

float vtx_online_y0

float vtx_online_z0

float vtx_online_bkg_x0

float vtx_online_bkg_y0

float vtx_online_bkg_z0

float JetFitter_nVTX

float JetFitter_nSingleTracks

float JetFitter_nTracksAtVtx

float JetFitter_mass

float JetFitter_energyFraction

float JetFitter_significance3d

float JetFitter_deltaeta

float JetFitter_deltaphi

float JetFitter_N2Tpar

float sv0_NGTinSvx

float sv0_N2Tpair

float sv0_massvx

float sv0_efracsvx

float sv0_normdist

float sv1_pu

float sv1_pb

float sv1_pc

float sv1_c

float sv1_cu

float sv1_NGTinSvx

float sv1_N2Tpair

float sv1_massvx

float sv1_efracsvx

float sv1_normdist

float **sv1_Lxy**

float **sv1_sig3d**

float **sv1_L3d**

float **sv1_distmatlay**

float **sv1_dR**

float **IP2D_pu**

float **IP2D_pb**

float **IP2D_pc**

float **IP2D**

float **IP2D_c**

float **IP2D_cu**

float **nIP2DTracks**

std::vector<float> **IP2D_gradeOfTracks**

std::vector<float> **IP2D_flagFromV0ofTracks**

std::vector<float> **IP2D_valD0wrtPVofTracks**

std::vector<float> **IP2D_sigD0wrtPVofTracks**

std::vector<float> **IP2D_weightBofTracks**

std::vector<float> **IP2D_weightCofTracks**

std::vector<float> **IP2D_weightUofTracks**

float **IP3D_pu**

float **IP3D_pb**

float **IP3D_pc**

float **IP3D_c**

float **IP3D_cu**

float **nIP3DTracks**

std::vector<float> **IP3D_gradeOfTracks**

std::vector<float> **IP3D_flagFromV0ofTracks**

std::vector<float> **IP3D_valD0wrtPVofTracks**

std::vector<float> **IP3D_sigD0wrtPVofTracks**

std::vector<float> **IP3D_valZ0wrtPVofTracks**

std::vector<float> **IP3D_sigZ0wrtPVofTracks**

std::vector<float> **IP3D_weightBofTracks**

std::vector<float> **IP3D_weightCofTracks**

std::vector<float> **IP3D_weightUofTracks**

int **is_DL1r_FixedCutBEff_60**

std::vector<float> **SF_DL1r_FixedCutBEff_60**

int **is_DL1r_FixedCutBEff_70**

std::vector<float> **SF_DL1r_FixedCutBEff_70**

int **is_DL1r_FixedCutBEff_77**

std::vector<float> **SF_DL1r_FixedCutBEff_77**

int **is_DL1r_FixedCutBEff_85**

std::vector<float> **SF_DL1r_FixedCutBEff_85** int **is_DL1dv00_FixedCutBEff_60**

std::vector<float> **SF_DL1dv00_FixedCutBEff_60**

int **is_DL1dv00_FixedCutBEff_70**

std::vector<float> **SF_DL1dv00_FixedCutBEff_70**

int **is_DL1dv00_FixedCutBEff_77**

std::vector<float> **SF_DL1dv00_FixedCutBEff_77**

int **is_DL1dv00_FixedCutBEff_85**

std::vector<float> **SF_DL1dv00_FixedCutBEff_85**

int **is_DL1dv01_FixedCutBEff_60**

std::vector<float> **SF_DL1dv01_FixedCutBEff_60**

int **is_DL1dv01_FixedCutBEff_70**

std::vector<float> **SF_DL1dv01_FixedCutBEff_70**

int **is_DL1dv01_FixedCutBEff_77**

std::vector<float> **SF_DL1dv01_FixedCutBEff_77**

int **is_DL1dv01_FixedCutBEff_85**

std::vector<float> **SF_DL1dv01_FixedCutBEff_85**

int **is_GN120220509_FixedCutBEff_60**

std::vector<float> **SF_GN120220509_FixedCutBEff_60**

int **is_GN120220509_FixedCutBEff_70**

std::vector<float> **SF_GN120220509_FixedCutBEff_70**

int **is_GN120220509_FixedCutBEff_77**

std::vector<float> **SF_GN120220509_FixedCutBEff_77**

int **is_GN120220509_FixedCutBEff_85**

std::vector<float> **SF_GN120220509_FixedCutBEff_85**

int **is_GN2v00LegacyWP_FixedCutBEff_60**

std::vector<float> **SF_GN2v00LegacyWP_FixedCutBEff_60**

int **is_GN2v00LegacyWP_FixedCutBEff_70**

std::vector<float> **SF_GN2v00LegacyWP_FixedCutBEff_70**

int **is_GN2v00LegacyWP_FixedCutBEff_77**

std::vector<float> **SF_GN2v00LegacyWP_FixedCutBEff_77**

int **is_GN2v00LegacyWP_FixedCutBEff_85**

std::vector<float> **SF_GN2v00LegacyWP_FixedCutBEff_85**

int **is_GN2v00NewAliasWP_FixedCutBEff_60**

std::vector<float> **SF_GN2v00NewAliasWP_FixedCutBEff_60**

int **is_GN2v00NewAliasWP_FixedCutBEff_70**

std::vector<float> **SF_GN2v00NewAliasWP_FixedCutBEff_70**

int **is_GN2v00NewAliasWP_FixedCutBEff_77**

std::vector<float> **SF_GN2v00NewAliasWP_FixedCutBEff_77**

int **is_GN2v00NewAliasWP_FixedCutBEff_85**

std::vector<float> **SF_GN2v00NewAliasWP_FixedCutBEff_85**

int **is_DL1r_Continuous**

std::vector<float> **SF_DL1r_Continuous** std::vector<float> **inEffSF_DL1r_Continuous**

int **is_DL1dv00_Continuous**

std::vector<float> **SF_DL1dv00_Continuous**

std::vector<float> **inEffSF_DL1dv00_Continuous**

int **is_DL1dv01_Continuous**

std::vector<float> **SF_DL1dv01_Continuous**

std::vector<float> **inEffSF_DL1dv01_Continuous**

int **is_GN120220509_Continuous**

std::vector<float> **SF_GN120220509_Continuous**

std::vector<float> **inEffSF_GN120220509_Continuous**

int **is_GN2v00LegacyWP_Continuous**

std::vector<float> **SF_GN2v00LegacyWP_Continuous**

std::vector<float> **inEffSF_GN2v00LegacyWP_Continuous**

int **is_GN2v00NewAliasWP_Continuous**

std::vector<float> **SF_GN2v00NewAliasWP_Continuous**

std::vector<float> **inEffSF_GN2v00NewAliasWP_Continuous**

int **ConeTruthLabelID**

int **TruthCount**

float **TruthLabelDeltaR_B**

float **TruthLabelDeltaR_C**

float **TruthLabelDeltaR_T**

int **PartonTruthLabelID**

float **GhostTruthAssociationFraction**

TLorentzVector **truth_p4**

double **charge**

char **passSel**

char **passOR**

const *Muon* ***matchedMuon** = nullptr

const *Jet* ***matchedJet** = nullptr

Class JetContainer

- Defined in file_xAODAnaHelpers_JetContainer.h

Nested Relationships

Nested Types

- *Struct JetContainer::btagOpPoint*

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Jet, HelperClasses::JetInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **JetContainer** : public xAH::*ParticleContainer*<*Jet*, *HelperClasses*::*JetInfoSwitch*>

Public Functions

JetContainer(const std::string &name = "jet", const std::string &detailStr = "", float units = 1e3, bool mc = false)

virtual **~JetContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillJet**(const xAOD::Jet *jet, const xAOD::Vertex *pv, int pvLocation)

virtual void **FillJet**(const xAOD::IParticle *particle, const xAOD::Vertex *pv, int pvLocation)

virtual void **updateParticle**(uint idx, *Jet* &jet)

Class L1JetContainer

- Defined in file_xAODAnaHelpers_L1JetContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Jet, HelperClasses::JetInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **L1JetContainer** : public xAH::*ParticleContainer*<*Jet*, *HelperClasses*::*JetInfoSwitch*>

Public Functions

L1JetContainer(const std::string &name = "L1Jet", float units = 1e3, bool mc = false)

virtual **~L1JetContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillLegacyL1Jets**(const xAOD::JetRoIContainer *jets, bool sort)

virtual void **updateParticle**(uint idx, *Jet* &jet)

template<typename **T**>
inline void **FillPhase1L1Jets**(*T* *&jets, bool sort)

Class MetContainer

- Defined in file_xAODAnaHelpers_MetContainer.h

Class Documentation

class **MetContainer**

Public Functions

MetContainer(const std::string &name = "met", const std::string &detailStr = "", float units = 1e3)

~MetContainer()

void **setTree**(TTree *tree)

void **setBranches**(TTree *tree)

void **clear**()

void **FillMET**(const xAOD::MissingETContainer *met)

template<typename **T_BR**>
void **connectBranch**(TTree *tree, std::string name, *T_BR* *variable)

template<typename **T_BR**>
void **setBranch**(TTree *tree, std::string name, *T_BR* *variable, std::string type)

Public Members

std::string **m_name**

HelperClasses::*METInfoSwitch* **m_infoSwitch**

bool **m_debug**

float **m_units**

float **m_metFinalClus**

float **m_metFinalClusPx**

float **m_metFinalClusPy**

float **m_metFinalClusPhi**

float **m_metFinalClusSumEt**

float **m_metFinalClusOverSqrtSumEt**

float **m_metFinalClusOverSqrtHt**

float **m_metFinalClusSignificance**

float **m_metFinalClusSigDirectional**

float **m_metFinalClusRho**

float **m_metFinalClusVarL**

float **m_metFinalClusVarT**

float **m_metFinalTrk**

float **m_metFinalTrkPx**

float **m_metFinalTrkPy**

float **m_metFinalTrkPhi**

float **m_metFinalTrkSumEt**

float **m_metFinalTrkOverSqrtSumEt**

float **m_metFinalTrkOverSqrtHt**

float **m_metFinalTrkSignificance**

float **m_metFinalTrkSigDirectional**

float **m_metFinalTrkRho**

float **m_metFinalTrkVarL**

float **m_metFinalTrkVarT**

float **m_metEle**

float **m_metEleSumEt**

float **m_metElePhi**

float **m_metGamma**

float **m_metGammaSumEt**

float **m_metGammaPhi**

float **m_metTau**

float **m_metTauSumEt**

float **m_metTauPhi**

float **m_metMuons**

float **m_metMuonsSumEt**

float **m_metMuonsPhi**

float **m_metJet**

float **m_metJetSumEt**

float **m_metJetPhi**

float **m_metJetTrk**

float **m_metJetTrkSumEt**

float **m_metJetTrkPhi**

float **m_metSoftClus**

float **m_metSoftClusSumEt**

float **m_metSoftClusPhi**

float **m_metSoftTrk**

float **m_metSoftTrkSumEt**

float **m_metSoftTrkPhi**

Class Muon

- Defined in file_xAODAnaHelpers_Muon.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **Muon** : public xAH::*Particle*

Public Functions

inline TLorentzVector **vec_eLoss()** const

Public Members

float **charge**

int **isTrigMatched**

std::vector<int> **isTrigMatchedToChain**

std::vector<std::string> **listTrigChains**

std::map<std::string, int> **isIsolated**

float **ptcone20**

float **ptcone30**

float **ptcone40**

float **ptvarcone20**

float **ptvarcone30**

float **ptvarcone40**

float **topoetcone20**

float **topoetcone30**

float **topoetcone40**

float **neflowisol20**

float **ptcone20_Nonprompt_All_MaxWeightTTVA_pt500**

float **ptcone20_Nonprompt_All_MaxWeightTTVA_pt1000**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVA_pt500**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVA_pt1000**

float **topoetcone20_CloseByCorr**

float **neflowisol20_CloseByCorr**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVA_pt500_CloseByCorr**

float **ptvarcone30_Nonprompt_All_MaxWeightTTVA_pt1000_CloseByCorr**

std::map<std::string, int> **quality**

std::map<std::string, std::vector<float>> **RecoEff_SF**

std::map<std::string, std::vector<float>> **IsoEff_SF**

std::map<std::string, std::vector<float>> **TrigEff_SF**

std::map<std::string, std::vector<float>> **TrigMCEff**

std::vector<float> **TTVAEff_SF**

float **trkd0**

float **trkd0sig**

float **trkz0**

float **trkz0sintheta**

float **trkphi0**

float **trktheta**

float **trkcharge**

float **trkqOverP**

int **trknSiHits**

int **trknPixHits**

int **trknPixHoles**

int **trknSCTHits**

int **trknSCTHoles**

int **trknTRTHits**

int **trknTRTHoles**

int **trknBLayerHits**

int **trknInnermostPixLayHits**

float **trkPixdEdX**

float **EnergyLoss**

float **EnergyLossSigma**

unsigned char **energyLossType**

float **MeasEnergyLoss**

float MeasEnergyLossSigma

float ParamEnergyLoss

float ParamEnergyLossSigmaMinus

float ParamEnergyLossSigmaPlus

float PromptLeptonInput_DL1mu

float PromptLeptonInput_DRlj

float PromptLeptonInput_LepJetPtFrac

float PromptLeptonInput_PtFrac

float PromptLeptonInput_PtRel

int PromptLeptonInput_TrackJetNTrack

float PromptLeptonInput_ip2

float PromptLeptonInput_ip3

float PromptLeptonInput_rnnip

int PromptLeptonInput_sv1_jf_ntrkv

float PromptLeptonIso

float PromptLeptonVeto

char isLRT

char passIDcuts

char passSel

char passOR

Class MuonContainer

- Defined in file_xAODAnaHelpers_MuonContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Muon, HelperClasses::MuonInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **MuonContainer** : public xAH::*ParticleContainer*<*Muon*, *HelperClasses*::*MuonInfoSwitch*>

Public Functions

MuonContainer(const std::string &name = "muon", const std::string &detailStr = "", float units = 1e3, bool mc = false, bool storeSystSFs = true)

virtual **~MuonContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillMuon**(const xAOD::Muon *muon, const xAOD::Vertex *primaryVertex)

virtual void **FillMuon**(const xAOD::IParticle *particle, const xAOD::Vertex *primaryVertex)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *Muon* &muon)

Class OnlineBeamSpotTool

- Defined in file_xAODAnaHelpers_OnlineBeamSpotTool.h

Nested Relationships

Nested Types

- *Struct OnlineBeamSpotTool::LBData*

Class Documentation

class **OnlineBeamSpotTool**

Public Types

enum **BSData**

Values:

enumerator **BSx**

enumerator **BSy**

enumerator **BSz**

Public Functions

OnlineBeamSpotTool()

~OnlineBeamSpotTool()

float **getOnlineBSInfo**(const xAOD::EventInfo *eventInfo, *BSData* datakey)

float **getOnlineBSInfo**(const xAH::*EventInfo* *eventInfo, *BSData* datakey)

float **getOnlineBSInfo**(int runNumber, int lumiBlock, bool isMC, *BSData* datakey)

Class Particle

- Defined in file_xAODAnaHelpers_Particle.h

Inheritance Relationships

Derived Types

- `public xAH::Cluster` (*Class Cluster*)
- `public xAH::Electron` (*Class Electron*)
- `public xAH::FatJet` (*Class FatJet*)

- public xAH::Jet (*Class Jet*)
- public xAH::Muon (*Class Muon*)
- public xAH::Photon (*Class Photon*)
- public xAH::Tau (*Class Tau*)
- public xAH::TrackPart (*Class TrackPart*)
- public xAH::TruthPart (*Class TruthPart*)

Class Documentation

class **Particle**

Subclassed by *xAH::Cluster*, *xAH::Electron*, *xAH::FatJet*, *xAH::Jet*, *xAH::Muon*, *xAH::Photon*, *xAH::Tau*, *xAH::TrackPart*, *xAH::TruthPart*

Public Functions

inline virtual **~Particle**()

Public Members

TLorentzVector **p4**

Template Class ParticleContainer

- Defined in file_xAODAnaHelpers_ParticleContainer.h

Class Documentation

template<class **T_PARTICLE**, class **T_INFOSWITCH**>

class **ParticleContainer**

Public Functions

inline **ParticleContainer**(const std::string &name, const std::string &detailStr = "", float units = 1e3, bool mc = false, bool useMass = false, bool storeSystSFs = true, const std::string &suffix = "")

inline virtual **~ParticleContainer**()

inline virtual void **setTree**(TTree *tree)

inline virtual void **setBranches**(TTree *tree)

inline virtual void **clear**()

inline virtual void **FillParticle**(const xAOD::IParticle *particle)

inline void **updateEntry**()

inline std::vector<*T_PARTICLE*> &**particles**()

inline *T_PARTICLE* &**at_nonConst**(uint idx)

inline const *T_PARTICLE* &**at**(uint idx) const

inline const *T_PARTICLE* &**operator[]**(uint idx) const

inline uint **size**() const

Public Members

T_INFOSWITCH **m_infoSwitch**

bool **m_mc**

bool **m_debug**

float **m_units**

bool **m_storeSystSFs**

int **m_n**

Protected Functions

inline std::string **branchName**(const std::string &varName)

template<typename **T_BR**>
inline void **connectBranch**(TTree *tree, const std::string &branch, std::vector<*T_BR*> **variable)

template<typename **T**>
inline void **setBranch**(TTree *tree, std::string varName, std::vector<*T*> *localVectorPtr)

template<typename **T**, typename **U**, typename **V**>
inline void **safeFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<*T*> &accessor, std::vector<*U*> *destination, *U* defaultValue, int units = 1)

template<typename **T**, typename **U**, typename **V**>
inline void **safeVecFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<std::vector<*T*>> &accessor, std::vector<std::vector<*U*>> *destination, int units = 1)

template<typename **T**, typename **V**>
inline void **safeSFVecFill**(const *V* *xAODObj, SG::AuxElement::ConstAccessor<std::vector<*T*>> &accessor, std::vector<std::vector<*T*>> *destination, const std::vector<*T*> &defaultValue)

inline virtual void **updateParticle**(uint idx, *T_PARTICLE* &particle)

Protected Attributes

std::string **m_name**

std::vector<*T_PARTICLE*> **m_particles**

Class Photon

- Defined in file_xAODAnaHelpers_Photon.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **Photon** : public xAH::*Particle*

Public Members

int **isIsolated_Cone40CaloOnly**

int **isIsolated_Cone40**

int **isIsolated_Cone20**

float **ptcone20**

float **ptcone30**

float **ptcone40**

float **ptvarcone20**

float **ptvarcone30**

float **ptvarcone40**

float **topoetcone20**

float **topoetcone30**

float **topoetcone40**

int **IsLoose**

int **IsMedium**

int **IsTight**

float **radhad1**

float **radhad**

float **e277**

float **reta**

float **rphi**

float **weta2**

float **f1**

float **wtot**

float **deltae**

float **eratio**

float **LooseEffSF**

float **MediumEffSF**

float **TightEffSF**

float **LooseEffSF_Error**

float **MediumEffSF_Error**

float **TightEffSF_Error**

std::vector<std::string> **trigMatched**

Class PhotonContainer

- Defined in file_xAODAnaHelpers_PhotonContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Photon, HelperClasses::PhotonInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **PhotonContainer** : public xAH::*ParticleContainer*<*Photon*, *HelperClasses*::*PhotonInfoSwitch*>

Public Functions

PhotonContainer(const std::string &name = "ph", const std::string &detailStr = "", float units = 1e3, bool mc = false)

virtual **~PhotonContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillPhoton**(const xAOD::Photon *photon)

virtual void **FillPhoton**(const xAOD::IParticle *particle)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *Photon* &photon)

Class Tau

- Defined in file_xAODAnaHelpers_Tau.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **Tau** : public xAH::*Particle*

Public Members

int **isTrigMatched**

std::vector<int> **isTrigMatchedToChain**

std::string **listTrigChains**

int **ntrk**

float **charge**

std::map<std::string, std::vector<float>> **TauEff_SF**

std::map<std::string, std::vector<float>> **TauTrigEff_SF**

int **isJetRNNSigVeryLoose**

int **isJetRNNSigLoose**

int **isJetRNNSigMedium**

int **isJetRNNSigTight**

float **JetRNNScore**

float **JetRNNScoreSigTrans**

int **isEleRNNLoose**

int **isEleRNNMedium**

int **isEleRNNTight**

float **EleRNNScore**

int **passEleOLR**

float **matchedJetWidth**

float **matchedJetJvt**

std::vector<float> **tracks_pt**

std::vector<float> **tracks_eta**

std::vector<float> **tracks_phi**

std::vector<int> **tracks_isCore**

std::vector<int> **tracks_isWide**

std::vector<int> **tracks_failTrackFilter**

std::vector<int> **tracks_passTrkSel**

std::vector<int> **tracks_isClCharged**

std::vector<int> **tracks_isClIso**

std::vector<int> **tracks_isClConv**

std::vector<int> **tracks_isClFake**

Class TauContainer

- Defined in file_xAODAnaHelpers_TauContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< Tau, HelperClasses::TauInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **TauContainer** : public xAH::*ParticleContainer*<*Tau*, *HelperClasses*::*TauInfoSwitch*>

Public Functions

TauContainer(const std::string &name = "tau", const std::string &detailStr = "", float units = 1e3, bool mc = false, bool storeSystSFs = true)

virtual **~TauContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillTau**(const xAOD::TauJet *tau)

virtual void **FillTau**(const xAOD::IParticle *particle)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *Tau* &tau)

Class TrackContainer

- Defined in file_xAODAnaHelpers_TrackContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< TrackPart, HelperClasses::TrackInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **TrackContainer** : public xAH::*ParticleContainer*<*TrackPart*, *HelperClasses*::*TrackInfoSwitch*>

Public Functions

TrackContainer(const std::string &name = "track", const std::string &detailStr = "", float units = 1e3)

virtual **~TrackContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillTrack**(const xAOD::TrackParticle *track)

virtual void **FillTrack**(const xAOD::IParticle *particle)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *TrackPart* &track)

Class TrackPart

- Defined in file_xAODAnaHelpers_TrackPart.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **TrackPart** : public xAH::*Particle*

Public Members

float **chiSquared**

float **d0**

std::vector<float> **definingParametersCovMatrix**

char **expectInnermostPixelLayerHit**

char **expectNextToInnermostPixelLayerHit**

float **numberDoF**

char **numberOfInnermostPixelLayerHits**

char **numberOfNextToInnermostPixelLayerHits**

char **numberOfPhiHoleLayers**

char **numberOfPhiLayers**

char **numberOfPixelDeadSensors**

char **numberOfPixelHits**

char **numberOfPixelHoles**

char **numberOfPixelSharedHits**

char **numberOfPrecisionHoleLayers**

char **numberOfPrecisionLayers**

char **numberOfSCTDeadSensors**

char **numberOfSCTHits**

char **numberOfSCTHoles**

char **numberOfSCTSharedHits**

char **numberOfTRTHits**

char **numberOfTRTOutliers**

float **phi**

float **qOverP**

float **theta**

Int_t **vertexLink**

UInt_t **vertexLink_persIndex**

UInt_t **vertexLink_persKey**

float **vz**

float **z0**

Class TruthContainer

- Defined in file_xAODAnaHelpers_TruthContainer.h

Inheritance Relationships

Base Type

- `public xAH::ParticleContainer< TruthPart, HelperClasses::TruthInfoSwitch >` (*Template Class ParticleContainer*)

Class Documentation

class **TruthContainer** : public xAH::*ParticleContainer*<*TruthPart*, *HelperClasses*::*TruthInfoSwitch*>

Public Functions

TruthContainer(const std::string &name = "truth", const std::string &detailStr = "", float units = 1e3)

virtual **~TruthContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillTruth**(const xAOD::TruthParticle *truth)

virtual void **FillTruth**(const xAOD::IParticle *particle)

inline void **setTree**(TTree *tree)

Protected Functions

virtual void **updateParticle**(uint idx, *TruthPart* &truth)

Class TruthPart

- Defined in file_xAODAnaHelpers_TruthPart.h

Inheritance Relationships

Base Type

- `public xAH::Particle` (*Class Particle*)

Class Documentation

class **TruthPart** : public xAH::*Particle*

Public Members

int **pdgId**

int **status**

int **barcode**

bool **is_higgs**

bool **is_bhad**

float **Bdecay_x**

float **Bdecay_y**

float **Bdecay_z**

int **nParents**

std::vector<int> **parent_pdgId**

std::vector<int> **parent_barcode**

std::vector<int> **parent_status**

int **nChildren**

std::vector<int> **child_pdgId**

std::vector<int> **child_barcode**

std::vector<int> **child_status**

float **pt_dressed**

float **eta_dressed**

float **phi_dressed**

float **e_dressed**

unsigned int **origin**

unsigned int **type**

Class VertexContainer

- Defined in file_xAODAnaHelpers_VertexContainer.h

Class Documentation

class **VertexContainer**

Public Functions

VertexContainer(const std::string &detailStr, const std::string &name = "vertex")

virtual **~VertexContainer**()

virtual void **setTree**(TTree *tree)

virtual void **setBranches**(TTree *tree)

virtual void **clear**()

virtual void **FillVertices**(const xAOD::VertexContainer *vertices)

virtual void **FillTruthVertices**(const xAOD::TruthVertexContainer *truthVertices)

inline std::string **branchName**(const std::string &varName)

template<typename **T_BR**>
inline void **connectBranch**(TTree *tree, const std::string &branch, std::vector<*T_BR*> **variable)

template<typename **T**>
inline void **setBranch**(TTree *tree, std::string varName, std::vector<*T*> *localVectorPtr)

Public Members

std::string **m_name**

Enums

Enum ContainerType

- Defined in file_xAODAnaHelpers_HelperClasses.h

Enum Documentation

enum class *HelperClasses*::**ContainerType**

Values:

enumerator **UNKNOWN**

enumerator **CONSTDV**

enumerator **CONSTCONT**

Enum ToolName

- Defined in file_xAODAnaHelpers_HelperClasses.h

Enum Documentation

enum class *HelperClasses*::**ToolName**

Values:

enumerator **MUONSELECTOR**

enumerator **ELECTRONSELECTOR**

enumerator **PHOTONSELECTOR**

enumerator **JETSELECTOR**

enumerator **BJETSELECTOR**

enumerator **CALIBRATOR**

enumerator **CORRECTOR**

enumerator **SELECTOR**

enumerator **DEFAULT**

Enum ShowerType

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Enum Documentation

enum *HelperFunctions*::**ShowerType**

The different supported shower types.

Values:

enumerator **Unknown**

enumerator **Pythia8**

enumerator **Herwig7**

enumerator **Sherpa21**

enumerator **Sherpa22**

enumerator **Sherpa2210**

Functions

Function ANA_MSG_ERROR

- Defined in file_xAODAnaHelpers_tools_ReturnCheckConfig.h

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_ERROR" with arguments "("Could not find the " "configuration file:"<<CONFIG.)". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 1] ("Could not find the " "configuration file:"<<CONFIG.) -^

Function ANA_MSG_HEADER

- Defined in file_xAODAnaHelpers_ParticlePIDManager.h

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_HEADER" with arguments "(msgPIDManager)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 31] ANA_MSG_HEADER (msgPIDManager) class ElectronLHPIDManager ——————————-^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 15] ANA_MSG_HEADER (msgPIDManager) class ElectronLHPIDManager —————^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 29] ANA_MSG_HEADER (msgPIDManager) class ElectronLHPIDManager —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 29] ANA_MSG_HEADER (msgPIDManager) class ElectronLHPIDManager —————————^

Function ANA_MSG_SOURCE(msgClusterHists, "ClusterHists")

- Defined in file_Root_ClusterHists.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgClusterHists, "ClusterHists")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 18] (msgClusterHists, "ClusterHists") ——————^

Function ANA_MSG_SOURCE(msgElectronHists, "ElectronHists")

- Defined in file_Root_ElectronHists.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgElectronHists, "ElectronHists")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 19] (msgElectronHists, "ElectronHists") ——————-^

Function ANA_MSG_SOURCE(msgPIDManager, "PIDManager")

- Defined in file_Root_ParticlePIDManager.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgPIDManager, "PIDManager")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 16] (msgPIDManager, "PIDManager") ———————-^

Function ANA_MSG_SOURCE(msgPhotonHists, "PhotonHists")

- Defined in file_Root_PhotonHists.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgPhotonHists, "PhotonHists")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 17] (msgPhotonHists, "PhotonHists") ———————-^

Function ANA_MSG_SOURCE(msgTrackHists, "TrackHists")

- Defined in file_Root_TrackHists.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgTrackHists, "TrackHists")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 16] (msgTrackHists, "TrackHists") ———————-^

Function ANA_MSG_SOURCE(msgTracksInJetHists, "TracksInJetHists")

- Defined in file_Root_TracksInJetHists.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgTracksInJetHists, "TracksInJetHists")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 22] (msgTracksInJetHists, "TracksInJetHists") ——————————-^

Function ANA_MSG_SOURCE(msgVtxHists, "VtxHists")

- Defined in file_Root_VtxHists.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ANA_MSG_SOURCE" with arguments "(msgVtxHists, "VtxHists")". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 14] (msgVtxHists, "VtxHists") ————^

Function ClassImp(xAH::Algorithm)

- Defined in file_Root_Algorithm.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(xAH::Algorithm)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(BasicEventSelection)

- Defined in file_Root_BasicEventSelection.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(BasicEventSelection)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(BJetEfficiencyCorrector)

- Defined in file_Root_BJetEfficiencyCorrector.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(BJetEfficiencyCorrector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(ClusterHistsAlgo)

- Defined in file_Root_ClusterHistsAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(ClusterHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(DebugTool)

- Defined in file_Root_DebugTool.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(DebugTool)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(ElectronCalibrator)

- Defined in file_Root_ElectronCalibrator.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(ElectronCalibrator)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(ElectronEfficiencyCorrector)

- Defined in file_Root_ElectronEfficiencyCorrector.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(ElectronEfficiencyCorrector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(ElectronHistsAlgo)

- Defined in file_Root_ElectronHistsAlgo.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(ElectronHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to

member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(ElectronSelector)

- Defined in file_Root_ElectronSelector.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(ElectronSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(HLTJetGetter)

- Defined in file_Root_HLTJetGetter.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(HLTJetGetter)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(IParticleHistsAlgo)

- Defined in file_Root_IParticleHistsAlgo.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(IParticleHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(IsoCloseByCorr)

- Defined in file_Root_IsoCloseByCorr.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(IsoCloseByCorr)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(JetCalibrator)

- Defined in file_Root_JetCalibrator.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(JetCalibrator)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(JetHistsAlgo)

- Defined in file_Root_JetHistsAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(JetHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(JetSelector)

- Defined in file_Root_JetSelector.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(JetSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(MessagePrinterAlgo)

- Defined in file_Root_MessagePrinterAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MessagePrinterAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers

If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(METConstructor)

- Defined in file_Root_METConstructor.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(METConstructor)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(MetHistsAlgo)

- Defined in file_Root_MetHistsAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MetHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(MinixAOD)

- Defined in file_Root_MinixAOD.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MinixAOD)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ————^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(MuonCalibrator)

- Defined in file_Root_MuonCalibrator.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MuonCalibrator)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ————^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(MuonEfficiencyCorrector)

- Defined in file_Root_MuonEfficiencyCorrector.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MuonEfficiencyCorrector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ————^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(MuonHistsAlgo)

- Defined in file_Root_MuonHistsAlgo.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MuonHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(MuonInFatJetCorrector)

- Defined in file_Root_MuonInFatJetCorrector.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MuonInFatJetCorrector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH —————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH —————————^

Function ClassImp(MuonSelector)

- Defined in file_Root_MuonSelector.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(MuonSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to

member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(OverlapRemover)

- Defined in file_Root_OverlapRemover.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(OverlapRemover)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(PhotonCalibrator)

- Defined in file_Root_PhotonCalibrator.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(PhotonCalibrator)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(PhotonHistsAlgo)

- Defined in file_Root_PhotonHistsAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(PhotonHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(PhotonSelector)

- Defined in file_Root_PhotonSelector.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(PhotonSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TauCalibrator)

- Defined in file_Root_TauCalibrator.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TauCalibrator)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TauEfficiencyCorrector)

- Defined in file_Root_TauEfficiencyCorrector.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TauEfficiencyCorrector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ———————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ———————^

Function ClassImp(TauJetMatching)

- Defined in file_Root_TauJetMatching.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TauJetMatching)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ———————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ———————^

Function ClassImp(TauSelector)

- Defined in file_Root_TauSelector.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TauSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to

member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TrackHistsAlgo)

- Defined in file_Root_TrackHistsAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TrackHistsAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TrackSelector)

- Defined in file_Root_TrackSelector.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TrackSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TreeAlgo)

- Defined in file_Root_TreeAlgo.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TreeAlgo)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TrigMatcher)

- Defined in file_Root_TrigMatcher.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TrigMatcher)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(TruthSelector)

- Defined in file_Root_TruthSelector.cxx

Function Documentation

Warning: doxygenfunction: Unable to resolve function "ClassImp" with arguments "(TruthSelector)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ——————————^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Function ClassImp(Writer)

- Defined in file_Root_Writer.cxx

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "ClassImp" with arguments "(Writer)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Invalid C++ declaration: Expected end of definition or ;. [error at 26] ClassImp (xAH::Algorithm) xAH ————————–^ If the function has a return type: Error in declarator If declarator-id with parameters-and-qualifiers: Invalid C++ declaration: Expected identifier in nested name. [error at 9] ClassImp (xAH::Algorithm) xAH ———^ If parenthesis in noptr-declarator: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 24] ClassImp (xAH::Algorithm) xAH ————————^ If declarator-id: Invalid C++ declaration: Expecting "(" in parameters-and-qualifiers. [error at 24] ClassImp (xAH::Algorithm) xAH ————————^

Template Function HelperFunctions::__attribute__

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

> **Warning:** doxygenfunction: Unable to resolve function "HelperFunctions::__attribute__" with arguments "((deprecated("retrieve<T>(. . . , bool) is deprecated. See https://github.com/UCATLAS/xAODAnaHelpers/pull/882")))". Could not parse arguments. Parsing eror is Invalid C++ declaration: Expected identifier in nested name. [error at 1] ((deprecated("retrieve<T>(. . . , bool) is deprecated. See https://github.com/UCATLAS/xAODAnaHelpers/pull/882"))) -^

Function HelperFunctions::applyPrimaryVertexSelection

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

bool *HelperFunctions*::**applyPrimaryVertexSelection**(const xAOD::JetContainer *jets, const xAOD::VertexContainer *vertices)

Template Function HelperFunctions::connectBranch

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T_BR**>
void *HelperFunctions*::**connectBranch**(std::string name, TTree *tree, const std::string &branch, std::vector<*T_BR*> **variable)

Function HelperFunctions::countPrimaryVertices

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

int *HelperFunctions*::**countPrimaryVertices**(const xAOD::VertexContainer *vertexContainer, int Ntracks = 2)

Function HelperFunctions::dPhi

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

float *HelperFunctions*::**dPhi**(float phi1, float phi2)

Function HelperFunctions::found_non_dummy_sys

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

inline bool *HelperFunctions*::**found_non_dummy_sys**(std::vector<std::string> *sys_list)

Template Function HelperFunctions::getLink

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<class **T**>
const *T* **HelperFunctions*::**getLink**(const xAOD::IParticle *particle, std::string name)

Access to element link to object of type T stored in auxdata.

Function HelperFunctions::getListofSystematics

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

std::vector<CP::SystematicSet> *HelperFunctions*::**getListofSystematics**(const CP::SystematicSet inSysts, std::string systNames, float systVal, MsgStream &msg)

Get a list of systematics.

Parameters

- **inSysts** – systematics set retrieved from the tool
- **systNames** – comma separated list of wanted systematics names, use "Nominal" for nominal and "All" for all systematics
- **systVal** – continuous systematics sigma value
- **msg** – the MsgStream object with appropriate level for debugging

Function HelperFunctions::getMCShowerType

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

ShowerType *HelperFunctions*::**getMCShowerType**(const std::string &sample_name)

Determines the type of generator used for the shower from the sample name.

The name of the generator is determined using some common definitions in the ATLAS MC dataset naming scheme. The case independent strings that are searched for are:

PYTHIA8EVTGEN or Py8EG or PYTHIA : Pythia8 HERWIG : Herwig7 SHERPA_CT : Sherpa21 SHERPA : Sherpa22 (if not Sherpa 21)

Parameters **sample_name** – The name of the sample, usualy the dataset name

Function HelperFunctions::getPrimaryVertex(const xAOD::VertexContainer *, MsgStream&)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

const xAOD::Vertex **HelperFunctions*::**getPrimaryVertex**(const xAOD::VertexContainer *vertexContainer, MsgStream &msg)

Function HelperFunctions::getPrimaryVertex(const xAOD::VertexContainer *)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

inline const xAOD::Vertex **HelperFunctions*::**getPrimaryVertex**(const xAOD::VertexContainer *vertexContainer)

Function HelperFunctions::getPrimaryVertexLocation(const xAOD::VertexContainer *, MsgStream&)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

int *HelperFunctions*::**getPrimaryVertexLocation**(const xAOD::VertexContainer *vertexContainer, MsgStream &msg)

Function HelperFunctions::getPrimaryVertexLocation(const xAOD::VertexContainer *)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

inline int *HelperFunctions*::**getPrimaryVertexLocation**(const xAOD::VertexContainer *vertexContainer)

Function HelperFunctions::getPrimaryVertexZ

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

float *HelperFunctions*::**getPrimaryVertexZ**(const xAOD::Vertex *pvx)

Function HelperFunctions::has_exact

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

bool *HelperFunctions*::**has_exact**(const std::string input, const std::string flag)

Template Function HelperFunctions::isAvailable(std::string, xAOD::TEvent *, xAOD::TStore *, MsgStream&)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
bool *HelperFunctions*::**isAvailable**(std::string name, xAOD::TEvent *event, xAOD::TStore *store, MsgStream &msg)

Return true if an arbitrary object from TStore / TEvent is available.

This tries to make your life simple by providing a one-stop container check shop for all types

Example Usage:

```
const xAOD::JetContainer  jets(0);
// look for "AntiKt10LCTopoJets" in both TEvent and TStore
HelperFunctions::isAvailable<xAOD::JetContainer>("AntiKt10LCTopoJets", m_event, m_
↪store)
// look for "AntiKt10LCTopoJets" in only TStore
HelperFunctions::isAvailable<xAOD::JetContainer>("AntiKt10LCTopoJets", 0, m_store)
// look for "AntiKt10LCTopoJets" in only TEvent, enable verbose output
HelperFunctions::isAvailable<xAOD::JetContainer>("AntiKt10LCTopoJets", m_event, 0,␣
↪MSG::VERBOSE)
```

Parameters

- **name** – the name of the object to look up
- **event** – the TEvent, usually wk()->xaodEvent(). Set to 0 to not search TEvent.

- **store** – the TStore, usually wk()->xaodStore(). Set to 0 to not search TStore.
- **msg** – the MsgStream object with appropriate level for debugging

Template Function HelperFunctions::isAvailable(std::string, xAOD::TEvent *, xAOD::TStore *)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
bool *HelperFunctions*::**isAvailable**(std::string name, xAOD::TEvent *event, xAOD::TStore *store)

Function HelperFunctions::isAvailableMetaData

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

StatusCode *HelperFunctions*::**isAvailableMetaData**(TTree *metaData)

Function HelperFunctions::isFilePrimaryxAOD

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

bool *HelperFunctions*::**isFilePrimaryxAOD**(TFile *inputFile)

Function HelperFunctions::jetReclustering

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

std::vector<TLorentzVector> *HelperFunctions*::**jetReclustering**(const xAOD::JetContainer *jets, double radius = 1.0, double fcut = 0.05, fastjet::JetAlgorithm rc_alg = fastjet::antikt_algorithm)

Function HelperFunctions::jetTrimming(const xAOD::JetContainer *, double, double, fastjet::JetAlgorithm)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

std::vector<TLorentzVector> *HelperFunctions*::**jetTrimming**(const xAOD::JetContainer *jets, double radius = 0.3, double fcut = 0.05, fastjet::JetAlgorithm s_alg = fastjet::kt_algorithm)

Function HelperFunctions::jetTrimming(const xAOD::Jet *, double, double, fastjet::JetAlgorithm)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

TLorentzVector *HelperFunctions*::**jetTrimming**(const xAOD::Jet *jet, double radius = 0.3, double fcut = 0.05, fastjet::JetAlgorithm s_alg = fastjet::kt_algorithm)

Template Function HelperFunctions::makeDeepCopy

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T1**, typename **T2**, typename **T3**>
StatusCode *HelperFunctions*::**makeDeepCopy**(xAOD::TStore *m_store, std::string containerName, const *T1* *cont)

Make a deep copy of a container and put it in the TStore.

This is a very powerful templating function. The point is to remove the triviality of making deep copies by specifying all that is needed. The best way is to demonstrate via example:

```
const xAOD::JetContainer  selected_jets(nullptr);
ANA_CHECK( m_event->retrieve( selected_jets, "SelectedJets" ));
ANA_CHECK( (HelperFunctions::makeDeepCopy<xAOD::JetContainer, xAOD::JetAuxContainer,
↪ xAOD::Jet>(m_store, "BaselineJets", selected_jets)));
```

Template Parameters

- **T1** – The type of the container you’re going to deep copy into
- **T2** – The type of the aux container you’re going to deep copy into
- **T3** – The type of the object inside the container you’re going to deep copy

Parameters

- **m_store** – A pointer to the TStore object
- **containerName** – The name of the container to create as output in the TStore
- **cont** – The container to deep copy, it should be a container of pointers (IParticleContainer or ConstDataVector)

Template Function HelperFunctions::makeSubsetCont(T1 *&, T2 *&, MsgStream&, const std::string&, HelperClasses::ToolName)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T1**, typename **T2**>
StatusCode *HelperFunctions*::**makeSubsetCont**(*T1* *&intCont, *T2* *&outCont, MsgStream &msg, const std::string &flagSelect = "", *HelperClasses*::*ToolName* tool_name = *HelperClasses*::*ToolName*::*DEFAULT*)

Function to copy a subset of a generic input xAOD container into a generic output xAOD container.

If the optional parameters aren't specified, the function will just make a full copy of the input container into the output one.

Author Marco Milesi (marco.milesi@cern.ch)

Parameters

- **intCont** – **[in]** input container
- **outCont** – **[inout]** output container
- **flagSelect** – **[in]** (optional) the name of the decoration for objects passing a certain selection (e.g. "passSel", "overlaps" …). When explicitly specified, it must not be empty.
- **tool_name** – **[in]** (optional) an enum specifying the tool type which is calling this function (definition in `HelperClasses::ToolName`)

Template Function HelperFunctions::makeSubsetCont(T1 *&, T2 *&, const std::string&, HelperClasses::ToolName)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T1**, typename **T2**>
StatusCode *HelperFunctions*::**makeSubsetCont**(*T1* *&intCont, *T2* *&outCont, const std::string &flagSelect = "", *HelperClasses*::*ToolName* tool_name = *HelperClasses*::*ToolName*::*DEFAULT*)

Function HelperFunctions::msg

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

MsgStream &*HelperFunctions*::**msg**(MSG::Level lvl = MSG::INFO)
Static object that provides athena-based message logging functionality

Function HelperFunctions::passPrimaryVertexSelection

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

bool *HelperFunctions*::**passPrimaryVertexSelection**(const xAOD::VertexContainer *vertexContainer, int Ntracks = 2)

Template Function HelperFunctions::recordOutput

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T1**, typename **T2**>
StatusCode *HelperFunctions*::**recordOutput**(xAOD::TEvent *m_event, xAOD::TStore *m_store, std::string containerName)

Copy a container from the TStore to be recorded in the TEvent (eg: to an output)

If you have a container in the TStore, this function will record it into the output for you without an issue. As an example:

```
ANA_CHECK( HelperFunctions::recordOutput<xAOD::JetContainer, xAOD::JetAuxContainer>
↪(m_event, m_store, "BaselineJets"));
```

where we build off the previous example of making a deep copy (see `HelperFunctions::makeDeepCopy()`).

Template Parameters

- **T1** – The type of the container you're going to record
- **T2** – The type of the aux container you're going to record

Parameters

- **m_event** – A pointer to the TEvent object
- **m_store** – A pointer to the TStore object
- **containerName** – The name of the container in the TStore to record to TEvent

Template Function HelperFunctions::remove_duplicates

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
void *HelperFunctions*::**remove_duplicates**(std::vector<*T*> &vec)

Function HelperFunctions::replaceString

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

std::string *HelperFunctions*::**replaceString**(std::string subjet, const std::string &search, const std::string &replace)

Template Function HelperFunctions::retrieve(T *&, std::string, xAOD::TEvent *, xAOD::TStore *, MsgStream&)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
StatusCode *HelperFunctions*::**retrieve**(*T* *&cont, std::string name, xAOD::TEvent *event, xAOD::TStore *store, MsgStream &msg)

Retrieve an arbitrary object from TStore / TEvent.

This tries to make your life simple by providing a one-stop container retrieval shop for all types.

Example Usage:

```
const xAOD::JetContainer  jets(0);
// look for "AntiKt10LCTopoJets" in both TEvent and TStore
ANA_CHECK( HelperFunctions::retrieve(jets, "AntiKt10LCTopoJets", m_event, m_store)␣
↪);
// look for "AntiKt10LCTopoJets" in only TStore
ANA_CHECK( HelperFunctions::retrieve(jets, "AntiKt10LCTopoJets", 0, m_store) );
// look for "AntiKt10LCTopoJets" in only TEvent, enable verbose output
ANA_CHECK( HelperFunctions::retrieve(jets, "AntiKt10LCTopoJets", m_event, 0, msg())␣
↪);
```

Checking Order:

- start by checking TStore
  - check if store contains 'xAOD::JetContainer' named 'name'
    - attempt to retrieve from store
    - return if failure
- next check TEvent
  - check if event contains 'xAOD::JetContainer' named 'name'
    - attempt to retrieve from event
    - return if failure
  - return FAILURE
- return SUCCESS (should never reach this last line)

Parameters

- **cont** – pass in a pointer to the object to store the retrieved container in
- **name** – the name of the object to look up
- **event** – the TEvent, usually wk()->xaodEvent(). Set to 0 to not search TEvent.
- **store** – the TStore, usually wk()->xaodStore(). Set to 0 to not search TStore.
- **msg** – the MsgStream object with appropriate level for debugging

Template Function HelperFunctions::retrieve(T *&, std::string, xAOD::TEvent *, xAOD::TStore *)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
StatusCode *HelperFunctions*::**retrieve**(*T* *&cont, std::string name, xAOD::TEvent *event, xAOD::TStore *store)

Template Function HelperFunctions::sort_container_pt(T *)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
T HelperFunctions::**sort_container_pt**(*T* *inCont)

Template Function HelperFunctions::sort_container_pt(const T *)

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
const *T HelperFunctions*::**sort_container_pt**(const *T* *inCont)

Function HelperFunctions::sort_pt

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

bool *HelperFunctions*::**sort_pt**(const xAOD::IParticle *partA, const xAOD::IParticle *partB)

Function HelperFunctions::SplitString

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

std::vector<TString> *HelperFunctions*::**SplitString**(TString &orig, const char separator)

Function HelperFunctions::string_pos

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

std::size_t *HelperFunctions*::**string_pos**(const std::string &haystack, const std::string &needle, unsigned int N)

Function which returns the position of the n-th occurence of a character in a string searching backwards. Returns -1 if no occurencies are found.

Source: http://stackoverflow.com/questions/18972258/index-of-nth-occurrence-of-the-string

Template Function HelperFunctions::type_name

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

template<typename **T**>
std::string *HelperFunctions*::**type_name**(bool useXAOD = true)

Function HelperFunctions::writeSystematicsListHist

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

void *HelperFunctions*::**writeSystematicsListHist**(const std::vector<CP::SystematicSet> &systs, std::string histName, TFile *file)

Function xAH::addRucio

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Function Documentation

void xAH::**addRucio**(SH::SampleHandler &sh, const std::string &name, const std::string &dslist)

Directly add a SampleGrid to a SamplerHandler listing several datasets.

Parameters

- **sh** – SampleHander to which the sample will be added to
- **name** – Name of the sample
- **list** – List of datasets to be included in the sample

Variables

Variable else

- Defined in file_xAODAnaHelpers_tools_ReturnCheckConfig.h

Variable Documentation

```
else {ANA_MSG_INFO( "Found configuration file: " << CONFIG.c_str())
```

Variable HelperFunctions::debug

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Variable Documentation

```
StatusCode std::string xAOD::TEvent xAOD::TStore bool HelperFunctions::debug = { return retrieve<T>(c
name, event, store, msg())
```

Variable HelperFunctions::event

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Variable Documentation

```
StatusCode std::string xAOD::TEvent * HelperFunctions::event
```

Variable HelperFunctions::name

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Variable Documentation

```
StatusCode std::string HelperFunctions::name
```

Variable HelperFunctions::store

- Defined in file_xAODAnaHelpers_HelperFunctions.h

Variable Documentation

`StatusCode std::string xAOD::TEvent xAOD::TStore * HelperFunctions::store`

Defines

Define EL_RETURN_CHECK

- Defined in file_xAODAnaHelpers_tools_ReturnCheck.h

Define Documentation

EL_RETURN_CHECK(CONTEXT, EXP)

Define RETURN_CHECK

- Defined in file_xAODAnaHelpers_tools_ReturnCheck.h

Define Documentation

RETURN_CHECK(CONTEXT, EXP, INFO)

Typedefs

Typedef floatAccessor

- Defined in file_xAODAnaHelpers_ClusterContainer.h

Typedef Documentation

typedef SG::AuxElement::Accessor<std::vector<float>> **floatAccessor**

Typedef floatAccessor

- Defined in file_xAODAnaHelpers_HelpTreeBase.h

Typedef Documentation

typedef SG::AuxElement::Accessor<std::vector<float>> **floatAccessor**

Typedef floatAccessor

- Defined in file_xAODAnaHelpers_PhotonContainer.h

Typedef Documentation

typedef SG::AuxElement::Accessor<std::vector<float>> **floatAccessor**

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- search

INDEX

B

C

D

E

F

H

I

J

M

O

P

R

T

W

X