TopxAODStartGuideR21

Not yet Certified as ATLAS Documentation

Introduction

AnalysisTop is a general purpose xAOD analysis framework. It is highly configurable and well supported, with a strong team of liaisons within the top reconstruction group who are plugged into every CP group. These liaisons have a responsibility to maintain the code and ensure that the latest CP recommendations are implemented. AnalysisTop will calibrate all physics objects (including systematic variations), apply user-defined physics object cuts, apply overlap removal and then apply user-defined event level cuts. It is able to save to flat ntuples (xAOD in the future), with users able to define the content of the output.

Historically, AnalysisTop was built as a standalone release, but since December 2019 it became part of the AnalysisBase release (since release 21.2.98).

From the Release 21 release page it is possible to get the list of most relevant recent changes to the AT code, otherwise you can see here full list of merge requests. The list of packages compiled in AnalysisBase are found in the package filter file for the project. It takes a long time to compile all of those locally and on the grid, so pre-compiled versions are regularly build and deployed on cvmfs, making AnalysisBase accessible to any cvmfs connected computer. This means that it is available on all grid nodes. Additionally, all nightly builds are installed in cvmfs and available at grid sites. AnalysisTop has a dedicated JIRA, please report any bugs and feature requests here. Source code can be browsed in Git. Updates to AnalysisTop can be followed by subscribing to the AnalysisTop labels.

This TWiki described how to use AnalysisTop without modifying any central athena package. If you are interested in modifying AnalysisTop packages, have a look at the AnalysisTopGuideToUpdatingAthena twiki page. You can however extend Analysistop without touching the central code, for example by adding a Custom Saver which saves some additional variables. This is explained below in this page or in the tutorials. List of AnalysisTop tutorial can be found at AnalysisTopTutorials: the TopRun2WorkshopPreTutorial2020 is very detailed and can be very helpful for newcomers.

Announcement: problem with the grid submission scripts

If you're using the grid submission scripts provided with AnalysisTop (see below or in the tutorial), you may see an errore like:

```
DANGER DANGER DANGER
Could not find rucio.client.Client. If you use setupATLAS (you should) then
"localSetupRucioClients" and run this again
```

This is due to a recent change in rucio, unfortunately making the old script not usable. **The simplest solution is to use releases >=21.2.125 (which will become available soon), where the error is solved**.

If you need to keep using a previous release, you'll need a workaround to be able to still use the scripts. We list here some possibilities.

Option A: copy the submission scripts locally

In the folder where you have the python scripts you use for the submission (e.g. `01SubmitToGrid_Tutorial.py`), do "cp -r /cvmfs/atlas.cern.ch/repo/sw/software/21.2/AnalysisBase/21.2.125/InstallArea/x86_64-centos7-gcc8-opt/python/TopExamples/ ." to copy the most recent version of the central python scripts locally.

This is the simplest solution, but you'll have to check in the future if new changes to the submission scripts happen in the AnalysisBase release, to make sure you're always using up-to-date submission scripts.

Option B: Solving the issue for older releases with a sparse checkout

You'll need a working area prepared with a sparse checkout, following e.g. the instructions here but checking out just the TopExamples package. There are two different cases below:

If you don't have any local package (i.e. you never did a sparse checkout and you don't have an athena folder)

First of all, if you never forked athena, do that (see twiki, follow also the Add the ATLAS Robot step).

Then, prepare a backup of your working directory just in case any mistake happens in this procedure.

Then, inside your working directory, you'll need to do:

```
setupATLAS
lsetup git
git atlas init-workdir -b 21.2 https://:@gitlab.cern.ch:8443/atlas/athena.git #this will create a folder called "athena/"
cd athena
git atlas addpkg TopExamples
git checkout release/21.2.XXX #use the release you prefer
git checkout upstream/21.2 PhysicsAnalysis/TopPhys/xAOD/TopExamples
```

If you have a Custom Saver e.g. in the `source/` folder, move it to the `athena/`. You should be now able to remove the `source/` folder safely (check carefully if you still have something there before doing that though). then remove the build directory you have already, and do:

```
cd build
asetup AnalysisBase,21.2.XXX,here
```

then inside the `build` directory you can compile with:

```
cmake ../athena/Projects/WorkDir
cmake --build ./
source */setup.sh
```

If you already have something in the sparse checkout (you have already some packages in the athena folder)

You'll need to add TopExamples in you athena folder; from that folder, do:

```
setupATLAS
lsetup git
git atlas addpkg TopExamples
#git checkout release/21.2.XXX #you should have this already set, so this shouldn't be needed in principle, unless you want to change the release you're using
git checkout upstream/21.2 PhysicsAnalysis/TopPhys/xAOD/TopExamples
```

Then remove the build directory, recreate it and recompile as usual (as described above for the other case).

1 First time setup of AnalysisTop

Following instructions are provided for one of the most up-to-date version of AnalysisTop, but please verify on the release notes of any additional package that might be required to fix most recently discovered bugs. Login to lxplus.cern.ch or any cvmfs machine you have at you local institute. Full inofrmation on all suggested method to setup AnalysisTop are on the SettingUpAnalysisTop twiki page.

important: starting from release 21.2.98, you have to setup AnalysisBase instead of AnalysisTop

First create the directory structure

```
mkdir MyAnalysis
cd MyAnalysis
mkdir build source
```

Then do the basic ATLAS setup

```
setupATLAS
lsetup git
```

Setup the analysis release you want in the source directory

```
cd source
asetup AnalysisBase,21.2.XXX,here #adjust the release number to the one you want to use, you can find here https://twiki.cern.ch/twiki/bin/view/AtlasProtected/Analy
```

You should notice a file called CMakeLists.txt has been created in the directory. Go back to the home directory:

```
cd ..
```

Then open a file called `setup.sh` and add the following lines to that:

```
setupATLAS --quiet
lsetup git
cd source/
asetup --restore
cd ../build
source */setup.sh
cd ..
#lsetup panda
#lsetup pyami
#voms-proxy-init --voms atlas
```

(note: rucio cannot be setup together with an AnalysisRelease, if you need that you'll need to work in a separate terminal/shell)

this will be used the next times you will start working in this area (uncomment the last lines if you plan to use the GRID and if you have a valid certificate for that).

Let's prepare some compilation scripts. Go to the work directory MyAnalysis ("cd .." if you're still inside source/ from the previous step).

Open a file called `full_compile.sh` and add the following lines:

```
cd build
cmake ../source
cmake --build ./
source */setup.sh
cd ..
```

and then give the command:

```
source full_compile.sh
```

You can also prepare another script called `quick_compile.sh`, with the following lines

```
cd build
cmake --build ./
source */setup.sh
cd ..
```

You will have to use the `source full_compile.sh` for the first compilation and every time you add new classes or change package dependencies; if you just modify the code of existing files, you can just do `source quick_compile.sh`.

Each time you open a new shell you just need to repeat these steps

```
cd MyAnalysis #or any path to go to the workdir you created
source setup.sh
```

If you want to change the release you're using

Let's say you started using a given release, but you want to change it at some point. Start with a fresh shell, it's always a good idea. Go in the source/ folder and setup the new release you want to use:

```
cd MyAnalysis/source #or whatever the path is to the source/ folder of your working directory
setupATLAS --quiet
asetup AnalysisBase,21.2.YYY
cd..
```

then remove your build directory and recreate it, then do a full recompilation, finally do the usual setup:

```
rm -rf build/
mkdir build
source full_compile.sh
source setup.sh
```

If your setup becomes messed up

If you see weird linking problems or other strange errors, one of the first quick tests you can do is removing the build directory and re-creating it:

```
rm -rf build/
mkdir build
source full_compile.sh
source setup.sh
```

2 . Running locally

2.1 Make a flat ntuple containing only calibrated and selected objects

The main program is top-xaod and if everything is correctly setup, it should be already available in your path. It takes the input and applies the CP corrections and object selection. After that it does the event-level cuts specified in the configuration file, with only calibrated and selected objects written out, and only for those events that pass the selection.

It will also calculate scale factors for electrons, muons and b-tagging. The nominal scale factors are written for each 4-momentum shifting systematic, while for the nominal calibration you also get the various scale factors systematics (ID up/down etc etc). More information can be found in the corresponding CP TWiki pages (start here).

2.2 Test input files

If you want to test the code, have a look at the list of recommended top samples MC16 samples. Pick a dataset you want to use in your test run. e.g this ttbar nonallhad dataset.

```
mc16_13TeV.410470.PhPy8EG_A14_ttbar_hdamp258p75_nonallhad.deriv.DAOD_TOPQ1.e6337_s3126_r9364_p3629
```

then download one file

```
rucio download --nrandom 1 mc16_13TeV.410470.PhPy8EG_A14_ttbar_hdamp258p75_nonallhad.deriv.DAOD_TOPQ1.e6337_s3126_r9364_p3629
```

and save the downloaded file path to a txt file, e.g. test_file_mc16a.txt.

3 "I was elected to lead, not to read"

To jump straight in, we can try to run on some events from the MC ttbar sample. The code needs for that a list of input files and a steering file. You can get the steering file from AnalysisTop examples, environmental variable "$AnalysisBase_DIR" (or "$AnalysisTop_DIR" if you're still using AnalysisTop for old realeses) is set when you set up AnalysisTop. You can use your own test file to run on that you created in the previous step, "test_file_mc16a.txt", or simply use a provided test file from AnalysisTop.

Please type in the following now:

```
cd ~/MyAnalysis/run/
top-xaod $AnalysisBase_DIR/data/TopAnalysis/validation-cuts.txt $AnalysisBase_DIR/data/TopAnalysis/input-mc-rel21.txt
```

While the event selection is running, have a look into validation-cuts.txt. There are ten different cut flows: three dilepton and two lepton+jets channels, each of them defined once for 2015 and once for 2016 data.

When completed, there will be a flat ntuple file called output.root which can be opened with root.

```
root -l output.root
new TBrowser()
```

One folder per channel is available, with stored the cutflow and more histograms for the events that survived the selection. For example ejets folder contains events passing electron+jets requirements, with all objects and event information like weights, missing et, etc.

The trees contain the information of all selections, stored together in branches, and allows for further analysis of the event. For each selection enabled there is a boolean variable that tells if the event passed the specific selection or not. One tree for each active systematic uncertainty is created. The tree called nominal is the basic one without variations.

3.1 Command line options

As you have seen in the previous example, the executable called top-xaod (source code) takes exactly two command line arguments, both of which are file names of text files. The first is the configuration information, the second is a list of input files.

Feel free to take a look inside $AnalysisBase_DIR/data/TopAnalysis/ dir to find out other examples of configuration cut files:

```
less $AnalysisBase_DIR/data/TopAnalysis/dil-cuts.txt
```

or input files:

```
less $AnalysisBase_DIR/data/TopAnalysis/input-data.txt
```

We deliberately limit the command line options. All settings are handled in the configuration file. We do not want a hybrid system where settings can also be applied via the command line.

To locally test an event selection, a local copy of the steering file is needed. E.g.

```
cp $AnalysisBase_DIR/data/TopAnalysis/validation-cuts.txt .
```

See below for how to handle some of the available options.

To compare the old and new selection, a script is provided:

```
mkdir comparison_plots
$AnalysisBase_DIR/user_scripts/TopAnalysis/validation.py comparison_plots/ output.root output-modifiedSelection.root
```

It is then possible to browse through the plots with:

```
firefox comparison_plots/index.html
```

3.2 Configuration options

AnalysisTop can be configured in many different ways, with everything controlled through text files. As the number of different options was growing and growing, rather than force everyone to change their configuration files every time we add a new feature, many options have default values and do not need to be specified. There is a complete list of the options currently available in the gitlab link. A description of commonly used ones follows.

Number of events to run over

This is very useful for debugging and developing your code. E.g. to just test some new feature on first few events:

```
NEvents 20
```

First events to run over

This is very useful for debugging and developing your code. E.g. to just test a bug occurring at a specific event, especially if late in a large file:

```
FirstEvent 100
```

Modify output file name

Change the default to your favorite name:

```
OutputFilename output.root
```

Good Runs Lists

GRL files are stored in CVMFS.

This twiki shows the recommended way how apply the GRL selection

Hopefully latest GRL xml files should be available there, eventually you can download it from here SpecialRunsIn2015. With your new GRL file, you can:

```
GRLDir MyAwesomeAnalysis
GRLFile SomeGRLFile.xml
```

PRW and Lumicalc files

To perform pileup reweighting (at least) two files are needed. One contains the information about the average mu distribution in your MC (see Generating PRW config files) and one containing the information about the average mu distribution in data, which can be generated from a GRL (Luminosity Calculator). PRW files are stored in CVMFS.

This twiki stores the most up-to-date recommended configurations

Mode information about the PRW files is provided in this link.

FullSim vs. AFII

If you use

```
UseAodMetaData True
```

the information about the simulation will be taken from the metadata automatically. If this fails, the code will then rely on "IsAFII" key:

```
IsAFII True
```

Physics Objects

The object collection name needs to be specified, otherwise AnalysisTop won't run with that object. So if you want to turn off muons for whatever reason, just specify

```
MuonCollectionName None
```

From the ConfigurationSettings should be easy to figure out the available kinematic cuts, working points and default values. For example these are the "defaults" (but each analysis may use different WPs)

```
ElectronID TightLH
ElectronIDLoose LooseAndBLayerLH
ElectronIsolation Gradient
ElectronIsolationLoose None
ElectronPt 25000.
ElectronVetoLArCrack True
...

MuonQuality Medium
MuonQualityLoose Medium
MuonIsolation FCTight_FixedRad
MuonIsolationLoose None
MuonPt 25000.
MuonEta 2.5
...
```

Please be aware that the definitions used here will always affect the MET and the Overlap Removal, e.g. even if you're not using explicitely muons in your analysis, the definition you have for muons in your config file will be used for calculating the MET in the event (see here) and for the Overlap Removal procedure.

More information on the implementation of the physics objects in AnalysisTop are provided in this twiki.

MET

MET (i.e. the transverse missing energy) is rebuilt using the METMaker in all events. This uses information from objects present in the event: for this reason, even if an analysis is not using some specific objects, it's important to use a sensible definition of such objects in the AT config file. For example, an analysis which is not explicitely using muons, will still find different MET values depending on the muon definitions (e.g. MuonQuality, MuonPt...) used in the config file; also not using at all a given object collection (e.g. setting MuonCollectionName None) will impact the MET wrt using muons with some definition.

Muons

Since release 21.2.118 *doExtraSmearing* and *do2StationsHighPt* options have been added with this MR. Names have been changed to *MuonDoExtraSmearingHighPt* and *MuonDoSmearing2stationHighPt* for releases >=21.2.121.

More information will be provided on this twiki.

Soft Muons (use release>=21.2.101)

MR27156, available from AnalysisTop release 21.2.96 on (but with a significant bug that will be fixed from release .101), introduced the possibility of running on soft muons. In practice this means that a second collection of muons, with a different selection wrt to that used for "prompt muons", is saved in the output ntuples. The following options can be used in the configuration file:

```
UseSoftMuons True #default is False
SoftMuonQuality Tight #LowPt, Loose, Medium, Tight
SoftMuonPt 4000.
SoftMuonEta 2.4
SoftMuonDRJet 0.4 #can be set to 999 if no selection has to be applied here
```

The options are quite self-explanatory. The SoftMuonDRJet represent the upper DeltaR cut which is applied between soft muons and selected jets; this is useful because the typical use case for soft muons is to select muons inside jets. The cut can be however set to an arbitrary large number in case no matching between soft muons and jets is required. **Currently it's not possible to apply isolation or TTVA cuts on soft muons.**

It is possible to apply a request on the number of soft muons in the event in the selections in the config file with a line like:

```
SOFTMU_N 6500 >= 1
```

The soft muon 4-vector, d0, d0sig, z0*sintheta, truth origin, truth type, isPrompt variables are saved. Also reconstruction SFs are saved FOR EACH soft muon, to allow then the user to do what he/she wants with that. **Note that soft muons SFs have to be manually applied by users and ARE NOT included in the event-level leptonic SFs**. In the nominal output tree, systematic variation of the soft muons SFs are also stored. Isolation and TTVA scale factors are not stored for soft muons, since isolation and TTVA cuts are currently not applied on them. In systematic trees corresponding to muon momentum calibration uncertainties, soft muons momenta are varied (as well as prompt muons ones as usual).

Additional variables for soft muons can be added in CustomEventSaver. Selected soft muons are available from the TopEvent: event.m_softmuons collection.

Truth origin of the soft muons (available from AB 21.2.106 on)

MR29088 added the possibility of writing in output information on the origin of a soft muon.

The following options are added to the config files for AT:

- SoftMuonAdditionalTruthInfo True/False (default: False) -> decide if you want to store additional truth information on the particle-level origin for soft muons (see TopParticleLevel/TruthTools.h)
- SoftMuonAdditionalTruthInfoCheckPartonOrigin True/False (default: False) -> Decide if you want to store additional truth information on the parton-level origin for soft muons (see TopParticleLevel/TruthTools.h, this makes sense only if also SoftMuonAdditionalTruthInfo is True)
- SoftMuonAdditionalTruthInfoDoVerbose True/False (default: False) -> produce debug printouts for these options

since these options are by default disabled, the standard user won't experience any change to the AT behaviour. If these options are used, the following output is added to the output reco-level ntuples:

- softmu_parton_origin_flag -> flag with information on the partonic origin of the soft muon, see the corresponding enum

in TopParticleLevel/TruthTools.h
- softmu_particle_origin_flag -> flag with information on the particle origin of the soft muon, see the corresponding enum

in TopParticleLevel/TruthTools.h
- softmu_parent_pdgid -> pdg ID of the direct mother of the soft muon
- softmu_b_hadron_parent_pdgid -> pdg ID of the B hadron originating the muon for B->mu, B->C->mu, B->tau->mu, B->C->tau->mu decays
- softmu_c_hadron_parent_pdgid -> pdg ID of the C hadron originating the muon for C->mu, C->tau->mu, B->C->mu, B->C->tau->mu decays

Decorations such as SG::AuxElement::Accessor Mother("truthMotherLink") and analougous truthBMotherLink, truthCMotherLink (and more) are added to the soft muon, in case one user wants to retrieve links to these particles in a CustomSaver. A ("truthMuonLink") decoration is also added to the reconstructed-level soft muons, to easily retrieve the associated truth muon.

This code has been tested with Pythia8, Herwig7 and Sherpa samples. Due to loop in ancestors history in Sherpa, the parton information option cannot be used with Sherpa; there are protections based on the number of iterations when running through particle ancestors to protect from such loops, inspired to the analogous protections used in the MCTruthClassifier.

Small-R Jets

Small radius jets collection to be used is specified using the option

```
JetCollectionName AntiKt4EMTopoJets
```

where `AntiKt4EMTopoJets` and `AntiKt4EMPFlowJets` are the two currently supported options. Note that it is not possible to run AT without small-R jets, i.e specifying `None` is not supported.

IMPORTANT b-tagging related changes for small-R jets configuration

For derivations with tag p3954 or higher (corresponding to derivation cache release >=21.2.72.0), the Flavour Tagging group has changed the structure of storing b-tagging information in the derivations to enable possibility of re-training flavour tagging algorithms. This affects all analysis frameworks including AnalysisTop. If you are using the above derivation release or newer, you must specify in the top-xaod config an extended jet collection name:

```
JetCollectionName AntiKt4EMTopoJets_BTagging201810
```

Where the supported options are `AntiKt4EMTopoJets_BTagging201810`, `AntiKt4EMPFlowJets_BTagging201810` (EMPFlow jets with b-tagging algorithms trained on EMTopo) and `AntiKt4EMPFlowJets_BTagging201903` (new, dedicated re-training of b-tagging algorithms on PFlow jets).

Internally, AT will check the `JetCollectionName` and the derivation release from AODMetadata and throw an error if you try to use the new jet collection name configuration with older derivation release or vice-versa. **Note that it is highly recommended to set option `UseAodMetaData True` so that AT reads this kind of meta-data from the derivation files.**

If you are interested in the technical details on the updated derivation structure of storage of jets and b-tagging information, see these slides.

Since AnalysisTop 21.2.95 - you can choose which CDI file to use for your analysis. This is important as new derivations (p-tag >= p39XX) work only with the new CDI files. For example you can use:

```
BTagCDIPath xAODBTaggingEfficiency/13TeV/2019-21-13TeV-MC16-CDI-2019-10-07_v1.root #this is just an example here
```

However, the older ptags (e.g. p3832) will work only with the older CDI file (set as default), and the old style JetCollectionName.

AnalysisTop by default will use the most updated one available at the time the release was built. When you use different CDI file that the default, a flashy warning will be printed.

Whatever you do, i.e. setting the CDI file manually or using the defaul one, always make sure to check carefully you're doing what you really intend to do. We keep updated information on the TopRecoObjTwikiModel page.

Large-R jets

If the user wants to use LC-Topo jet, he/she should do:

```
LargeJetCollectionName AntiKt10LCTopoTrimmedPtFrac5SmallR20Jets
```

while if the user wants to use TCC jet, he/she should do:

```
LargeJetCollectionName AntiKt10TrackCaloClusterTrimmedPtFrac5SmallR20Jets
```

Some useful information about TCC jet can found in these slides, and this document

BadBatman

Since AnalysisTop 21.2.73 (May 2019), an option to remove events in data flagged with BadBatman (see JetMET TWiki) is available. If you want to apply the BadBatman cleaning use

```
UseBadBatmanCleaning True
```

By default, no BadBatman cleaning is applied. The cleaning is applied during the JETCLEAN selection step. If you turn on the cleaning, it is by default applied to 2015 and 2016 data events. You can also specify a custom range of run numbers where the cleaning is applied by using

```
BadBatmanCleaningRange XXXX:YYYY
```

The run numbers cannot include run numbers for 2017 as the cleaning should not be applied to 2017 data taking (cannot include 325713-348835).

If you apply the cleaning read the JetMET twiki! You will need to adjust luminosity value and its uncertainty.

JVT WP

It is now possible since AnalysisTop 21.2.74 (May 2019) to specify the JVT WP. It is by default set to "Default" (sets Tight for PFlow jets and Medium for EMTopo jets) but this can be changed using

```
JVTWP Tight
```

The available options can be found here.

fJVT WP

It is possible since AnalysisTop 21.2.127 (June 2020) to specify the fJVT WP for forward jets, |eta| > 2.5. It is by default set to "None" which does not run any fJVT tools but can be set to Tight (recommended) or Medium to match the Tight/Loose working points defined here. These options include use of the updated Hard-Scatter SFs released in May 2020 to build nominal and systematic fJVT weights, available in the output nominal tree.

```
ForwardJVTWP Tight
```

IMPORTANT: The fJVT calculation for PFlow jets needs to be done at derivation level, this functionality was added in derivation release 21.2.97.0 so users MUST use ForwardJVTWP None if they want to use PFlow jets with an earlier derivation release

An additional boolean option is also present:

```
ForwardJVTinMETCalculation False #(True, False [default])
```

This allows the fJVT decision to be used within the METmaker to improve the MET resolution. As this option can be used for analyses with a central-only jet selection this can be set independently of the fJVTWP option, in which case the Tight WP will be used, and SFs will **NOT** be applied.

More information on using the different fJVT options can be found here

Forward electrons (available since AnalysisTop 21.2.80)

IMPORTANT: Recently, a bug has been found in the configuration of the forward electron identification. All recommendations related to forward electrons are therefore not valid, and forward electrons should not be used in analysis until further notice. See announcement on the hn-atlas-top-reconstruction mailing list on 25/7/19 09:47 CERN time. Further news on this topic will be announced on the same mailing list.

From AnalysisTop 21.2.80 (July 2019) Forward electrons have been included in AnalysisTop (MR23841). By default they're not included when running AT, they can be used with the following options in the config file:

```
FwdElectronCollectionName ForwardElectrons #(default is None)
FwdElectronID Tight #(Loose, Medium, Tight [default])
FwdElectronIDLoose Tight #(Loose, Medium, Tight [default])
FwdElectronPt 25000. #(25000. is default)
```

</sticky>

It must be noted that their SFs ARE NOT inlcuded in the standard leptonSFs, but there are separate SF weights for them which should be taken into account when using Forward electrons.

FwdElectrons can be requested in selections with e.g.: FWDEL_N 22000 >= 1. It must be noted though that this statement will only have effect at reconstructed level: at truth level there is no distinction between forward and "standard" electrons, so this statement in the selection is designed to always be true at truth level. *TruthElectronPt* and *TruthElectronEta* options in the config file can be used to specify what is the pt/eta range considered for electrons at truth level.

B-tagging showering algorithm and TopDataPreparation

B-tagging SFs are derived by default on Powheg+Pythia8 samples. Since for other showering algorithms they can differ, the b-tagging group provides MC/MC SFs, see b-tagging twiki. When running on a MC sample, AnalysisTop thus needs to know what is the showering algorithm used by the sample, so that the appropriate MC/MC b-tagging SFs can be applied. This information is retrieved from a TopDataPreparation (TDP) file read from the cvmfs area /cvmfs/atlas.cern.ch/repo/sw/database/GroupData/dev/AnalysisTop/TopDataPreparation . The file is copied every few hours in that area from the one included in this git repository. That TDP file contains lines like:

```
410470 396.87 1.1398 pythia8
```

with the first column being the DSID, the second the MC xsec times the filter efficiency, the third is the k-factor, and the fourth one is the showering algorithm used in the sample.

For AnalysisTop releases <=21.2.83, the TDP file used by default from AnalysisTop is XSection-MC15-13TeV.data; the allowed showering algorithms are pythia8, herwig/herwigpp (same MC/MC SFs are applied for both), sherpa (applied MC/MC SFs for sherpa 2.2), sherpa21 (sherpa 2.1, MC/MC SFs ARE NOT AVAILABLE, THESE ARE SET TO 1). For aMCatNLO+Pythia8 samples, MC/MC SFs are set to 1 (so the same SFs as for Powheg+Pythia8 samples are used).

For AnalysisTop releases >=21.2.84, the TDP file used by default from AnalysisTop is XSection-MC16-13TeV.data; in this case amcatnlopythia8 is added as a possible showering, and therefore appopriate MC/MC SFs are used for aMCatNLO+Pythia8 samples.

In order to request the addition of a sample to the TDP file, please open a JIRA ticket on the AnalysisTop board.

Which TDP file is used by AnalysisTop can be changed in the configuration file with the option TDPPath. By default this option is *TDPPath dev/AnalysisTop/TopDataPreparation/XSection-MC15-13TeV.data* for releases <=21.2.83, and *TDPPath dev/AnalysisTop/TopDataPreparation/XSection-MC16-13TeV.data* for releases >=21.2.84. Users using the MC16-13 TeV campaign will most likely not need to set this option, and just use the AnalysisTop default.

More info on the topic are in the AT tutorial

Systematics

A lot of thought was given to systematics in the AnalysisTop design. We only calibrate objects once, and re-use these objects multiple times. This cuts down the number of calculations required and results in faster execution times and a lot of flexibility.

For example, using the Full NP JES model (see below) you will end up with around 150 systematics TTrees. Each of these systematics will be using the nominal electrons and muons. We share these nominal electrons and muons with all the jet systematics. In fact, we only calibrate (CPU-expensive) the jets once, we copy (CPU-cheap) the calibrated jets and smear these 150 times. Other methods of doing systematics, like a simple for loop over a list of known systematics, will typically perform 150 calibrations of the nominal electrons and muons, which we think is a waste of time.

In your configuration file you should see

```
Systematics Nominal
```

Change this to

```
Systematics All
```

and you will get everything. For people who want something in between, then you can specify:

```
Systematics AllElectrons
```

And similarly for AllPhotons, AllMuons, AllTaus, AllSmallRJets, AllLargeRJets, AllTrackJets, AllGhostTrack and AllMET

You can even specify exact systematics, like:

```
Systematics MUONS_SCALE__1up
```

and you will get a file with the nominal and MUONS_SCALE__1up TTrees. You can form comma separated lists, so:

```
Systematics MUONS_SCALE__1up,EG_RESOLUTION_ALL__1down
```

will give you all 4 of these systematics as well as the nominal. Please note that we do not control the names assigned to the systematics, this comes directly from the CP groups.

Since 21.2.105, it is possible to use * as wildcard to select systematics. Example:

```
Systematics MUONS_*__1up
```

will give you all shifts up for muons.

The additional parameter DoSysts allows you to select which lepton objects definition to use: Tight, Loose or Both (default). In combination with the DoTight and DoLoose parameters, which allow to run only the corresponding nominal trees, you could for instance keep a loose selection for fake estimation while running and saving systematic variations only from the tight nominal TTree by adding the following lines to your config file:

```
DoTight Both
DoLoose Both
DoSysts Tight
```

Other systematic options

Jets

The Jet Energy Scale (JES) model group currently recommends either using all nuisance parameters (leading to roughly 150 TTrees), 30 nuisance parameters, 20 nuisance parameters or 4 nuisance parameters different strongly-reduced sets. The default settings in AnalysisTop is to run all nuisance parameter which is equivalent to have

```
JetUncertainties_NPModel AllNuisanceParameters
```

To run the 30 NP JES parametrization, do:

```
JetUncertainties_NPModel CategoryReduction
```

Note that you need to use this option if you want to combine results with CMS.

Other options are (20 NP):

```
JetUncertainties_NPModel GlobalReduction
```

and (4 NP)

```
JetUncertainties_NPModel StrongReduction
```

When using StrongReduction, this will produce trees for all 4 scenarios. These must be compared to show there is no sensitivity to correlations in your given selection. Once this has been shown, you can use the first scenario using

```
JetUncertainties_NPModel SR_Scenario1
```

StrongReduction should only be used for analyses which are insensitive to JES. Such analyses cannot be used in combinations. Please see here for more detailed information.

You can set bunch spacing with:

```
JetUncertainties_BunchSpacing 25ns
```

Jet Energy Resolution. The default is to run in the simple mode, which only smears the MC with a 8 NPs. There are different option available: "Full" smears both data and MC and results in 13 NPs. "Full_PseudoData" smears pseudodata and MC. "All" smears both data and MC and results in 34 NPs. "All_PseudoData" smears pseudodata and MC. You can specify the JER mode with:

```
JetJERSmearingModel Full
```

Warning: Not all combinations of JES and JER modes are available. List of available combinations can be found in this Twiki.

NB: In AnalysisTop versions < 21.2.44 the default option for JER provides only 1 NP.

Jet calibrations for analyses using boosted vector bosons or jet reclustering

As of August 2019, there are new Jet Mass Scale calibrations for R=0.4 PFlow and EMTopo jets. These are useful to analyses wanting to use the jet mass to identify extremely boosted vector bosons (or other particles) where all the decay products are contained within a single R=0.4 jet. This calibration will also affect analyses using jet reclustering.

These calibrations are not used in AT by default however you can add them in yourself. To do so, first checkout the TopCPTools package in AT. Then go into Root/TopJetMETCPTools.cxx. The lines which need changing are 38-54. These strings contain the config file and calibration sequence for each jet type, data or MC and FS/AFII. The config name and calibration sequence should be changed to those shown in the Jet Calibration Twiki. The corresponding file names are shown below the section titled "Calibration for data or FullSim + nominal JER MC-smearing + mass calibration available in 21.2.71 or later".

Once this is updated, rebuild AT and the new calibrations will be used.

Large-R jet

LC-Topo large-R jet

The JES uncertainties for LC-Topo large-R jets have 3 options in the recommendation, all (~180) nuisance parameters, category reduction (~30 nuisance parameters) or global reduction (~20 nuisance parameters). The default settings in AnalysisTop is to run category reduction which is equivalent to have

```
LargeRJetUncertainties_NPModel CategoryReduction
```

To run the all nuisance parameters, do:

```
LargeRJetUncertainties_NPModel AllNuisanceParameters
```

Note that you need to use this option if you want to combine results with CMS.

or to run in the global reduction:

```
LargeJetUncertainties_NPModel GlobalReduction
```

Noe that if the user specify the same option in JetUncertainties as in LargeRJetUncertainties, only one tree will be generated for each nuisance parameter which correlate the large-R jet JES to the small-R jet JES.

Also, the uncertainties for JER, JMS and JMR are propagation from the previous recommendation, for now.

The JES and JMS calibration for LC-Topo large-R jets has 3 options in the recommendation, Combined mass, calorimeter mass and track-assisted mass. But in the current version of JetUncertaintiesTool, the second and the third option is temporarily disabled. This is saying that all 3 of them can be used in the nominal tree but only the calorimeter mass can be used for uncertainties now.

The default option in AnalysisTop is calorimeter mass which is equivalent to

```
LargeRJESJMSConfig CombMass
```

To switch to calorimeter mass, do: (cannot be used when doing systematic trees for now)

```
LargeRJESJMSConfig CaloMass
```

To switch to track-assisted mass, do: (cannot be used when doing systematic trees for now)

```
LargeRJESJMSConfig TAMass
```

TCC large-R jet

The JES/JMS uncertainties for TCC large-R jets have only 1 option in the recommendation, which must be set manually by the user in the following way:

```
LargeRJetUncertainties_NPModel Scale_TCC_all
```

The recommendation mentions some kinematic limits for the uncertainties which might be broken in extreme cases: "Mass cuts depending on stats High pT and high m/pT jets lack sufficient statistics for the method to derive an uncertainty. The tool will tell you if your jet is outside of validity or not as mentioned above. Roughly speaking, this will only happen for jets with m/pT > 0.5 and reasonably high pT (varying with m/pT)"

The JES/JMS calibration for TCC large-R jets have only 1 option in the recommendation, which must be set manually by the user in the following way:

```
LargeRJESJMSConfig TCCMass
```

Jet Ghost Tracks

AnalysisTop supports ghost tracks associated to jets. However, their usage is still EXPERIMENTAL since the implementation was not yet tested in physics analyses.

Ghost tracks are activated by adding line

```
JetGhostTrackDecoName GhostTrack
```

to the cuts-file. One needs to know what is the thinning procedure applied on the specific jet collection in the derivation used in analysis.

One can control jet pt and eta thresholds to load ghost tracks in order to avoid loading vectors of tracks affected by jet thinning. This jet pt and eta criteria are always applied on nominal jets. Therefore, the Pt criterion should be set below the 'JetPt' cut which is applied on shifted jet after systematic variations are applied. Default values for Pt and Eta are

```
JetPtGhostTracks 30000 [MeV]
JetEtaGhostTracks 2.5
```

If nominal jet pt is above 'JetPtGhostTracks' and |jet eta| <= 'JetEtaGhostTracks' cut, all tracks are kept in the ghost track vector. However, null pointers are removed from ghost track vectors. Warning message is printed if vector of ghost tracks contains null pointers only.

Systematic variations related with usage of ghost tracks can be added to the output using

```
Systematics AllGhostTrack
```

They are also activated using 'Systematics All' option which activates all systematics variations.

Quark/Gluon fraction in Jets Uncertainties

It is possible to reduce the JES if the true composition of quark and gluons is known. In order to use this feature, a 3-step procedure was implemented in AnalysisTop. More information can be found in this presentation, and in the TopJetFlavourComposition13TeV twiki.

1) Run with AnalysisTop with the JETFLAVORPLOTS flag in your selection, for example:

```
JETFLAVORPLOTS pt:15:20:30:45:50 abseta:0.:0.3:0.8:1.2:2.1
```

This will create a root file that is used as input for the next step

2) Run the stand-alone MakeQuarkGluonFractionPlots code that you can find here.

```
root[0].x MakeQuarkGluonFractionPlots.cxx++ ("MakeQuarkGluonFractionPlots.config")
```

This will create several files with control plots and an input file to be used in the next step

3) Rerun AT specifying the new file in AT config file (do not forget to add this file to prun with "--extFile " if you run on the grid):

```
JetUncertainties_QGFracFile path/FlavourCompostion.root
```

The procedure should be able to produce also the q/g fraction in Njet regions but AT currently does not support this.

Jet response plots

You can generate 2D histograms where on Y axis the reco pT - truth pT / truth pT for matched jets is stored as a function of truth jet pT (X axis). The same TruthJetCollection is used for the truth jets. The matching is done with a simple delta R criterion. Inclusive plot (all jets) as well as per flavour plots are stored. The following syntax should be used:

```
JETRESPONSEPLOTS deltaR:0.2 bins:10 min:-1 max:1 ptBinning:0,30,50,100,200,500,999999
```

Where `deltaR` controls the critical delta R value for matching, `bins`, `min` and `max` control the bins and range of the Y axis (the jet reposponse) and `ptBinning` controls the bin edges for the X axis.

Egamma

Egamma Systematic Model. By default AnalysisTop uses the 1NP_v1 model, the other options are FULL_ETACORRELATED_v1 and FULL_v1. You could do:

```
EgammaSystematicModel FULL_ETACORRELATED_v1
```

B-tagging eigenvector uncertainties.

The eigenvector variation method is developed to reduce the number of variations in the b-tagging calibration. More details can be found here.

Sometimes, the sources of systematics are shared between the construction of b-tagging eigenvector and the physics analysis. In that case, these sources should be removed from the eigenvector construction and varied simultaneously in the b-tagging calibration and in the physics analysis. The following configuration is used to specify those sources that are to be excluded from b-tagging eigenvector construction:

```
BTaggingSystExcludedFromEV
```

where the possible names of the systematics can be listed by:

```
top-tool-ftag cutfile inputfile
```

In this way, in the systematic tree corresponding to the correlated systematics, the nominal b-tagging weight will be replaced by the varied one, thus the systematics is correlated.

Sometimes, the naming of sources in b-tagging calibration (from the CDI file) could be different from the naming of the systematics tree (from ASG). In that case, the nominal b-tagging weight will not be replaced automatically in the systematic tree, but has to be done by hand. To do that, first you need to store the varied b-tagging weight in the systematic tree by:

```
DumpBtagSystsInSystTrees True
```

which will save, in addition to the nominal b-tagging weight, all the varied b-tagging weights corresponding to the requested b-tagging uncertainties listed in "BTaggingSystExcludedFromEV". Then you can do a manual correlation by using the varied b-tagging weight rather than the nominal b-tagging weight with the systematic tree.

Store per-jet btag SFs

By default AnalysisTop stores per-event b-tag weights to reduce the output file size. If you want to store per-jet b-tag weights, use

```
StorePerJetBtagSFs True
```

Example of systematic variation on scale factors

Tau

AnalysisTop uses the default setups for the Tau systematics provided by the TauCP group.

Current tau systematic variations in AnalysisTop include scale factor corrections to:

- Tau reconstruction in high pt (TAU_SF_RECO_HIGHPT_UP/DOWN) and all pt range (TAU_SF_RECO_TOTAL_UP/DOWN).
- Tau identification in high pt (TAU_SF_JETID_HIGHPT_UP/DOWN) and all pt range (TAU_SF_JETID_TOTAL_UP/DOWN).
- Electron veto scale factors aimed at tau candidates (TAU_SF_ELEOLR_TOTAL_UP/DOWN) and electrons (TAU_SF_TRUEELECTRON_ELEOLR_TOTAL_UP/DOWN).

The systematics are retrieved by the function ScaleFactorRetriever::tauSF where you provide the tau candidate and the systematics such as top::topSFSyst::TAU_SF_RECO_HIGHPT_UP.

and tau energy scale variations at:

- Detector level (TAUS_TRUEHADTAU_SME_TES_DETECTOR__1up/1down)
- In-situ level (TAUS_TRUEHADTAU_SME_TES_INSITU__1up/1down)
- Model level (TAUS_TRUEHADTAU_SME_TES_MODEL__1up/down)

Electrons

The Electron Charge ID Selection tool (ElectronChargeFlipTaggerTool) aims at reducing the numbers of electrons that are reconstructed with a wrong charge. The variable el_ECIDS is set to 1 if the electron it passes the tool selection (N.B. the only available working points are MediumLH /TightLH ID and FCTight/Gradient isolation), while el_ECIDSResult stores the corresponding BDT output value. The following scale factors should be used associated to the tool: those computed by the EgammaChargeMisIdentificationTool (adjust the fractions of right/wrong charge electrons), stored in the weight_indiv_SF_EL_ChargeMisID variable; and those by the XAODElectronEfficiencyCorrectionTool (difference in selection efficiency of the Charge ID selector tool between data and MC), stored in the weight_indiv_SF_EL_ChargeID variable. The first scale factor is associated to four additional variables with _STAT_UP/DOWN and _SYST_UP/DOWN variations. To the second scale factor two additional variables are stored for the systematic _UP/DOWN variations.

Photons

The scale factors are provided by the tool PhotonEfficiencyCorrection, to be applied on MC to match the efficiency measured in data, and saved in the variables (systematic variation name):
weight_photonSF (nominal)
weight_photonSF_ID_UP (PH_EFF_ID_Uncertainty__1up)
weight_photonSF_ID_DOWN (PH_EFF_ID_Uncertainty__1down)

The uncertainties corresponding for photon track isolation efficiency are also calculated. Tools with name "AsgPhotonEfficiencyCorrectionTool_IsoSF" + "isolation working point" (Loose, Tight, TightCaloOnly) are initialized and can be retrieved, but only Tight isolation working point scale factors are currently saved in the output. The following variables (systematic variation name) are available :
weight_photonSF_effIso (nominal)
weight_photonSF_effLowPtIso_UP (PH_EFF_LOWPTISO_Uncertainty__1up)
weight_photonSF_effLowPtIso_DOWN (PH_EFF_LOWPTISO_Uncertainty__1down)
weight_photonSF_effTrkIso_UP (PH_EFF_TRKISO_Uncertainty__1up)
weight_photonSF_effTrkIso_DOWN (PH_EFF_TRKISO_Uncertainty__1down)

Truth options

The following are True/False options that are all set to False as default. So if you want any, you can turn them on

Name	Parameter values	Description
TruthBlockInfo	True/False (default: False)	full Truth block info - this is very large
TopParticleLevel	True/False (default: False)	For performing particle level analysis, stored in the particleLevel tree in the output file
for release >=21.2.115 TopPartonLevel	True/False (default: True)	For performing parton level analysis, stored in the truth tree in the output file

Three options are available for defining which PDF output is written

Name	Description
PDFInfo	False: nothing is written (Default), True: PDF written in truth and particle trees, Nominal: Stored in truth, particle and nominal trees

If you do set

```
TopParticleLevel True
```

You can also set the truth electron, muon and jet pT and eta cuts; refer to TopParticleLevel for a more complete description of the functionality provided by TopParticleLevel and how to use the data produced by this tool.

The TopPartonHistory option can also be used to dump parton-level information on the event kinematics.

The topologies currently supported are:

- ttbar
  - where only the t->bW decay is taken into account
  - where the t->qW (q=u, d, c, s, b) decay is considered (NB: you need to run over special DSIDs for this option to be meaningful)
- W'->tb (or SM single-top s-channel)
- Wt(b)

They are enabled by doing, respectively:

```
TopPartonHistory ttbar
```

or

```
TopPartonHistory ttbarlight
```

or

```
TopPartonHistory tb
```

or

```
TopPartonHistory Wtb
```

or

```
TopPartonHistory ttz
```

or

```
TopPartonHistory ttgamma
```

or

```
TopPartonHistory tHqtautau
```

The default parameter is False which deactivates this feature. Note that depending on the thinning of the truth contents at the stage of derivations, it may not work on all derivations. Up to now, things have been checked to work fine for TOPQ1 and EXOT4.

NB: Wtb option stores information about the top quark and W boson and their decay products. If there is additional b-quark present in the event, the information about the additional b-quark is also stored. Due to the nature of histograms contributing to Wtb events and the way the parton information is stored in MC samples, it is very difficult to identify the "other b-quarks". The code looks for b-quarks that do not come from top quarks, W bosons or directly from protons. The output of this algorithm for identifying the other b-quarks needs to be carefully cross-checked by the analyzers.

NB: The parton history calculation relies on the existance of the TruthParticles Collection in the derivation. This means, e.g. TRUTH3 and the ongoing DAOD_PHYS won't work with top parton history routines. Adaption of these routines is needed for R22/Run-3.

Fakes control region determination

Would you also like to be able to select loose MC events in order to determine a fakes control region?

```
FakesControlRegionDoLooseMC True
```

If you set this to True, please report your studies to the top fakes sub-group.

Additional user defined keys

You can also define your new configuration keys in the configuration file. Use the DynamicKeys option and add all of them comma-separated. For example having two additional keys:

```
DynamicKeys customkey,customkey2
customkey keyvalue
customkey2 keyvalue2
```

The custom key value can be accessed in a tool via:

```
top::ConfigurationSettings* configSettings = top::ConfigurationSettings::get();
std::string keyvalue = configSettings->value("customkey");
```

Jet Tagging

B-tagging

You need to define b-tagging working point first, or multiple working points/taggers, see example below. If you use track jets (have set `TrackJetCollectionName`) the specified b-tagging working points below are executed on track jets as well, assuming they are supported.

```
BTaggingWP MV2c10:FixedCutBEff_77
```

BoostedJetTaggers configuration

Boosted jet taggers are used to tag large-R jets. They can be defined in AnalysisTop config file (cuts file) with the following naming convention:

```
BoostedJetTagging &lt;TAGGER_TYPE&gt;:&lt;TAGGER_NAME&gt;
```

TAGGER_TYPE for the specific tagger can be found in the BoostedJetTaggers twiki in the tagger description. TAGGER_NAME is identical with DecorationName used by the tagger to decorate jet. The DecorationName can be found in the config file of the tagger and it is also printed during its initialization. Config files of all taggers can be found in atlas-groupdata/BoostedJetTaggers. This option will add tagging results into the output NTuple as a char variable

```
ljet_&lt;TAGGER_TYPE&gt;_&lt;TAGGER_NAME&gt;
```

Example:

```
BoostedJetTagging JSSWTopTaggerDNN:DNNTaggerTopQuarkInclusive50
```

</sticky> This will add DNN inclusive top tagger results at 50% working point to the output.

If more taggers are needed, simply add each one of them separated by a space or a comma. The lists of available taggers are in the source code. Note that the taggers are different for LC-topo jets and TCC jets. More information about boosted jet taggers can be found here

Note that each tagger may return more granular information, i.e. whether jet passed kinematic cuts denoting the region of validity of the tagger, or for example which variable cut was passed. This information is encoded in `TAccept` object. For advanced use, since 21.2.93, we decorrate each large-R jet with a bitmask (unsigned long) that encodes information about each cut defined in the `TAccept` object (more on `TAccept` usage here). The decorration is of type `unsigned long` and the name is `"TAccept_bitmap_<TAGGER_TYPE>_<TAGGER_NAME>"`.

An example how to retrieve the information about 2nd cut from DNN top tagger in TAccept (inside `saveEvent` of custom EventSaver) from a pointer to `xAOD::Jet jetPtr`:

```
unsigned long result = jetPtr-&gt;auxdata<unsigned long>("TAccept_bitmap_JSSWTopTaggerDNN_DNNTaggerTopQuarkInclusive80");
bool result_cut2 = (result &gt;&gt; 2) & 1; // use bit shift and logical and with 1 to obtain result of 2nd cut
```

To actually determine how many cuts are there and which one is which, check the log from running top-xaod where cuts for each initialized tagger is defined.

Which event-level cuts can I use?

We currently have a few "standard" cuts implemented. If you have a super awesome cut you can implement it yourself (following the instructions later) in your own library, or ask the top software guys to include it in the release if it's of general interest.

Here's what we have in TopEventSelectionTools:

Event vetoes and Utility

Name	Description
INITIAL	Does nothing, but appears in cutflows. Useful to know how many events you started with
GRL	Select only events on the specified good runs list
GOODCALO	Ensure that the LAr and Tile are working properly for the event (no noise bursts)
JETCLEAN	Apply the jet cleaning, e.g. JETCLEAN LooseBad
TRACKJETCLEAN	Apply track jet cleaning, needed for VR track jets, it is not doing anything if other track jet type is used
NOBADMUON	Remove events with a bad muon passing final analysis cut
PRIVTX	Ensure that at least a primary vertex exists in the event - Only applied if in **ALL** cutflows (inc. particle level)
EXAMPLEPLOTS	Plots some stuff in histograms
RECO_LEVEL	Allows to define a reco-level-only selection: particle-level events do not pass!
PARTICLE_LEVEL	Allows to define a particle-level-only selection: reco-level events do not pass!
SAVE	Tells the main code to save this event in the output file tree
PRINT	Prints a lot of information about this event to the terminal - useful for debugging

Physics Object Selection

Name	Description
EL_N	Select events with a certain number of electrons above a certain pt cut, e.g. EL_N 25000 == 1
MU_N	Select events with a certain number of muons, e.g. MU_N 25000 > 3
EL_N_OR_MU_N	Select at least N electrons or at least N muons, e.g EL_N_OR_MU_N 30000 >= 2
TAU_N	Take a guess based on the above
JET_N	Select events with a certain number of jets above a certain pt cut, e.g. JET_N 30000 >= 2
LJET_N	Select events with a certain number of large-R jets above a certain pt cut, e.g. LJET_N 200000 >= 1
MET	Select events with a certain missing energy e.g. MET > 60000
MWT	Select events with a certain transverse W mass

Composed Object Selection

MET+MWT	Cut on the sum of MET and transverse W mass
MLL	Select events above a certain dilepton invariant mass, e.g. MLL > 15000
MLLWIN	Reject events in a certain invariant mass window (e.g. Z peak, MLLWIN 80000 100000)
HT	Select events with scalar sum e + mu + jet pT > some value
OS	Select events with at least one opposite sign lepton pair
SS	Select events with at least one same sign lepton pair

Trigger - global trigger (recommended for lepton triggers)

First you need to define your triggers for each year before your selection, e.g.:

```
UseGlobalLeptonTriggerSF True
GlobalTriggers 2015@e24_lhmedium_L1EM20VH_OR_e60_lhmedium_OR_e120_lhloose,mu20_iloose_L1MU15_OR_mu50 2016@e26_lhtight_nod0_ivarloose_OR_e60_lhmedium_nod0_OR_e140_lh
GlobalTriggersLoose 2015@e24_lhmedium_L1EM20VH_OR_e60_lhmedium_OR_e120_lhloose,mu20_iloose_L1MU15_OR_mu50 2016@e26_lhtight_nod0_ivarloose_OR_e60_lhmedium_nod0_OR_e1
```

GTRIGDEC	Should select events based on whether they passed one of the triggers, includes trigger thresholds
GTRIGMATCH	Only select events where an electron / muon matches the trigger

You can find more information about global triggers in this Presentation.

Trigger (only for non-lepton triggers)

WARNING: You should not be using this if you use lepton triggers due to wrong calculation of trigger-related lepton SFs!

TRIGDEC	Should select events based on whether they passed the trigger, e.g TRIGDEC e25i

Jet Tagging in event selection

JET_N_BTAG	**(Recommended for Reco-Level)** Select (reco-level) events with a certain number of fixed-efficiency b-tags using the recommended definitions, e.g. JET_N_BTAG MV2c10:FixedCutBEff _77 >= 1 to select events with at least one b-tags at 77 % efficiency
TJET_N_BTAG	(New since 21.2.89) Same as above, but applies b-tagging selection on track jets instead of small-R calorimeter jets.
JET_N_GHOST	**(Recommended for Particle-Level)** Select (particle-level) events with a certain number of ghost-tagged jets, e.g. JET_N_GHOST B >= 1 to select events with at least one ghost-hadron b-tag. Tagging for other ghost-particles (c-hadrons, ...) is possible, use the source luke!

Reclustered jets

The following options can be used to get reclustered jets in the output ntuple. Event selectors based on RC jets are available since release 21.2.39.

If you want to learn more about re-clustered jets, have a look here: https://cds.cern.ch/record/1753401/files/ATL-COM-PHYS-2014-1117.pdf

Do not forget to put the following line: JetCalibSequence JMS

Options shared by fixed-R and variable-R reclustered jets (Available since 21.2.127):

Name	Option
RCInputJetPtMin	Default: 30000.0
RCInputJetEtaMax	Default: 2.5

These two options are additional object selections to filter input jet collections used in reclustering. They are applied after the standard object selection in the AnalysisTop21. This allows to use tighter object selection for RC jet inputs.

Specific requirements for RC and vRC jet substructure for jets clustered from PFlow jets: Substructure variables are calculated from ghost tracks associated to PFlow jets. For this reason, the options should be set in a consistent way with ghost tracks settings if substructure calculation is on: RCInputJetEtaMax <= JetEtaGhostTracks <= 2.5 and RCInputJetPtMin >= JetPtGhostTracks.

For fixed radius jets:

Name	Option
UseRCJets	True/False (default: False)
UseRCJetSubstructure	True/False (default: False)
UseRCJetAdditionalSubstructure	True/False (default: False)
RCJetEta	Default: 2.0
RCJetPt	Default: 100000.0
RCJetRadius	Default: 1.0
RCJetTrim	Default: 0.05
RCJET_N	Example RCJET_N 200000 >= 1

For variable-radius jets:

Name	Option
UseVarRCJets	True/False (default: False)
UseVarRCJetSubstructure	True/False (default: False)
UseVarRCJetAdditionalSubstructure	True/False (default: False)
VarRCJetEta	Default: 2.0
VarRCJetPt	Default: 100000.0
VarRCJetMassScale	Default list: m_w,m_z,m_h,m_t
VarRCJetMaxRadius	1.0
VarRCJetRho	2.0
VarRCJetTrim	0.05
VRCJET_N	Example VRCJET_N 2m_t 200000 >= 1

Event reconstruction

We've also started work on a common interface for top quark reconstruction codes in TopEventReconstructionTools

Name	Description
RECO::KLFITTERRUN	Run the KLFitter and get the output in your file. Options: kElectron, kMuon, kTriElectron, kTriMuon. Ignored if AllHadronic LH is requested
RECO::PSEUODTOP	Run the pseudo top routine for reco and particle level
RECO::SONNENSCHEIN	A start at solving exactly the dilepton kinematics
RECO::MT2	Coming soon - mt2 dilepton reconstruction

More options for KLFitter:

Name	Option
KLFitterJetSelectionMode	Options: kLeadingFour , kLeadingFive , kLeadingSix , kLeadingSeven , kBtagPriorityFourJets , kBtagPriorityFiveJets , kBtagPrioritySixJets , kBtagPrioritySevenJets
KLFitterLH	Can decide between ttH, ttbar, ttbar_JetAngles, ttZTrilepton, ttbar_AllHadronic, ttbar_BoostedLJets. Default: ttbar
KLFitterBTaggingMethod	kNotag,kVetoNoFit,kVetoNoFitLight,kVetoNoFitBoth,kWorkingPoint,kVeto,kVetoLight,kVetoBoth or kVetoHybridNoFit, see KLFitter documentation: https://github.com/KLFitter/KLFitter/blob/master/doc/WhatIsKLF.md
KLFitterTopMassFixed	True/False
KLFitterSaveAllPermutations	Default: False
KLFitterTransferFunctionsPath	Path to the transfer functions, more info under KLFitterTF. Default: mc12a/akt4_LCtopo_PP6

Event veto cuts

Events that will be finally written out can be established with the option:

```
OutputEvents SelectedEvents
```

The cuts GRL, GOODCALO, TRIGDEC, GTRIGDEC, PRIVTX are a bit special, as they have the ability to veto events. If a user selects these cuts in all of their selections then they will run in veto mode. There is absolutely no point in calibrating all the various systematics, running object selection and overlap removal if any of these cuts fail.

NOTE that PRIVTX selection is always enforced, various CP tools cannot work without the existance of a primary vertex in event. Even if you omit PRIVTX selector in your selection, events without primary vertex will be rejected, the only difference is that cutflow histograms and cutflow summary after event loop will not show the impact of primary vertex. If you want this information visible in your cutflow, leave the PRIVTX selector in the selection cuts. For advanced users: if needed (tipically for debugging, as it usually makes no sense to run without requestnig PVs) this request can be sidable setting "DemandPrimaryVertex False" in the config file.

Calibrating objects and selecting them is relatively CPU-expensive, we'd rather not do all of this and then fail an event on, for example, a trigger decision. If all of your selections request a (G)TRIGDEC line, then the trigger cut will be applied in a veto mode. This will collect all the assorted triggers in your various selections and check each of them. If none pass then we don't do any calibration, we just skip to the next event.

The way that these veto cuts are applied in AnalysisTop means that if you change the order you will get a funny looking cut flow, so please run them in the following order:

```
SELECTION MySelection
INITIAL
GRL
GOODCALO
PRIVTX
GTRIGDEC
...
```

What if you want to do a trigger study? or would rather disable this feature? Don't hack the code, define another selection:

```
SELECTION TriggerVetoOff
INITIAL
```

As there is no (G)TRIGDEC defined, the trigger will not run in veto mode. What's more as there is no SAVE, nothing will be written to your output file.

It is also possible to save all events regardless of the cuts applied, with the option:

```
OutputEvents AllEvents
```

PDF Reweighting

The recommendation for LHC experiments called PDF4LHC is followed to compute the uncertainty on the signal and/or background acceptance due to the Parton Density Functions. In order to perform PDF reweighting two things are needed. A per event weight to reweight to the new PDF set and the sum of weights (before cuts) obtained using the new PDF set for normalisation. To obtain the later (sum of weights per new PDF set), one can add the LHAPDFSets option to your configuration file followed by the PDF sets that you would like the sum of weights for e.g.

```
LHAPDFSets CT14nlo NNPDF30_nlo_as_0118 MMHT2014nlo68cl PDF4LHC15 _nlo_30
```

It is possible to have as many PDF sets as wanted, separated by spaces. This will create a new tree in your output called PDFsumWeights containing the sum of weights for each PDF set/member. For example if you chose PDF4LHC15_nlo_30 you would get a branch in the tree with 30+1 entries, one for each of the variations (the 0th corresponds to the nominal). Additionally, if you want the per event weights you can add:

```
LHAPDFEventWeights Nominal # This can also be True, in which case you will get the weights in the truth tree (for every event)
```

This will give you a new branch in your output tree per PDF set. For each event, you will have a vector of weights with the position in the vector corresponding to the variation. PDFsumWeights relies on the meta-data, stored in the DxAODs. Sometimes, the meta-data are not filled correctly, and/or XF1 and XF2 variables are not filled correctly. For these cases, it is possible to follow the old method, as described here. In particular, you can have AnalysisTop recompute XF1 and XF2 for you by using the LHAPDFBaseSet option, and give the name of the original PDF set:

```
LHAPDFBaseSet CT10
```

Note that the above described method is not supposed to work on Powheg MC samples. If your nominal sample is generated with Powheg, you have to evaluate the relative PDF uncertainty with another MC sample, and apply this uncertainty on your nominal sample.

More information on PDF reweighting may be found in the corresponding section of the TopRun2WorkshopTutorial2016.

MC samples with multiple MC generator weights

Many MC samples support on-the-fly variations performed by storing multiple MC weights in the output. In derivations, the `mc_generator_weights` variable is stored in `EventInfo` object. It is a vector of floating point numbers, where individual elements may represent various generator variations, such as e.g. various scale variations or alternative PDF sets.

For various recent MC samples that store these variations, an issue was identified, that there isn't a guarantee of the 0th element of the weights vector being the nominal MC weight. Because of this, ***since AnalysisBase 21.2.106*** we check derivation metadata to retrieve the names of the variations and by default attempt to find weights matching one of the following names defined by configuration option:

```
NominalWeightNames " nominal ","nominal","Default","Weight","1001"," muR=0.10000E+01 muF=0.10000E+01 ",""," "
```

The first matching weight is considered as nominal. If the MC sample contains more than one weight matching the list, we will print a warning in the AT log. Note the syntax for specifying the weights, the weight names are separated by commas, and each element is encapsulated in quotes, because ***spaces in the weight name do matter***. For some generators (e.g. MadGraph) rare cases were discovered where the weights include `#` and newline characters (`\n`). To be able to define such weight names in the configuration file, use at least AnalysisBase,21.2.113. Type the two special characters as `\#`, `\n`.

If you encounter a sample with a different nominal name weights from the ones above, AT will crash and print the MC generator weight names. You have to determine the nominal weight name and add it to the list above. Please also report such samples to top-reco mailing list, such that we add this weight centrally to the default options in AnalysisTop. In addition, if your sample has broken metadata it may not be possible to perform this weight name check, please refer to the following section for a workaround.

MC samples with broken metadata

Some MC samples were produced with buggy `TruthMetaData` container, due to which it is not possible to read what weights are available. If the sample has more than one weight, AT will crash reporting this, because it is not possible for us to automatically determine the nominal weight name. Instead, a config option is provided to specify the index of the weight in the MC generator weights vector: `NominalWeightFallbackIndex X`, where `X` is the index of the weight. **NOTE** that this option only applies if the `TruthMetaData` container is broken, otherwise we always try to use the method using weight names from metadata outlined above! This option is available from AT 21.2.110 on.

Storing MC generator weights

To store all MC generator weights in NTuples (for instance the Powheg+Pythia8 sample MC generator weights are needed for ISR/FSR uncertainty estimate TopFocusGroupTwiki), the following options needs to be set in configuration file:

```
MCGeneratorWeights Nominal # True - dump to truth tree, Nominal - dump to nominal tree
```

WARNING Do not forget to use appropriate sumWeights when using non-nominal weights, if you want to normalize the variation to the same cross-section as nominal.

Hint: The names of the individual weights are now stored in the sumWeights TTree in AT output files. You can view their names, for instance interactively in ROOT:

```
sumWeights->Scan("names_mc_generator_weights", "", "colsize=40");
```

This is useful to cross-check the corresponding weight indices provided by twikis, which in general should not be 100% trusted.

PDF Reweighting (Powheg+Pythia8)

It is now possible with the availability of LHEv3 weights in our nominal Powheg+Pythia8 (and other) samples, to use generator weights to reweight events to different PDFs (baseline or error sets). To store the weights in your file use:

```
MCGeneratorWeights Nominal # True - dump to truth tree, Nominal - dump to nominal tree
```

The vector elements you need are 115-144 (but be sure to check the weight names!), these corresponds to 30 NPs for PDF4LHC set, and these should be compared to the baseline which corresponds to element 11. In the particular case of Powheg, one should use this option instead of LHAPDF reweighting, as this method does not close for Powheg (though it works for aMcAtNlo - but here LHEv3 weights are also available).

WARNING Do not forget to use appropriate sumWeights when using non-nominal weights, if you want to normalize the variation to the same cross-section as nominal.

Bootstrap functionality

It is possible to store poisson boostraping weights. They can be used to evaluate the correlation between variables and to optimise binning or apply smoothing for detector systematics. Two configuration keys are implemented to enable the evaluation of the weights and to define the number of replicas.

```
SaveBootstrapWeights True #(Default: False)
NumberOfBootstrapReplicas 10000 #(Default: 100)
```

Removing branches from an output file

Sometimes you dont want all default branches stored in your output ntuples to save some disk space. Since 21.2.70, you can remove branches from the output via the standard config file with

```
FilterBranches weight_*,jet_pt
```

Which will remove all branches starting with "weight_" and a branch "jet_pt". You can provide a comma separated list of branch names that will be removed. The code supports wildcards with `*` as in the standard bash syntax. In case you need more complicated rules for removing branches, it is possible to do this in custom event saver using branchFilters.

Removing braches from truth trees

For releases <21.2.115, the same filters defined with the "FilterBranches" option are used also for the particleLevel tree and for the truth tree (parton-level) stored in the output. For releases >=21.2.115, the following options were added:

- FilterPartonLevelBranches
- FilterParticleLevelBranches

to give the user more freedom for the configuration of his job. they use the same syntax of the "FilterBranches" option.

Removing systematic trees from an output file

It is possible to remove systematic trees, which are not relevant for a given analysis, from an output file. Trees can be removed via the standard config file using option like

```
FilterTrees *JET_JER*_1up, *JET_MassRes*_1up
```

which removes all systematic trees with the matching pattern. Wildcards are supported with `*` as in the standard bash syntax. One can define multiple strings separated by commas. **This feature is available since 21.2.104.**

Monitoring disk size usage of trees

It's always a good idea to monitor how much disk space is used before sending a big production. Apart from obivous checks using e.g. "ls -ltrh" on files produced locally, a simple this root macro can be used to obtain a breakup of the disk usage in a TTree; with simple modifications (e.g. you can copy this version `/afs/cern.ch/user/m/mvanadia/public/printSizes.C`) you can get for each entry the disk size usage normalized by the number of entries in the trees. You can use this by copying the macro in a folder, opening a root file in root and do:

```
.L printSizes.C
printTreeSummary(nominal) //if you want to get the size summary for the "nominal" tree
```

3.3 Running the KLFitter algorithm

The KLFitter can be run in many different modes within AnalysisTop, to enable it in your configuration file please do:

```
SELECTION ejets
...
RECO::KLFITTERRUN kElectron
...

SELECTION mujets
...
RECO::KLFITTERRUN kMuon
...
```

Make sure that you only use the KLFitter algorithm if you have selected at least 4 jets!

Options to be set in your configuration file (BEFORE you specify your cut cutflow selection!) are:

- KLFitterJetSelectionMode = kLeadingFour , kLeadingFive , kBtagPriorityFourJets , kBtagPriorityFiveJets
- KLFitterBTaggingMethod = Recommend use kNotag or kVetoHybridNoFit, but see the KLFitter TWiki for more details
- KLFitterTopMassFixed = True or False
- KLFitterSaveAllPermutations = True (save up to 60 permutations) or False (save only the permutation with the highest event probability)

4 Running on the Grid

As in Run-I, we've put together some handy scripts. The scripts can be found here. You can copy the scripts by simply calling:

```
getExamples
```

4.1 Submission

Open up 01SubmitToGrid.py. You need to get a copy of the configuration file for the job, and put it in the submission directory. Important settings are:

- config.settingsFile - this tells the code which cut file to use
- config.gridUsername - user name
- config.suffix - unique suffix for the job name
- config.noSubmit - if set to true will not submit the job, useful for debugging

The code import two python modules: "Data_rel21" and "MC16_TOPQ1", these modules contain the exact names of the datasets

To submit the grid jobs we then run:

```
./01SubmitToGrid.py
```

Since AnalysisTop 21.2.65 a new protection has been added to the submission file `01SubmitToGrid.py`. It checks if the datasets that are about to be submitted use the latest p-tags.

```
TopExamples.ami.check_sample_status(samples)
```

The code will print warnings if you are not using latest p-tags as there is a chance that the older files are being deleted and you will have only small fraction of events. You can run the code with `config.noSubmit=True` to check the p-tags. You can also set

```
TopExamples.ami.check_sample_status(samples,True)
```

To halt the submission when an old p-tag is encountered.

You can monitor the jobs with: http://bigpanda.cern.ch.

4.2 Troubleshooting grid issues

The likelihood of encountering some issues when running jobs on grid is practically speaking 100%, therefore we provide a list of common issues that people encounter on grid, and aid how you can mitigate them here.

4.3 Download

To download the files you can use 02DownloadFromGrid.py. You need to change the datasetPattern so that it includes the sample you created. Note that (annoyingly) the output filename has been appended to the dataset name. So for example if in 01SubmitToGrid.py your suffix is called "02-04-19", you have to use in 02DownloadFromGrid.py the datasetPattern "*02-04-19_output.root" You also need to edit directory to be somewhere sensible you can download the file to. Then run:

```
./02DownloadFromGrid.py
```

4.4 Yield checks

Use 03LocalAnalysis.py to run a simple code that provides yields check. It should be the location you downloaded the files to previously (i.e. exactly equal to directory in the previous step). Then run:

```
./03LocalAnalysis.py
```

4.5 Local analysis

You can run:

```
./03LocalAnalysis.py --run
```

Files will be written to runDirectory which is test1/ by default. This script loops over all the datasets in its lists and executes the mini-to-plots code in TopExamples.

4.6 Plots

You should be able to run 04Plot.py like this:

```
./04Plot.py
```

and open index.html in your web browser. It will have some histograms containing a few top event candidates from the very first run.

5 Extend AnalysisTop for your own analysis

We have three ways that you can modify the code without having to change any of the core packages. Which, very generally, are:

- Object level selection. You can select electrons, muons, jets etc in any way you want.
- Event level selection. You can select events using your own custom cuts
- Add extra variables to the ouput.

All of these can be done in your own package, so when we update AnalysisTop in the future (and we will!) then you just need to add your new package. Also, the packages can be stored in gitlab repositories and thus can easily be shared within different groups. How to do this is presented in detail in this section of the AT tutorial. In the SettingUpAnalysisTop twiki ways of including custom packages in different type of working areas are presented.

You may be interested in setting up everything automatically in a new area: in that case you can refer to this twiki for the instructions for that: this doesn't need the setup steps described above, it's a separate procedure.

If you want instead to start from the basic custom saver provided by top-reco (that you can later modify for your own purposes), GO INSIDE THE source/ directory and then check that out:

```
cd source/
git clone https://gitlab.cern.ch/atlasphys-top/reco/howtoextendanalysistop.git
```

You can then compile as usual. The code of this example package is viewable here: HowtoExtendAnalysisTop.

This package will, for demonstration purposes only:

- Select muons with a positive eta,
- Select jets with a minimum and maximum pT cut
- Select only even numbered events
- Save additional information to the output

Of course, your package can do only part of the job, e.g. only store your custom variables or only apply your own selection.

As explained in more detail in this section of the AT tutorial, after having prepared your code in this custom package, you need to configure your config file in order to use it. You must for sure add the new packages to the list of library loaded in the in config file, e.g.:

```
LibraryNames libTopEventSelectionTools libTopEventReconstructionTools libHowToExtendAnalysisTop
```

Then, if you want for example to use your Custom Saver instead of the standard AT one, you need to do something like:

```
#OutputFormat top::EventSaverFlatNtuple
OutputFormat top::CustomEventSaver
```

Then you can run as usual.

Warning If you want to use a different naming for the package, you can change the name of the folders, headers, cxx files as you want, but do not forget to adjust the CMakeLists.txt and Root/LinkDef.h files! If you want to do this, maybe the automatic skeleton creator is useful for you, have a look at that!

Warning2 If you want to start from this package to create you custom package for an analysis, apart from renaming it as described above, you may want to upload this package to git at some point. To do that you can "disconnect" the package from the remote git repository you cloned it from with `git remote rm origin` and then set the origin to a new remote repository you may e.g. have created on gitlab (gitlab should provide you instructions on how to set the origin when you create a new package).

5.1 Event selection

We have already discussed the many event level selections in AnalysisTop. However, if you would like to add your own, this is an example of how to do it. A more complete example is in this section of the tutorial.

For the purposes of this TWiki, let's select even numbered events only. We need a loader for our package, this has to have the same name as the package itself, as such HowtoExtendAnalysisTopLoader is defined here and implemented here. The header has another ClassDef line in it, which is paired with corresponding code in the LinkDef file.

The implementation is a list of all possible event selectors you care to write. Our example EvenEventNumberSelector is very simple, it is defined here and implemented here.

The apply function is simply:

```
bool EvenEventNumberSelector::apply(const top::Event& event) const {
    return event.m_info->eventNumber() % 2 == 0;
}
```

We can then add this in any event selection, e.g.:

```
SELECTION customSelection
INITIAL
EVEN
SAVE
```

Which would save every even numbered event.

If you do come up with interesting and innovative event selections, please get in contact as we may want to integrate it into AnalysisTop for general use.

For your custom event selector you may want to consider to provide an implementation for a particle level analysis. By default, all event selectors (by means of inheriting from EventSelectorBase have a trivial implementation for particle level event selection: the selector function always returns true. A custom implementation is is only relevant if the action imposed by your event selector is meaningful on particle level; e.g. there is no meaningful GoodRunsList at particle level and consequentially there is no implementation required. The number of jets matching a pT cut however is meaningful; accordingly, a full implementation is provided.

Refer to Extending TopParticleLevel for more details on how to add a particle level implementation of your event selector.

5.2 Add extra variables to the output

The CustomEventSaver, defined here and implemented here, inherits from top::EventSaverFlatNtuple, which we let do most of the hard work.

This extension class yet again has a ClassDef line which is paired with corresponding in the LinkDef file. This allows us to write:

```
OutputFormat top::CustomEventSaver
```

To add variables to the output, you need to define them, initialize them and decide what to do for every systematic for every event. It should be simple. A complete example is available inthis section of the AT tutorial, including an example on how to add variables to the particleLevel tree.

You may also need to use variables in your event saver that are calculated in the base event saver. This is now possible trough the use of "const Get" functions. A detailed discussion of this implementation is given in this JIRA ticket. A similar solution was implemented in the same ticket for allowing cleaning of variables in a derived event saver.

5.2.1 Remove variables from the output via a custom event saver

It is recommended to remove the branches via the config file as shown in this section above. But if you like, you can do this via CustomEventSaver, which allows for more flexibility for branch removal at the cost having to write C++ code to do this.

To do this, you need to define this function in your header file:

```
int isBranchStored(top::TreeManager const *treeManager, const std::string &variableName);
```

then in the constructor of your CustomEventSaver you need to:

```
branchFilters().push_back(std::bind(&top::CustomEventSaver::isBranchStored, this, std::placeholders::_1, std::placeholders::_2));
```

and then you need to implement the function in your .cxx file. This example will remove all branches that contain "weight_photon" in their name

```
int CustomEventSaver::isBranchStored(top::TreeManager const *treeManager, const std::string &variableName){
  if (variableName.find("weight_photon") != std::string::npos){
    return 0;
  }
  return -1;
}
```

5.3 Adding information from TopPartons

The following instructions will only work when you run TopPartons in your config file, e.g. `TopPartonHistory ttbar`. These instructions will allow you to store the truth information in your reco trees in the output ntuples, so you wont need to do any reco to truth matching for the events. Be careful about the increase in disk usage though!

In your custom event saver, you need to include `#include "TopPartons/PartonHistory.h` and then in the `saveEvent` method, you can access the information from the parton history using the lines below:

```
const xAOD::PartonHistoryContainer *topPartonCont = nullptr;
top::check(evtStore()->event()->retrieve(topPartonCont, m_config->sgKeyTopPartonHistory()), "FAILURE");
const xAOD::PartonHistory *topParton = topPartonCont->at(0);

// Check if the container is valid
if (topParton == nullptr) {
  // now we will throw, but you can use this check to see if the PartonContainer is not available (which may be perfectly fine)
 throw std::runtime_error{"Problem reading top partons container"};
}

float MC_Wdecay1_from_t_pt = 0;

if(topParton->auxdata<float>("MC_Wdecay1_from_t_pt") > 0) {
  MC_Wdecay1_from_t_pt    = topParton->auxdata<float>("MC_Wdecay1_from_t_pt");
}
```

and similarly for other variables. Now you can store the variables in your custom event saver in the usual way.

5.4 Object selection

You might want a different selection for your physics objects. For the purposes of this TWiki, let's say we want to only select muons with a positive eta and jets with a pT between a minimum and a maximum value. This serves no physics purpose, it is just an example. If you want to change the selection for a physics purpose we're pretty sure you know what you want to do.

CustomMuon - Definition and implementation - this class inherits from MuonSelectionBase, which means that it has to implement the functions:

```
///For the main analysis object selection
virtual bool passSelection(const xAOD::Muon&) const = 0;

///For the loose (e.g. fakes) object selection
virtual bool passSelectionLoose(const xAOD::Muon&) const = 0;

///Because everybody likes to know what object definitions they ran with
virtual void print(std::ostream&) const = 0;
```

You are free to implement these function as you wish. For this example, selecting muons with only a positive eta value, we add these lines to the passSelection function:

```
///-- require positive eta muons for this demonstration --///
    if (mu.eta() < 0.)
        return false;
```

CustomJet - Definition and implementation - this class inherits from JetSelectionBase. As we wish to specify a maximum pT cut (no idea why you'd want to actually do this), we overload the constructor with:

```
CustomJet(const double ptMin,const double ptMax,const double etamax, const double jvtmin);
```

and in the implementation, a snippet of the passSelection function looks like:

```
///-- Jet is not below lower pT cut --///
    if (jet.pt() < m_ptMin)
        return false;

    ///-- Jet is not above upper pT cut --///
    ///-- This is a demonstration of how to extend AnalysisTop --///
    if (jet.pt() > m_ptMax)
        return false;
```

This ability of write your object selection yourself naturally extends to photons, electrons, taus and even the overlap removal.

We now need to be able to inject these selection tools into the main top-xaod program at runtime. This is done via the CustomObjectLoader class, which is defined here and implemented here. The definition has 2 key lines:

```
top::TopObjectSelection* init(std::shared_ptr<top::TopConfig> topConfig);
ClassDef(top::CustomObjectLoader, 0)
}
```

The init function must return a top::TopObjectSelection pointer. The ClassDef line is extremely important, this tells AnalysisTop that it should build a dictionary, which also needs specifying in the LinkDef file as:

```
#include "HowtoExtendAnalysisTop/CustomObjectLoader.h"
#pragma link C++ class top::CustomObjectLoader+;
```

The implementation follows the AnalysisTop ObjectLoaderStandardCuts apart from the following:

```
///-- Muons --///
    if (topConfig->useMuons()) {
      objectSelection -> muonSelection(new top::CustomMuon(topConfig->muonPtcut(), new top::StandardIsolation()));
    }

    ///-- Jets --///
    if (topConfig->useJets()) {
      double ptMax(100000.);
      objectSelection -> jetSelection(new top::CustomJet(topConfig->jetPtcut(), ptMax ,topConfig->jetEtacut(), topConfig->jetJVTcut()));
    }
```

In order to use this, you need as usual to add `libHowtoExtendAnalysisTop` (or however you called your package) to the list of library passed to the *LibraryNames* option in the config file, then to set the *ObjectSelectionName* to use your custom object loader, e.g.:

```
ObjectSelectionName top::CustomObjectLoader
```

This ensures that we actually load our custom library, and that we inject our own definition of muons and jets into the code at runtime. We are now using standard electrons, taus, large R jets and overlap removal, however we are now specifying our own muon and jet selection according to our own C++.

5.5 Still not happy and want more control?

If you're not happy starting from the existing event saver then you can write one from scratch. All you need to do is make sure that it inherits from top::EventSaverBase. Once it does this, and has the corresponding stuff in the LinkDef.h file then you can load it at run time.

You can even add decorations to the xAOD objects, or apply cuts (though maybe it's better to use the event selection mechanisms above so that you get the cutflow automagically?). But this should really allow you to do anything...

6 Debugging problems

Debuging analysis top in a controlled manner can be very useful. To save yourself from countless cout statements to trace down a problem you can run the debugging tool gdb.

First setup an alias to an update version of gdb

```
alias debug_top-xaod='/afs/cern.ch/sw/lcg/contrib/gdb/7.11/x86_64-slc6-gcc49-opt/bin/gdb --args top-xaod'
```

Then run as normal

```
debug_top-xaod validation-cuts.txt input-mc.txt
```

You will be met with the following output (roughly)

```
GNU gdb (GDB) 7.11
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from top-xaod...done.
(gdb)
```

Type 'r' and hit enter.

AnalysisTop will run as normal. Should a segfault occur type 'bt' to get information on the exact line in the code that has caused the problem.

ctrl+c followed by q will exit the debug environment.

For more details on debugging in C++ see here.

7 Unit tests and Continuous integration

Let's say you've made the kind of changes and additions described above, and now you've got a nicely put together package with some useful functionality. However, before you use this code in any actual results, you'll want to write some unit tests (tests that test the basic units of your code, e.g. classes and functions).

This way, you'll be sure everything gives correct results and behaves as you expect it to, as well as checking all possible uses of the code. Nothing worse than presenting a startling discovery to your group only to find out it's a bug in your code!

7.1 AnalysisTop continuous integration

Each merge request to the athena framework (this includes AnalysisTop) needs to pass integration tests, where the changes are compiled and tested in a simple AnalysisTop run. This is done automatically whenever a merge request is triggered, you only need to add AtlasBot as a developer to your athena fork. More information can be found here: AnalysisTopGit

7.2 Continuous integration in your custom packages

With gitlab it is possible to write your own continuous integration. You can follow these instructions. Or simply look at an existing repository and try to modify the ".gitlab-ci.yml" file to your needs.

8 Misc

8.1 I'd like a local copy of all the top packages

Have a look at: AnalysisTopGit

8.2 I'd like to run on truth derivations

And you can! This feature has been tested and is supported for the TRUTH1 derivation. When running on truth derivations, the reconstruction level processing features of AnalysisTop are forcibly disabled because there is no input data. Particle-level processing is available, however, it must be **enabled explicitly** by adding

```
TopParticleLevel True
```

to your configuration file.

For more information regarding the particle-level capabilities of AnalysisTop check the relevant twiki page.

It is also possible to run on TRUTH3 derivations provided that TopPartons is disabled (until it is adapted to them).

```
PhotonCollectionName None
TopPartonHistory False
TruthJetCollectionName AntiKt4TruthDressedWZJets
TruthLargeRJetCollectionName None
```

See recipe here (to be updated), for the additional configuration lines.

8.3 I'd like to run using multiple configuration files

This is a useful feature e.g. to compare different b-tagging WPs, or different jet collections, or MET definitions...

See this part of the tutorial for instructions.

8.4 Producing b-tagging MC/MC SFs maps in AnalysisTop (from AT 21.2.83 on)

Since AnalysisTop 21.2.83 it is possible to produce b-tagging plots on-the-fly. Since AnalysisTop 21.2.85 it is also possible to generate the maps for MC generator weights (useful for modelling variations) If you want to produce the plots you need to add a new selection line:

`JETFTAGEFFPLOTS`

You can also specify different options:

tagger:	allows you to setup the b-tagger, e.g. `tagger:MV2c10`
WP:	allows to specify the WP, e.g. `WP:FixedCutBEff_60`
NOMINAL	will use the nominal weights (is the default)
MURUP	will use the mu_R variation (up)
MURDOWN	will use the mu_R variation (down)
MUFUP	will use the mu_F variation (up)
MUFDOWN	will use the mu_F variation (down)
VAR3CUP	will use the Var3c variation (up)
VAR3CDOWN	will use the Var3c variation (down)
FSRUP	will use the mu_R variation in FSR variation (up)
FSRDOWN	will use the mu_R variation in FSR variation (down)

You can also use mutiple variations in one line, e.g. `JETFTAGEFFPLOTS NOMINAL MURUP MURDOWN MUFUP MUFDOWN VAR3CUP VAR3CDOWN FSRUP FSRDOWN tagger:MV2c10 WP:FixedCutBEff_60`

Different options are also available:

pt:	setting pT bins
abseta:	eta bins
max_pT:	maximum pT for binning
min_pT:	minimum pT for binning, used also for tool configuration, in GeV
N_pT::	pt bins, if set to < 1 uses non-constant binning (default)
max_eta:	maximum eta
min_eta:	minimum eta, used also for the tool configuration
N_eta:	eta bins, if set to < 1 uses non-constant binning (defaukt)
dont_use_event_weight	will not use event weights
use_track_jets	to use track jets
fill_total_hist	to fill the the total histogram

9 FAQ

9.1 ERROR:TopDataPreparation: The DSID * does not have a showering algorithm defined This is needed for MC/MC SF!!! ---> EXIT.

1st, check if the DSIS exists in the TDP. If you are not using customized TDP file, the following default one should be checked.

```
/cvmfs/atlas.cern.ch/repo/sw/database/GroupData/dev/AnalysisTop/TopDataPreparation/XSection-MC16-13TeV.data
```

If the DSID is not in the TDP file, please submit a Jira ticket asking AT manager to include it.

If the default one is empty, it's a known problem: the file is broken during synchronization. Please report to the top reco mailing list.

9.2 ERROR: No MC weight matches any of the names specified by NominalWeightNames option

Usually a list of available weight names will be printed in the log file, choose one and add it to your config file, e.g.

```
NominalWeightNames " dyn=  -1 muR=0.10000E+01 muF=0.10000E+01 "
```

Note, it is important to keep the spaces.

Besides, please try to use at least 21.2.118, as it can handle wield weight names correctly.

9.3 Availability of CP SF per object

By default, per object weight (except for jet btagging weight) is NOT stored in the output ntuple. Instead, event-based weights are stored. If you need per object weight, you have to save it manually in your event saver.

9.4 top-xaod: symbol lookup error: * undefined symbol: _ZN17siscone_spherical11CSphsiscone12_banner_ostrE

A recompilation (after make clean) in a clean shell will help

9.5 b-jet aware overlap removal

This OR is not supported in AT. The ftag SFs are not estimated with this OR, thus it could introduce a bias in the SFs. But if you still insist, you can checkout the TopOverlapRemovalTool in the TopCPTools package and hard-code this OR by yourself.

9.6 Object-level cutflow with the AnalysisTop

AT doesn't support object-level cutflow yet. It will be done in the future (but with low priority)

9.7 Using three different ID+Isolation level

AT doesn't support the three-level configuration yet (but it supports two levels). It will be improved in the future (but with low priority)

9.8 WARNING The dRJet decoration has not been found for the Muon

This is expected, as the current derivation misses this dRJet decoration. The muon SF is still fine.

9.9 CP::TPileupReweighting... ERROR Unrecognised channelNumber 346343 for periodNumber 300000

1st, please check if you are using the correct PRW file, which can be found here

2nd, you can open the PRW file and check if the channel number and period number exist in the file. Usually it should be there. Otherwise, you have to manully generate it by following twiki Or try your luck with the SUSY PRW file: PRWConfigFiles_FS dev/SUSYTools/merged_prw_mc16a_latest.root

9.10 Can AT work with two jet definitions (e.g. EMTopo and EMPflow)

No, it can't. However, there is an option to run two config files simultaneously and then merge the output, see twiki. So you can define two jets in two config files respectively and produce two ntuples in one go.

9.11 ERROR Attempt to retrieve nonexistent aux data item

It's usually because the code tries to retrieve a non-existing decoration. It's hard to debug just by the error message, although sometimes the name of the missing decoration could be a hint. It is suggested to use the "gdb" option, twiki, to find more information about the error. At last, you can report the issue to the top reco mailing list.

9.12 How to find k-factors for MC samples for TDP

To be added

10 References

- AnalysisTopTutorials
- Run 2 Framework Goals
- xAOD Analysis Tools
- Developing CPTools for xAOD
- ASG Tool Guidelines
- Documentation of how DataVector works - read this to get a much deeper understanding of xAOD collections

xAOD Event and Object Containers
EventInfo functions: eventNumber(), runNumber(), etc - xAOD::EventInfo
Containers: MuonContainer, JetContainer, ElectronContainer - xAOD Objects TopPhys/xAOD Repository xAOD TEvent this presentation. TestxAODFramework Repository

Major updates:
-- SimonHead - 03 Sep 2014

Responsible: SimonHead
Last reviewed by: **Never reviewed**