TopView is a common analysis package which is widely used in the ATLAS top physics working group. The package is fully based on the official ATLAS software Athena and EventView and playing a central role in the collaborative analysis model. It is a functional package which accounts for a broad range issues in implementing physics analysis. As well as being a modular framework suitable as a common workplace for collaborators, TopView implements numerous analysis tools including a complete top-antitop reconstruction and single top reconstruction. The package is currently used to produce common ntuple from Monte Carlo production and future use cases are under rapid development.

In this paper, the design and ideas behind TopView and the performance of the analyses implemented in the package are presented with detailed documentation of the contents and instruction for using the package.
CONTENTS

Preface iii

1 Ideas Behind TopView 1

1.1 Motivation 1

1.1.1 How Common Analysis Framework Helps 1
1.1.2 Athena Framework and Event Data Model 1
1.1.3 ATLAS Analysis Model and Scope of TopView 2
1.1.4 Implementation of Common Analysis Framework 3

1.2 Design 3

1.2.1 EventViewAnalysis Framework 3
1.2.2 Tools and Modules 4
1.2.3 PhysicsView Packages and HighPtView 5
1.2.4 Physics Knowledge and Configuration 5
1.2.5 JobOptions 6
1.2.6 Distributed Analysis 6
1.2.7 Root Analysis 6

1.3 Use Cases 6

1.3.1 Types of Analysis 7
1.3.2 Production of Common DPD 7
1.3.3 Physics Validation 8
1.3.4 Future Use Cases 9

1.4 References 9

1.4.1 References on the web 9
## Appendix A: Common Ntuple Reference

### 2.1 TopView Ntuple Availability
- **2.1.1 How to get ntuples**
- **2.1.2 Currently Available Ntuples**
- **2.1.3 Datasets processed**
- **2.1.4 Naming Convention:**
- **2.1.5 Jobs types:**
- **2.1.6 Check Status and Retrieve Output**
- **2.1.7 Reading Output**
- **2.1.8 Links:**

### 2.2 Object Preselection in TopView
- **2.2.1 Introduction**
- **2.2.2 Use Preselected Objects in External AOD Analysis**
- **2.2.3 Object Insertion Order and Reconstruction Algorithm**
- **2.2.4 Loose Electron Selection**

### 2.3 TopView Ntuple Variables
- **2.3.1 Top Ntuple Trees and Variables**
- **2.3.2 FullReco0 Tree**
- **2.3.3 FullRecoAna0 Tree**
- **2.3.4 FastSim0 and FastSimAna0**
- **2.3.5 Truth0 Tree**
- **2.3.6 TruthAna0 Tree**

### 2.4 Producing TopViewNtuple
- **2.4.1 Running the Default Job**
- **2.4.2 Running TopView Job on Distributed Analysis**
- **2.4.3 Customizing the Parameters.**
- **2.4.4 Changing the Structure of Trees**
3 Appendix B: The TopView Manual

3.1 Getting Started

3.1.1 How to Get Started - 12.0.x

3.1.2 How to Get Started - 11.0.x

3.2 Inside TopView

3.2.1 Package Contents

3.2.2 Python Modules (python directory)

3.2.3 JobOptions (share directory)

3.2.4 C++ Tools (src directory)

3.2.5 Root Packages (root and pyroot directory)

3.2.6 Scripts (run directory)

3.3 Physics Validation Using TopView

3.3.1 Validation Histograms

3.3.2 Event Info Histograms

3.3.3 Particle plots

3.3.4 Top Monte Carlo Validation

3.4 TopViewAnalysis

3.4.1 Preparation Stage

3.4.2 Analysis Stage

3.4.3 Other analysis available in TopView

3.5 Customizing TopView

3.5.1 Different Ways of Customization

3.5.2 Using existing jobOption or writing your jobOption

3.5.3 Writing TopView python modules

3.5.4 Writing your own C++ tools

3.6 Root Packages

3.6.1 Root Package (found under root directory)

3.6.2 PyRoot Package (found under pyroot directory)
Preface

Contributors

A lot of people contributed to the TopView project. Among with many other people special thanks to, Stan Bentvelsen and Pamela Ferrari who has kindly supported the project as convener of the ATLAS top working group; Marina Cobal who has organized the TopView tutorial in University of Udine; Vikas Bansal who has provided numerous bug fixes and also provided a few tools for the package; Nathan Triplett who has contributed to the production of the common ntuple for CSC analysis and RTT development; Patrick Ryan who has contributed to documenting the contents of TopView ntuple; Joerg Walbersloh who wrote secondary vertex tool; Joe Foster who is active in physics validation based on TopView; Steve Lloyd who has supported this project as my supervisor.
Chapter 1

Ideas Behind TopView

1.1 Motivation

1.1.1 How Common Analysis Framework Helps

A large physics collaboration like ATLAS involves a huge number of physicists analysing the same data. Collaboration between physicists commonly takes place even within one analysis and one such analysis may become a part of larger analysis projects. Eventually such analyses may be published but only after detailed investigation of the results through validation of the analysis procedure. Reproduction of results should also be publicly done to maintain the quality of the outcome. The need for common analysis framework arises very naturally in such an environment.

There is a great deal of overlap among different analyses. For example, clustering algorithm for calorimeter cells is used by almost all physics analyses. In addition, there may be several algorithms to do the same job differently and the analysis framework should be flexible enough such that one can just replace one algorithm by another, leaving the rest untouched. It is also required that the same algorithm can be easily used by many people and it need also be possible to write a new algorithm when one wishes to extend the framework.

Therefore, construction of common analysis framework requires a strategy to accommodate flexibility requested by the analysis performed in a collaborative environment. Despite the need to achieve complex tasks, the interface of the framework has to be simple to improve efficiency of analysis development. It must be designed so that common analysis tasks are available to all. Yet the framework has to be flexible so that new components can be added to it without duplicating the efforts that have already been made.

1.1.2 Athena Framework and Event Data Model

The Athena framework is the ATLAS software framework used for most part of the offline computing. It is a dynamic modular framework where components can be added to it with runtime configuration. The type of algorithms run by
Athena varies broadly, including Monte Carlo event generation, Geant4 detector simulation, event reconstruction, and physics analysis. After each step of data processing, the data (either simulated or real) are written out in Root format using the Pool technology and the contents can be read back to Athena algorithms through a single interface called StoreGate. The end of the chain is particularly important for physics analysis with output files called ESD (Event Summary Data) and AOD (Analysis Object Data). ESD contains the full information from event reconstruction though, because of its size (∼MB per event), only smaller part of the information is made available more globally as AOD. One aim of the event data model using ESD/AOD is that the right amount of information is in AOD so that most physics analysis but detailed detector investigation can be carried out with it. Supposing such goal is fulfilled, how do we analyse AOD?

1.1.3 ATLAS Analysis Model and Scope of TopView

Since the amount of data produced by ATLAS is so vast to fit in a single computer, both in terms of processing power and disc storage, the Grid computing facility is constructed to analyse the data acquired by the detectors on LHC. ESDs will be available only on the Tier 1 centres though AODs will be made available in all Tier 2 sites. Analysis tasks will be sent to and executed at those locations which hold the datasets requested by the task. The analysis carried out in these jobs can vary from a simple analysis with little processing assuming further local analysis on output DPD (Derived Physics Data), to a complete analysis with output histograms ready to be published. Most analyses are in between these two categories, starting from former and approaching to latter gradually as they grow more mature.

TopView is a physics analysis package which runs on the Athena framework. It aims to be the common AOD analysis framework and its capability ranges from copying AOD information into DPD to doing a complete analysis. Its role in the analysis model lies between the in-framework analysis using Athena and local (using DPD produced on Grid) analysis using Root. As the figure shows, these are the two large functional components of ATLAS analysis model: writing of analysis using TopView (typically done on CERN computing node or on a local machine with Athena installation), submission of TopView jobs to Grid to produce DPD and further local processing using Root.

Although analysis using the framework and Grid lacks interactivity, one of the major motivations for moving a greater part of analysis to a common framework like TopView is to standardise physics contents. It is frequently necessary for the members of the collaboration to agree on using the same methods for parts of their analysis. For example, definition of the analysis objects (electron, particle jet, b-tagged jet) used in an analysis need to be agreed on by all members of the same analysis. Another important aspect of using common framework is the ability to compare results from different analyses. Such comparison becomes necessary when two analyses shows significant discrepancy. Having components of each analysis available through the TopView interface allows one to replace parts of one analysis by ones in another analysis. For example, one analysis may have used different jet calibration from another...
but having both available as modules in TopView, one can easily assess the effect of different calibration to the final result. Furthermore, availability of such modules in a common framework improves the efficiency of the collaboration as a whole since work done by one can easily be passed to another. These are the considerations that defines the requirements and the role of TopView.

1.1.4 Implementation of Common Analysis Framework

Analysis tools necessary to analyse AOD (and ESD) are developed within the activities of PAT (Physics Analysis Tools) working group. The conventional approach for developing common framework is to create a library of algorithms that can be called in order. While this approach can be found in any analysis framework which requires modularisation, the EventView framework takes this further by exploiting Athena’s ability to dynamically instantiate arbitrary tools using runtime configuration. Flexibility of this framework is inherent in this design since the dependencies between components of the framework are kept small and each tool is self-contained. This is enhanced by the design of the tools based on object oriented hierarchy which allows optimised abstraction of tools’ functionality.

What EventView framework provides is an environment which scales with the complexity and the size of the analysis where specialisation can be achieved by a set of independent tools. TopView is therefore a collection of EventView compatible tools useful for top physics analysis in particular. It also includes a set of configurations which specifies the behaviour of existing tools since the EventView framework provides a collection of fairly general tools which can be configured to work in different contexts.

In summary, TopView is a collection of tools which extends the EventView framework. Such a specialised analysis package is necessary to support communication within the group where standardisation of tools and convention is meaningful and practical. By design the framework also have a simple mechanism to allow communication between different groups with their own specialised packages as described in the next section.

1.2 Design

1.2.1 EventViewAnalysis Framework

The TopView package can be simply described as a set of tools and modules which extends the EventView framework. Though details of the EventView framework is beyond the scope of this text, the concept of tools and modules should be introduced here to show how this is done. The figure shows a very crude illustration of an analysis which can be written using EventView. At this point EVTool (or simply Tool) can be thought of as the smallest building block of the analysis and EVModule (or simply Module) as secondary building block which consists of EVTools. An analysis, in this framework, can therefore be thought of as a collection of Tools and Modules chained together in a specific manner.

Tools, being the smallest component, is typically a very simple entity and are only meant to do a single task, though some Tools are more complicated than others. In fact, there’s no limitation as to how complex one Tool should be: one can write the whole of his/her analysis in one tool. However, from the point of view of efficient modular analysis,
there are several considerations. If one kind of operation is conducted in an analysis repeatedly, maybe with slightly
different configuration, that operation should be isolated into a separate Tool so that duplication of code can be
minimised. If one part of an analysis is useful for various purposes, including other analyses, again it should become
one Tool so that it can be shared with others. Even in other cases, it may still be useful to put parts of the analysis
into separate Tools to aid organisation of the analysis. At the basic level, one Tool can be thought of as one action.
Often there are practical issues related to what a Tool should do: one tool should become context independent as
much as possible for it to be more useful and applicable in many situations.

Modules are similar to Tools as one independent component of the analysis. They are meant to represent something
more sophisticated than Tools. Typical actions involved in Tools are e.g. “Calculate this variable”, “Combine these
objects” and “preselect this kind of objects”. One can form more meaningful, but context dependent, operations out
of these basic building blocks such as “Leptonic top reconstruction”, “Do preselection of all objects” and “Calculate
this variable for the daughters of W”. One can see that there are several purposes to Modules:

- combine multiple Tools to form a larger action,
- configure one or more Tools to behave in a specific way other than their default, and
- group related actions into one place.

It is clear from the above that functionality of the EventView framework scales very naturally. To add new features,
one can simply write a new Tool in case no combination of existing Tools can give the same result, or write a new
Module otherwise. TopView therefore consists of EVTTools and EVModules as the main part of the package. One
thing that has not been explained is the jobOptions. Once Tools and Modules are ready, they are finally scheduled to
an Athena job via jobOptions. In the current framework, jobOptions are little more than putting Tools and Modules
in order though there are several other complexities involved. This will be explained in a later section.

### 1.2.2 Tools and Modules

![Diagram of EVTTools and EVModules]

EVTTools and Modules are treated equivalently under the new configuration scheme. The main algorithm sequencer
called EventViewToolLooper takes the list of Tools and Modules in sequence. One of the main differences between
them is that Tools are written in C++ and Modules are written in Python. In Athena C++ is the main language for
writing algorithmic part of the application. Python on the other hand is an interpreted language which is not suitable
for such purposes for performance reasons. On the other hand, Python is a very suitable language for job steering
and configuration due to its interactivity. Therefore, in the EventView framework, most algorithmic operations are
implemented in C++ Tools. Python part provides manipulation of these Tools: one can dynamically instantiate
Tools through runtime configuration (without re-compilation). These differences also contribute to making decision
of what to be done in Tools and Modules.
1.2.3 PhysicsView Packages and HighPtView

As described in the previous section, one can extend the EventView framework by constructing a set of Tools and Modules relevant for particular purposes. It is by this mechanism that “PhysicsView” packages were constructed and several such packages were developed in physics groups: TopView for top quark physics, SUSYView for SUSY group and a few other packages for individual Higgs analysis. It became apparent that there was much in common between these packages; although all packages made full use of the standard set of EVTools and EVModules, additional efforts were made to develop Derived Physics Data (DPD).

Objects in AODs are only loosely selected and there can be much ambiguity in what one selects to start his/her analysis with. Loose selection in AOD also means that there are significant overlaps between different types of objects, e.g. most of the objects reconstructed as electrons are also reconstructed as jets. EventView comes naturally into the picture: at the beginning of AOD analysis, one need to define the “view” of the event by selecting objects and removing overlap. Although details of selection criteria may differ from one group to another, given the nature of “High Pt” physics (including Higgs, Top, Exotics, and SUSY), a common baseline selection cuts could be derived which includes the objects of interest for most analyses in this subject. This was done in consultation with the detector performance group. The HighPtView package contains this information in the form of EVModules (as parameters configuring Tools). This package also includes all Modules necessary to produce DPD under HighPt requirements, which includes object inserter Modules, variable calculator Module and ntuple dumping Module. This provides a common set of machinery for DPD production in all PhysicsView packages and the format of the DPD can be kept the same though they may differ in contents.

Each physics group may still produce their own DPD with more specialised or different object selection and additional variables. A part of this can now be done by providing their own selection criteria to HighPtView Modules (without constructing their own Modules.) Additional Tool and Modules are added to the package as before and no limitations are imposed to extend the framework.

1.2.4 Physics Knowledge and Configuration

The top physics group produces their own version of ntuples through this mechanism using TopView. Detailed discussion with performance groups produced a baseline cuts somewhat more specific compared to the HighPtView. In addition, a \( \bar{t}t \) reconstruction analysis is performed and included in the ntuple. This analysis is taken from the “Commissioning Analysis” which is relevant in the early stage of data taking and not only that it is useful in several purposes including determination of jet energy scale and missing et calibration, it forms a reference to all \( tt \) analysis within the group. Other analyses are also available in TopView and the performance of these are shown in a later section. By design, the addition of these features is trivial and one can quickly incorporate them in the existing TopView job though runtime configuration.
1.3. USE CASES

1.2.5 JobOptions

It has been explained that TopView is essentially a collection of Tools and Modules, though one needs to have a job steering, or jobOption, which assembles all the components. In the EventView framework, jobOptions are relatively simple entities. They merely specify input files and output files and schedule Tools and Modules. However, there are additional material necessary to support broader range of use cases. For example, one can run fast simulation on-the-fly to process generator output files without intermediate processing. Such functionality can be provided by additional job configurations not necessarily within the EventView framework. These jobOptions are useful tools for analysis in any case and included in TopView.

1.2.6 Distributed Analysis

Distributed analysis is a central part of any analysis in ATLAS. Considering the amount of data recorded by ATLAS (~Peta Byte), it is impractical to try and process output data on a local resource. The grid service called PANDA provides a robust user front end for Athena job submission. Any Athena job can be sent seamlessly (users just need to use pathena rather than athena with a couple of extra options specifying input data) to the grid resource and the result can be obtained through distributed data management (DDM) tools known as dq2. Given these features of PANDA, TopView needs very little extension to make the most of this service since it is already an Athena package. It does, though, have a small wrapper tools to help submission of TopView jobs. This includes some additional mechanism to organise relevant datasets. Under the rapid processing cycle of MC data like in CSC production, a number of jobs need to be submitted repeatedly with numerous input datasets. Therefore, detailed list of relevant AOD and EVNT files are kept and used as a data card to batch submission.

1.2.7 Root Analysis

Though it is not the main scope of TopView to support analysis after Athena and DPD production, Root analysis is another main component of physics analysis in the ATLAS analysis model. TopView includes a simple Root macro with support for event loop so that local analysis using TopView ntuple can be started.

1.3 Use Cases

In principle, there is nothing to stop one from doing full analysis in TopView. Athena has a support for interactive analysis which is suitable for later analysis stage after DPD production as well. One of the main practical limitations to this approach is the speed. The current event processing rate is around 1 to 10 Hz in a typical EventView analysis. Performance profiling revealed that much overhead is created from AOD access speed. Work is under way to resolve this issue by separating transient and persistent representation of the object classes and speed increase by a factor of 10 is expected in I/O access. Distributed analysis is another solution to efficiency issues. Performance scales linearly with number of available machines available to physics analysis though this does not improve interactive analysis.

Athena algorithms are optimised for event-level processing. While this is suitable for the purpose of event simulation and reconstruction, physics analysis also requires sample level analysis with global view of the sample over a large number of events. Again, it is possible to do such analysis within Athena, but it is unpractical, specially under slow turn-around.

The current use case of TopView is centred around these areas and the development of the package is related to finding an optimal working point. Suppose one wants to extract jet energy scale from $t\bar{t}$ events using kinematic fit exploiting the kinematic constraints within the event. Kinematic fitter can be executed every event to calculate an event level estimator for jet energy scale possibly under multiple combinatorial assignments. One can then try to extract energy scale from a large sample e.g. using maximum likelihood fit. Local Root analysis is suitable for the latter part of this analysis.
1.3.1 Types of Analysis

It is instructive to divide physics analysis into stages to explain the issue in more detail. There are very many things that are called “analysis” and it is often confusing to just say “I do my analysis using !TopView” etc.

1. **Pre Physics Analysis**: Event reconstruction usually done in central production system and the result of this is AOD (and ESD). Some components of event reconstruction, though, is used in further analysis to refine, or redefine objects after further processing.

2. **Baseline Analysis**: The first step in AOD is to do preselection and overlap removal. In EventView analysis, this task is done by inserters. In addition, there can be further processing such as calibration, b-tagging, clustering, rerunning of jet reconstruction algorithm etc.

3. **Event Analysis**: Analysis which is done event-by-event and it consists of very many elements. This includes simple event variable calculation (e.g. sphericity and aplanarity), truth match, object reconstruction (top and W reconstruction), event selection and so on. In the current scheme, this is the grey area which is partly done by Athena though in many cases, most of this is done in Root analysis.

4. **Full Sample Analysis**: Per sample analysis after full sample has been processed. It uses the information from the whole sample and usually done in Root but in some special cases done in Athena. This stage of analysis includes e.g. plotting histograms from trees, fitting templates to histograms, generating toy MC for ensemble test, and so on. This part of analysis requires a very high degree of interactivity.

The present working point is to process, using Grid or otherwise, up to some of event analysis in Athena and save the results in Root ntuple. The ntuple is then read in local Root analysis for further processing. For the reasons explained in previous sections, it is beneficial to move larger part of one’s analysis into a common framework such as TopView and this is where much of the current effort lie. For example, useful information can be added to baseline analysis such as pre-calculated calibration constants which users can apply when they read the ntuple in an Root analysis. It is also possible to replace, before DPD production, non-calibrated jets with calibrated to gain more consistency. Inclusion of basic \( \bar{t}t \) analysis is beneficial because very many people are working with this channel under different context (mass measurement, cross section measurement, jet calibration, missing et calibration etc.) but top reconstruction is usually common to all.

1.3.2 Production of Common DPD

One of the main use case of the TopView package is the production of group DPD. As explained in the previous sections, much of the capability of producing DPD come from HighPtView. In addition to this, there are several event variables and event level analysis as explained in the previous section. The “default” JobOption contains the definition of the analysis contents to be included in the common ntuple production jobs and are submitted to distributed analysis using the PANDA service. Relevant datasets are processed on demand whenever new samples arrive and additional processing required with different set-up are also submitted when needed. A large number of ntuple datasets are created from this effort and distributed through distributed data management (DDM) tools. This enables users to download the ntuple datasets to their local resource for further analysis.

The contents of common ntuple were finalised though discussion on the collaboration meetings in consultation with the performance groups. Some selection cuts were optimised for top analysis and refined though iteration. New variables or analyses are added when needed and development of the tools are aided by contributions from the group members in various forms including discussions in the collaboration meeting and actual implementation of the tools. Several private Root analysis frameworks are now compatible with the format of the TopView ntuple such as SFrame. Comparison of results from different analysis is simplified by using the same ntuple produced with standard selection criteria though it is not mandatory for an analysis to be based on TopView ntuple. There are several other Athena based analysis packages used in various analysis. Comparison of results in this case is slightly tedious since one has to check that every step of the analysis is compatible with what is done in TopView. Therefore, TopView can produce transient StoreGate containers of selected objects. This means that other Athena analysis can access objects preselected by TopView in the same way as reading standard AOD files and the rest of the analysis can stay completely unmodified (Figure below).
1.3 USE CASES

CHAPTER 1. IDEAS BEHIND TOPVIEW

The current form of the default ntuple job is described in the appended document “Common Ntuple Reference” including the details of the preselection and how to produce the default or modified ntuples.

Size of Common Ntuple

Although the size of DPD is not currently a well defined issue, it is useful to indicate this in relation to the contents of DPD. The size and time of execution quoted here are for one event measured by averaging over 250 events.

- Fully reconstructed object information only: 3.9Kb / 0.25s
- Add trigger decision and trigger match: 4.4Kb / 0.3s
- Add detailed trigger information: 10.9Kb / 0.42s
- Add atfast information: 12.5Kb / 0.46s
- Add truth information (only selected truth and match): 21.1Kb / 0.79s
- Add full truth information: 95.9Kb / 1.30s

Size of the AOD used for the above ($\bar{t}t$) is 388 Kb per event, though the size specified by computing model is 100Kb and is currently under active revision. Since full truth information occupies nearly 80% of the total contents and is not used by all analyses, the default contents do not include this.

1.3.3 Physics Validation

![Physics Validation Graphs]
Since certain level of standardisation has been achieved through the TopView framework within Athena, it is a useful tool to be used for the on-going physics validation effort. Along with the ntuple, the default job produces a set of validation histograms (Figure). These histograms are now being integrated into the Run Time Tester (RTT). RTT automatically processes validation samples and compares the histograms produced with reference histograms on nightly basis. This allows us to continuously monitor the performance of the event reconstruction and the TopView package itself. Besides the automatic validation, manned validation effort is taking place making important contribution to the Physics Validation group. Here, validation sample (sample A) are processed through PANDA when they become available and ntuples are processed locally and compared with previous results.

### 1.3.4 Future Use Cases

The current output of TopView analysis (DPD) is Athena-Aware Root ntuple which provides navigation capability (to go back to AOD and ESD) to the standard “flat” ntuple containing integers and float numbers (and arrays of them). Current developments in Event Data Model will enable reading of structured objects (such as an object of class `Electron`) directly from Root. This will give more flexibility between Athena analysis and Root analysis since the difference between DPD and AOD will merely be in their contents but both of them can be read by both Athena and Root. This will not alter the main objective of TopView and the current use cases such as common DPD production: there will be a need for common framework for analysis, common preselection of objects and common tools to calculate variables even after structured DPD is introduced. With further development in EventView it will also become possible to write out EventViews in Root-readable format not by copying their contents to flat ntuples. Therefore future developments in EDM will enable us to lower the hedge between Athena and Root analysis by increasing the portability of the analysis code.

### 1.4 References

- **Athena, The ATLAS Common Framework - Developer Guide** ATLAS Collaboration

### 1.4.1 References on the web

- **EventView**: The top page for EventView framework. All EventView related information are linked to this page.
- **EventViewConfiguration**: Explains the configuration method used in TopView.
- **SFrame**: A Root framework compatible with TopView ntuple
- **RTT**: ATLAS Run Time Tester
- **Top MC Validation**: Physics validation using TopView
2.1 TopView Ntuple Availability

2.1.1 How to get ntuples

From DQ2

Good instruction can be found on this page: UsingDQ2

Locally From DQ2

This is the recommended method for obtaining the data. If you’d like to know how to install dq2_get / dq2_ls (called dq2 end user tools), please see dq2 installation page. You will need grid middleware on the machine you’ll install dq2 and grid certificate is also necessary.

On Lxplus From DQ2

In case local installation won’t work, use the end user tools installed on afs by GridSetupLxplus.sh found under TopView/run. You’ll need grid certificate. /tmp/yourusername space can be used as temporarily storage for keeping the files before scp to your local machine.

From CASTOR

Most files are on CASTOR as indicated below. If you don’t know how to use CASTOR here’s a five seconds instruction:

- use rfcp instead of cp
- use rmdir instead of ls

and wild card such as * are not allowed. Again, /tmp/yourusername can be used to store the files temporarily before scp to your local machine since rf commands will not work over network.

2.1.2 Currently Available Ntuples

v1213

This is the set of ntuples to be used for CSC analysis there are several types of runs, so please read on

- JobType : description
- MuidTau1p3p, MuidTauRec, StacoTau1p3p, StacoTauRec : The main set of ntuples with 1mm bug fixed.
CHAPTER 2. APPENDIX A: COMMON NTUPLE REFERENCE. TOPVIEW NTUPLE AVAILABILITY

- **MuidTauRecNoFix**:
  - trig1_misal1_mc12.005200.T1_McAtNlo_Jimmy.recon.AOD.v12000601: Run without fix for comparison
  - trig1_misal1_mc12_V1.005200.T1_McAtNlo_Jimmy.recon.AOD.v12000601: This one is simulated with 30um range cut, for comparison
  - Other samples here are were simulated with 30um range cut and do not need 1mm fix
- **MuidTauRecTopo**: Sample with topo cluster jet rather than tower. (only ttbar sample available)
- **MuidTauRecTruthAll**: Limited number of all samples were processed with full truth information. In case full truth is required for check.
- **MuidTauRecNoCut**: No cut or overlap removal applied (1mm bug is aapplied). Only available for 5200 sample.
- **TestingDonotuse**: These are test runs, do not use for physics analysis

To find these datasets on Grid

```
dq2 ls *top*TopView1213*.JobType.*001
nsls /castor/cern.ch/grid/atlas/caf/top/csctopview/
```

v1212

This set was previously the official set of ntuples but due to the bug in LAr range cut (1mm bug), and fix made available on AOD level, everyone using this ntuple should move to v1213. To see the status:

```
dq2 ls *top*TopView1212*003
nsls /castor/cern.ch/grid/atlas/caf/top/csctopview/
```

Files ending in .003 are the most current version. They differ from files ending in .001 in that more AOD’s were available to be used for the .003 files.
There are 4 flavors of TopView1212: MuidTau1p3p, MuidTauRec, StacoTau1p3p, StacoTauRec. These use different muon and tau reconstruction methods.

For old ntuple, please see this page

2.1.3 Datasets processed

Following datasets are being processed. Some are waiting for AOD production. The status is based on the availability of v1209.ver11 ntuples which is the most recent production. (the chart is under construction, please fill in if you can!!)

Note on Status (the status is now based on CSC 12 AOD processing):
- done: Ntuple production done
- part: partly done, some AOD missing
- prog: in progress
- noda: No dataset ready

Note on Link:
- o: official sample page
- d1: dq2 sample page (for CSC11)
- d2: dq2 sample page (for CSC12)
- V: validation page
## TTbar

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5200</td>
<td>T1_McAtNlo_Jimmy</td>
<td>No all-hadronic</td>
<td>461</td>
<td>ood1 V</td>
<td>part</td>
</tr>
<tr>
<td>5201</td>
<td>Mcatnlo_jim_top_pt200</td>
<td>Inclusive, pt(one top)&gt;200GeV</td>
<td>100</td>
<td>ood1 V</td>
<td>prog</td>
</tr>
<tr>
<td>5202</td>
<td>Mcatnlo_jim_top_leptpt120</td>
<td>1 lepton, pt(lep)&gt;120GeV</td>
<td>5.5</td>
<td>od1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5204</td>
<td>TTbar_FullHad_McAtNlo_Jimmy</td>
<td>Only all-hadronic</td>
<td>369</td>
<td>o d1 V</td>
<td>part</td>
</tr>
<tr>
<td>5205</td>
<td>AcerMCTtbar</td>
<td>=&gt;1lepton</td>
<td>460</td>
<td>od1 V</td>
<td>part</td>
</tr>
<tr>
<td>5560</td>
<td>Mcatnlo_jimmy_top170</td>
<td></td>
<td>800</td>
<td>d1 V</td>
<td></td>
</tr>
<tr>
<td>5565</td>
<td>T1_McAtNlo_noUE</td>
<td>No all-hadronic, no underlying event</td>
<td>461</td>
<td>od1 V</td>
<td></td>
</tr>
<tr>
<td>6201</td>
<td>ttbar &quot;l+jets&quot;, mtop = 170 GeV</td>
<td>mtop= 170</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6202</td>
<td>ttbar &quot;l+jets&quot;, mtop = 180 GeV</td>
<td>mtop= 180</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: When using MC@NLO samples, you must use event weight. Calculation of cross section is also affected. More info can be found on TopViewComments. trig1_misall_mc12_v1 sample is with correct LAr range cut (30um rather than 1mm)

## Single Top

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5500</td>
<td>!AcerMC_Wt</td>
<td>e/mu/tau</td>
<td>26.7</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>5501</td>
<td>!AcerMC_schan</td>
<td>e/mu/tau</td>
<td>3.3</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>5502</td>
<td>!AcerMC_tchan</td>
<td>e/mu/tau</td>
<td>81.3</td>
<td>d1 V</td>
<td>part</td>
</tr>
</tbody>
</table>

## W+Jets (ttbar)

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXX</td>
<td>example</td>
<td>example</td>
<td>Alpgen<em>MLM</em>Filter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8240</td>
<td>AlpgenJimmyWenuNp2</td>
<td>pt20_filt4jet</td>
<td>2032<em>0.402</em>0.262</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8241</td>
<td>AlpgenJimmyWenuNp3</td>
<td>pt20_filt4jet</td>
<td>773<em>0.301</em>0.534</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8242</td>
<td>AlpgenJimmyWenuNp4</td>
<td>pt20_filt4jet</td>
<td>273<em>0.252</em>0.780</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8243</td>
<td>AlpgenJimmyWenuNp5</td>
<td>pt20_filt4jet</td>
<td>91<em>0.264</em>0.929</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8244</td>
<td>AlpgenJimmyWmunuNp2</td>
<td>pt20_filt4jet</td>
<td>2032<em>0.402</em>0.020</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8245</td>
<td>AlpgenJimmyWmunuNp3</td>
<td>pt20_filt4jet</td>
<td>771<em>0.304</em>0.276</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8246</td>
<td>AlpgenJimmyWmunuNp4</td>
<td>pt20_filt4jet</td>
<td>273<em>0.229</em>0.576</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8247</td>
<td>AlpgenJimmyWmunuNp5</td>
<td>pt20_filt4jet</td>
<td>91<em>0.264</em>0.840</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8248</td>
<td>AlpgenJimmyWtaunuNp2</td>
<td>pt20_filt4jet</td>
<td>2032<em>0.415</em>0.104</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8249</td>
<td>AlpgenJimmyWtaunuNp3</td>
<td>pt20_filt4jet</td>
<td>771<em>0.303</em>0.373</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8250</td>
<td>AlpgenJimmyWtaunuNp4</td>
<td>pt20_filt4jet</td>
<td>273<em>0.245</em>0.686</td>
<td>d1 V</td>
<td>part</td>
</tr>
<tr>
<td>8251</td>
<td>AlpgenJimmyWtaunuNp5</td>
<td>pt20_filt4jet</td>
<td>91<em>0.264</em>0.866</td>
<td>d1 V</td>
<td>part</td>
</tr>
</tbody>
</table>

## W+Jets (for ttbar study)
### CHAPTER 2. APPENDIX A: COMMON NTUPLE REFERENCE

#### TOPVIEW NTUPLE AVAILABILITY

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>8220</td>
<td>Alpgen.JimmyWenuNp0</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8221</td>
<td>Alpgen.JimmyWenuNp1</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8222</td>
<td>Alpgen.JimmyWenuNp2</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8223</td>
<td>Alpgen.JimmyWenuNp3</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8224</td>
<td>Alpgen.JimmyWenuNp4</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8225</td>
<td>Alpgen.JimmyWenuNp5</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8226</td>
<td>Alpgen.JimmyWmunuNp0</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8227</td>
<td>Alpgen.JimmyWmunuNp1</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8228</td>
<td>Alpgen.JimmyWmunuNp2</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8229</td>
<td>Alpgen.JimmyWmunuNp3</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8230</td>
<td>Alpgen.JimmyWmunuNp4</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>8231</td>
<td>Alpgen.JimmyWmunuNp5</td>
<td>pt(l)&gt;15</td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
</tbody>
</table>

**W+Jets (for single top study)**

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>6210</td>
<td>W + 0 parton</td>
<td>d1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6211</td>
<td>W + 1 parton</td>
<td>d1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6212</td>
<td>W + 2 parton</td>
<td>d1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6213</td>
<td>W + 3 parton</td>
<td>d1 V</td>
<td>noda</td>
<td></td>
</tr>
</tbody>
</table>

**W+Jets** NOTE: the Alpgen sample here have very tight cuts!! (WJet sample for top study is not ready yet, some atlfast preview for 6210-6213 are available under v1114_Fast)

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5100</td>
<td>JimmyWenu</td>
<td>lepton eta &lt; 2.7, pt&gt;10GeV</td>
<td>11225</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5101</td>
<td>JimmyWmunu</td>
<td>lepton eta &lt; 2.8, pt &gt; 5GeV</td>
<td>12318</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5102</td>
<td>JimmyWtautau</td>
<td>leptonetic tau decays pt(e/mu)&gt;5GeV, eta</td>
<td>3135</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5104</td>
<td>PythiaWenu</td>
<td>lepton eta 10GeV</td>
<td>10900</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5105</td>
<td>PythiaWmunu</td>
<td>lepton eta 5GeV</td>
<td>11946</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5106</td>
<td>PythiaWtauunu</td>
<td>leptonic tau decays, pt(e/mu)&gt;5GeV, eta</td>
<td>3400</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5107</td>
<td>!pythia_Wtauhad</td>
<td>Hadronic tau decays, pt(tau)&gt;10GeV, eta</td>
<td>5540</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5250</td>
<td>McAtNloWenu</td>
<td>lepton eta 10GeV</td>
<td>12798</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5254</td>
<td>McAtNloWplusenu</td>
<td>lepton eta 10GeV</td>
<td>12798</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5251</td>
<td>McAtNloWmunu</td>
<td>lepton eta 5GeV</td>
<td>13861</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5255</td>
<td>McAtNloWplusmunu</td>
<td>lepton eta 5GeV</td>
<td>13861</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5256</td>
<td>McAtNloWplusmunu</td>
<td>lepton eta 5GeV</td>
<td>13861</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5260</td>
<td>McAtNloWplusmunu</td>
<td>lepton eta 5GeV</td>
<td>13861</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5270</td>
<td>!PythiaWenu_pt100Fixed</td>
<td>pt(W)&gt;100GeV, e above</td>
<td>2.69</td>
<td>d1</td>
<td>noda</td>
</tr>
<tr>
<td>5271</td>
<td>!PythiaWmunu_pt100Fixed</td>
<td>pt(W)&gt;100GeV, mu above</td>
<td>2.74</td>
<td>d1</td>
<td>noda</td>
</tr>
</tbody>
</table>

**Wbb**

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5280</td>
<td></td>
<td></td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>5281</td>
<td></td>
<td>111</td>
<td>d1 V</td>
<td>noda</td>
<td></td>
</tr>
</tbody>
</table>

**Diboson**
### Zll

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5144</td>
<td>PythiaZee</td>
<td>$M_{ll}&gt;60$ GeV, 1e eta 10 GeV</td>
<td>1432</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5145</td>
<td>PythiaZmumu</td>
<td>$M_{ll}&gt;60$ GeV, 1mu eta 5 GeV</td>
<td>1497</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5146</td>
<td>PythiaZtautau</td>
<td></td>
<td>77.0 (incl. filter eff.)</td>
<td>d1 V</td>
<td></td>
</tr>
</tbody>
</table>

### Di Jet

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5031</td>
<td>Jimmy_jetsJ2</td>
<td>pt: 35-70 GeV</td>
<td>9.33E+007</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5032</td>
<td>Jimmy_jetsJ3</td>
<td>pt: 70-140 GeV</td>
<td>5.88E+006</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5033</td>
<td>Jimmy_jetsJ4</td>
<td>pt: 140-280 GeV</td>
<td>3.08E+005</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5034</td>
<td>Jimmy_jetsJ5</td>
<td>pt: 280-560 GeV</td>
<td>1.25E+004</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5035</td>
<td>Jimmy_jetsJ6</td>
<td>pt: 560-1120 GeV</td>
<td>360</td>
<td>d1 V</td>
<td>noda</td>
</tr>
<tr>
<td>5036</td>
<td>Jimmy_jetsJ7</td>
<td>pt: 1120-2240 GeV</td>
<td>5.71</td>
<td>d1 V</td>
<td>noda</td>
</tr>
</tbody>
</table>

### FCNC

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5510</td>
<td>toprex_tt_bWqgamma</td>
<td>$t\bar{t}\rightarrow bWqZ$ ($W\rightarrow l\nu$; $Z\rightarrow ll$; $l=e,mu$)</td>
<td>-</td>
<td>o d V</td>
<td>part</td>
</tr>
<tr>
<td>5511</td>
<td>toprex_tt_bWqZlep</td>
<td>$t\bar{t}\rightarrow bWqZ$ ($W\rightarrow l\nu$; $Z\rightarrow ll$; $l=e,mu$)</td>
<td>-</td>
<td>od1 V</td>
<td>part</td>
</tr>
<tr>
<td>5512</td>
<td>toprex_tt_bWqg</td>
<td>$t\bar{t}\rightarrow bWqgluon$ ($W\rightarrow l\nu$; $l=e,mu$)</td>
<td>-</td>
<td>od1 V</td>
<td>part</td>
</tr>
<tr>
<td>5513</td>
<td>toprex_tt_bWqZhad</td>
<td>$t\bar{t}\rightarrow bWqZ$ ($W\rightarrow qq$; $Z\rightarrow ll$; $l=e,mu$)</td>
<td>-</td>
<td>od1 V</td>
<td>part</td>
</tr>
<tr>
<td>5514</td>
<td>toprex_st_qgamma</td>
<td>single-top-$q\gamma$ (t channel)</td>
<td>-</td>
<td>od1 V</td>
<td>part</td>
</tr>
<tr>
<td>5515</td>
<td>toprex_st_qZ</td>
<td>single-top-$qZ$ (t channel; $Z\rightarrow ll$; $l=e,mu$)</td>
<td>-</td>
<td>od1 V</td>
<td>part</td>
</tr>
</tbody>
</table>

### Z'→ttbar

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>6230</td>
<td>$Z'$ (700 GeV) → tt</td>
<td></td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6231</td>
<td>$Z'$ (700 GeV) → tt</td>
<td></td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6232</td>
<td>$Z'$ (700 GeV) → tt</td>
<td></td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6233</td>
<td>$Z'$ (700 GeV) → tt</td>
<td></td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
<tr>
<td>6234</td>
<td>$Z'$ (700 GeV) → tt</td>
<td></td>
<td>od1 V</td>
<td>noda</td>
<td></td>
</tr>
</tbody>
</table>

### Minimum Bias

<table>
<thead>
<tr>
<th>data set</th>
<th>short name</th>
<th>info</th>
<th>xsec (pb)</th>
<th>Link</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>pythia_minbias</td>
<td></td>
<td></td>
<td>d1 V</td>
<td>nodanoda</td>
</tr>
</tbody>
</table>

**Note**
- Cross section includes filter efficiency.
- Lumi/10K is in the units of fb$^{-1}$

For more details, please see:
- DC3 requests (sm)
2.1.4 Naming Convention:

The names of the ntuples is of the following form:

\[ \text{user.top.JobType.SampleNo.SampleDescription.Version} \]

- **JobType**: TopView$\text{Ver}_\text{JobName}$, where $\text{Ver}$ is the version of TopView and $\text{JobName}$ is as explained below.
- **SampleNo**: as in 5200 for ttbar etc.
- **SampleDescription**: such as AlpgenJimmyWenuNp5, same as the one on dataset.
- **Version**: runs from 001, 002 and so on. Will increase when the job failed or there's problems with the ntuple.

2.1.5 Jobs types:

Currently, the one type of ntuple is produced using TopView (there used to be 3 but this one was by far the most useful).

- **!Default (Analysis_topOptions.py, TopDefault_jobOptions.py)**: This job inserts selected objects with overlap removal and copies kinematics and other type specific information from AOD. FullReco0 and Truth0 are the trees containing Reconstructed object information and Truth information respectively. Truth0 will contain daughters of top decay only. This job does default ttbar analysis (with b tag) and also does truth match from "Truth to Reco" and "Reco to Truth". Details on the ntuple can be found from the TopViewNtuple page.

2.1.6 Check Status and Retrieve Output

The ntuples are produced on PANDA distributed analysis system. This means that an output will become available as a DQ2 dataset which anyone can retrieve using dq2 end user tools. On Lxplus, you can use the setup script GridSetupLxplus.sh (found in run directory, you will need grid certificate) to set up to use those tools.

To retrieve a dataset, do the following:

```
$ dq2 get -rv user.top.TopView1114_Default.005200.T1_McAtNlo_Jimmyv11000505.005
```

More information on dq2 tools can be found in here

You can also check the status of current production on the Panda monitor. If you cannot find files inside a dataset, maybe the job has failed and not been re-run. Please report this and ask for re-run or you can do it yourself and update this page, see TopNtupleDIY.

2.1.7 Reading Output

The output ntuples are standard AANtuple format. You can read them in standalone ROOT or put it back in Athena for further analysis. The ntuple dumping is using EventViewNtupleDumpers and details can be found on this page. Currently, by default, only event-based tree is used but candidate based tree could be used too. ROOTNtupleHowTo outlines simple interactive analysis using ROOT.
2.2 Object Preselection in TopView

2.2.1 Introduction

The object selection in TopView is the result of several discussions and studies. It is meant to serve as a good selection optimised for most top physics analysis though it is not meant to contain all information available from AOD.

The information here is a direct mirror of what’s available under the TopViewInserters module.

The discussion which took place to come up with the following numbers is on this HyperNewsthread.

2.2.2 Use Preselected Objects in External AOD Analysis

Objects selected by TopView can be used in other AOD analysis by exporting the selected containers. By this, selected objects are recorded to StoreGate and subsequent analysis can be run only by changing the input container name. To use this facility, run the top option, TopView/Analysis_topOptions.py with

```
exportContainer=True
# if you want to study your own overlap removal, then use also: removeOverlap=False
```

More details on how to run top option can be found on CustomizeTopView but in short, you can run your analysis like

```
athena -c "exportContainer=True" TopView/Analysis_topOptions.py
YourAnalysis_jobOption.py
```
Names of the container are the same as AOD container name except it has prefix TV. For example, ElectronCollection will become TV.ElectronCollection. Beware that TopView only processes one ParticleJet container at a time so the container available depends on the job setting. For eg, if you decide to use Muid muons, then you will not find TV.StacoMuonCollection.

2.2.3 Object Insertion Order and Reconstruction Algorithm

Overlap between objects are removed by inserting objects in order. If another object is already inserted within the defined cone of deltaR (dR), then the object will not be inserted. Insertion is done in the following order: Muon, LooseMuon, Electron, LooseElectron, Photon, TauJet, ParticleJet. Overlap with muon and particle jet is not removed.

For Muon, Staco and Muid are available under separate ntuple. (preselection is the same) For TauJet, Tau1p3p and TauRec are available under separate ntuple. For ParticleJet, only Cone algorithm dR=0.4 is used.

Atlfast objects are selected with et cut 20GeV (except loose lepton which have 10GeV cut).

Muon Selection
- "etCut": 20*GeV,
- "onlyHighPt": True,
- "isolationCone": 0.20,
- "absolutelsolationCut": 6*GeV,
- "relativesolationCut": 0,
- "useChi2FromCombinedMuon": False,
- "chi2NdofCut":20,
- "makeEtaCuts": False,

dR=0.1 is used to remove overlap with other objects (in fact not relevant by default since muon is inserted first). Same for loose muon.

Loose Muon Selection
Same as muon selection except
- "etCut":10*GeV,
- "onlyHighPt": True, # trying to include lowpt, under investigation
- "absolutelsolationCut":14000*GeV # no isolation cut

Electron Selection
- "etCut":20*GeV,
- "onlyEgamma":True,
- "useIsEM":True,
- "useTRT":False,
- "useNN":False,
- "useIsolation":True,
- "isolationCone":0.20, # in deltaR
- "absolutelsolationCut":6*GeV,
- "makeEtaCuts": False,

dR=0.1 is used to remove overlap with other objects. Same for loose electron.
2.2.4 Loose Electron Selection

Same as electron selection except

- "etCut":10*GeV,
- "onlyEgamma":False, # currently under testing

Photon Selection

- "etCut":15*GeV,
- "useIsolation":True,
- "isolationCone":0.45,
- "absolutIsolationCut":10*GeV,
- "makeEtaCuts": False,

dR=0.1 is used to remove overlap with other objects. isEM is not used in 12.0.x since tuning was not done for this release. Instead, tuned values available in EVP Photon Inserter is used which gives more reasonable reco efficiency.

TauJet

- For TauRec:
  - "etCut":15*GeV,
  - "ParameterCutList":[TauJetParameters.logLikelihoodRatio],
  - "ParameterCutValues":[6.],
  - "RequireChargeToBeOne":False,
  - "HadronicEnergyFraction":0.1,
  - "makeEtaCuts": False,
- For Tau1p3p
  - "ParameterCutList":[TauJetParameters.discriminant],
  - "ParameterCutValues":[0.5], # ie, discriminant = 1
  - "onlyTauRec": False,

dR=0.1 is used to remove overlap with other objects.

ParticleJet Selection

- "etCut":15*GeV,
- "deltaRCut":.3,
- "makeEtaCuts": False,

Objects with

- "useWeight":True,
- "weightCut":3

are marked with BTagged label and available as PJet_BTagged etc.

dR=0.3 is used to remove overlap with other objects.

2.3 TopView Ntuple Variables

Go to TopNtuple11 for information on the ntuple created with release 11
2.3.1 Top Ntuple Trees and Variables

There are several trees and two types of histograms in TopView ntplues.
- **CollectionTree**: Contains event information such as event number. Hardly useful.
- **FullReco0** (previously EV0): Reconstructed objects tree.
- **FullRecoAna0**: Results of the default ttbar analysis. (previously in EV0)
- **FastSim0**: Atlfast objects tree.
- **FastSimAna0**: Results of default analysis on fast simulation objects.
- **Truth0**: Truth object tree.
- **TruthAna0**: Further info on truth specialized for top analysis.
- **RecoHistogram**: Validation histograms from reconstructed EventView.
- **TruthHistogram**: Validation histograms from truth EventView.

FullReco0 and Truth0 are the main trees containing selected information from AOD.

TopView Ntuple is a large collection of variables by default. Its basic format is `Prefix_(2ndPrefix)_name`. For vector variables, the ones with the same Prefix have the same number of components and the same index in these vectors come from the same object. For eg:
- **El_N**: is not a vector, it is the number of electrons and you should have number of entry = number of events.
- **El_e[]**: is the energy of the electrons.
- **El_eta[]**: is the electron eta.

([] used to indicate vector values)

So El_p_T have number of components El_N and El_e[0] corresponds to the energy of the highest Pt electron (ie, the container is pt ordered) and El_eta[0] is the eta of that electron. Please note that there’s at least one "dummy" variable saved to the vector values (such as El_e[] and El_eta[]), usually zero. So when El_N is 0, then El_e[] has one component with a dummy variable.

See this part of the tutorial which illustrates this in more detail.

2.3.2 FullReco0 Tree

(There’s another documentation by Patric Ryan: available here)

**Event variables**

Some variables have no prefix.
- **MissingEt** (Ex, Ey): Missing Et. From MET_Final by default.
- **eventWeight**: Event weight from MC generator.
- **HT**: HT, sum of Et of leptons and jets and missing et.
- **Event_Sphericity, Event_Aplanarity**: axes are also provided as TVector3 objects.
2.3. TOPVIEW NTUPLE VARIABLES

CHAPTER 2. APPENDIX A: COMMON NTUPLE REFERENCE

Prefix

- \( \text{El}_- \): Electron
- \( \text{Mu}_- \): Muon
- \( \text{Tau}_- \): TauJet
- \( \text{Ph}_- \): Photon
- \( \text{PJet}_- \): Particle Jet
- \( \text{MET}_- \): Missing Et
- \( \text{VectSumAll}_- \): Vector sum of all objects (not including Missing Et)

**Note** b-tagged jets are now indicated by \( \text{PJet}_BTagged \) variable. PJet will contain both TJet and PJet as previously called and the choice can easily be made to use b-tag information or not.

Second Prefix

- \( \text{L1}, \text{L2}, \text{EF} \): Matched trigger objects.
- \( \text{Tru} \): Reco matched to truth. \( \text{El}\_\text{Tru\_status}[] \) means the status code of the true electron matched to the given reconstructed electron.
- \( \_\text{TruJet} \): Reco matched to truth jets. Available for PJet objects only.

Value Matched[] should be used when dealing with matched object. For example, if you want to plot matched object variables (say you want to plot energy scale):

```plaintext
tree.Draw("\text{El\_e}/\text{El\_Tru\_e}", "\text{El\_Tru\_Matched==1}")
```

```plaintext
tree.Draw("\text{PJet\_TruJet\_p\_T}", "\text{PJet\_TruJet\_Matched==1}")
```

Variable Names

The object info comes from AOD information. For eg, electrons have isEM for quality cut and jets have calorimeter info. These are dumped using EventView calculator tools. See EventViewUserDataCalculator120x for details.

TRF Tag Info

Kinematic b-tagging based on parameterization of pt and eta is available thanks to B. Clement’s tool. This is useful to estimate fake btag rate when very high statistics is required. Both event weight and jet weights are available as well as random tagging based on the jet weights.

- Event weights are available under: eventWeightIP2D and eventWeightSV1
- Jet weights are available under: PJet.TRFPIDwp3_prefix etc.
- Random tag results are available under: PJet.TRFPIDwp3 etc.

Wp3 means efficiency of 60%. 1excl means one and only one jet tagged. 1incl means one or more jets tagged.

Secondary Vertex Info

Information about secondary vertex is available thanks to J. Walbersloh. * JetSecVtx\_x,y,z: position of the secondary vertex * JetSecVtx\_x,y,z\_SD: error on the xyz values * num_JetSecVertices: number of vertices found.
Trigger Info

Trigger information is available for all types of objects and trigger decisions are also available.
- All reco objects are matched with relevant trigger objects. This is available through second prefix _L1, _L2, _EF
- Trigger decisions are available under PassedL1, PassedL2 and PassedEF.

Calibration Info

Currently, light jet calibration is available (based on truth information.) Based on information on this page
- PJet_MCcalib: Calibration constant for the given jet, using calibration map.
- PJet_MCcalibError: Error given from the above map.
- PJet_MCcalib_Fn: Calibration constant using alternative method (function)

2.3.3 FullRecoAna0 Tree

This tree is the results of the default ttbar reconstruction run in the default job. The contents was previously in EV0. The procedure of the analysis is described on this page. The tree therefore consists of “inferred” objects which are the result of the analysis.

Prefix

First prefix used are the following
- Nu_: Neutrino (reconstructed using W mass constraint)
- HadW_: Hadronic W
- HadTop_: Hadronic top
- LepW_: Leptonic W
- LepTop_: Leptonic top

The information in this tree is fairly straightforward (kinematic info) and detailed explanation is not necessary except the following.

Selection Results

Results of some standard final selection cuts are available. The definition of these cuts are defined in TopViewSelectionmodule.
- tt_LooseCut : ttbar loosen cut result
- tt_TightCut : ttbar tight cut
- tt_CommissioningCut : ttbar commissioning cut
- wg_LooseCut : wgluon single top loose cut
- wg_TightCut : wgluon single top tight cut.

The above are the final result. The following stores the individual cut results:
- tt_LooseCut_bits[] : ttbar loose cut results separately

You can apply cut results separately by using bits from this vector. First bit for the result of the first cut, and so on.
2.3.4 FastSim0 and FastSimAna0

These are equivalent to FullReco0 and FullRecoAna0 but made out of atlfast objects.

2.3.5 Truth0 Tree

(There’s another documentation by Patric Ryan: see link)

The chart below show the contents of this tree:

<table>
<thead>
<tr>
<th>Vtx Filtered</th>
<th>Stable</th>
<th>True jets</th>
<th>True Tau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top and its decay products: top (6), W from top (24), b from top (5), quarks/leptons from W (1,2,3,4,11,12,13,14,15,16). No et cut.</td>
<td>Stable leptons (11,12,13,14,15,16) and photons (22) that except the ones selected above. (et&gt;5GeV for leptons, &gt;15 cut for photons)</td>
<td>True Jets as found in the AOD but removed the ones which overlap with any of the Vtx Filtered and Stable ture particles (15GeV cut).</td>
<td>Visible hadronic decay product of tau leptons. Leptonic decay is ignored (15GeV cut).</td>
</tr>
<tr>
<td>Truth selection in TopView is somewhat complex to collect various truth objects of interest. There are four types of truth available under Tru_ or TruJet_.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Truth from the hard process (Filtered): Vertex filter is used to filter out objects with decay pattern top→W+b→InuW and top→W+b→qqb. It is done separately so that W→Inu and W→qq decay in W+jets sample can also be selected. Z→l+l- decay is also defined. These objects are also available under El_, HadW_, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Other stable truth (Stable): there may also be other stable objects. All stable leptons, neutrinos and photons are inserted if they don’t overlap with Filtered truth.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Truth jet: Truth jets that do not overlap with the above are also inserted. They are inserted with et&gt;10GeV cut to remove objects that are too soft.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Visible tau: Visible hadronic decay product of tau is inserted as composite particle.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Event variables

Some variables have no prefix.

- MissingEtTruth (Ex, Ey): From MET_Truth by default.
- eventWeight: Event weight from MC generator.
CHAPTER 2. APPENDIX A: COMMON NTUPLE REFERENCE

2.3. TOPVIEW NTUPLE VARIABLES

Prefix

First prefix used are the following
- Tru_: Truth (contains 1 and 2 above)
- TruJet_: Truth Jets (contains 3 as above)
- TruTau_: Visible tau hadronic decay product (contains 4 as above), not tau lepton

Second Prefix

* El, Mu, Tau etc: Objects are matched against all types of reco objects, the matched objects are accessible separately. For example you can draw the pt of the electrons reconstructed as particle jet as

\[ \text{tree.Draw("Tru_PJet_p.T", "Tru_PJet_Matched \&\& abs(Tru_pdgId)==11")} \]

More examples:
- TruTau_Tau_: Visible hadronic tau decay product matched to reconstructed tau jet object.
- Tru_Tau: Truth object (1. and 2. as described above) that matched to reconstructed tau jet object.

(note that Tru_ may contain tau lepton but since reconstructed tau jet is the hadronic decay product, one should consider TruTau_Tau as appropriate truth to reco match rather than Tru_Tau with Tru_pdgId 15)

2.3.6 TruthAna0 Tree

Here, only objects found by vertex filter is stored for further investigation. El, Mu and Tau, for example, therefore, are only the ones from W or Z decay.

Event variables
- HT: HT, sum of Et of leptons, photons, quarks, truth jets and missing et truth.

Prefix

First prefix used are the following
- El_: Electron
- Mu_: Muon
- Tau_: Tau (not visible part, but tau lepton)
- Bot_: Bottom quark (only in truth)
- LQ_: Light quark (only in truth)
- Nu_: Neutrino
- HadW_: Hadronic W
- HadTop_: Hadronic top
- LepW_: Leptonic W
- LepTop_: Leptonic top
- VectSumAll_: Vector sum of all objects

Second Prefix
- Reco : Corresponding reco object matched. El_Reco_p.T[] means the pt of the reconstructed electron matched to the given truth electron.
- VHad_Reco (currently Reco_VHad): Only for taus. Taus in this tree are associated to the visible tau decay composite (Tau_VHad_). The composite is subsequently matched to reconstructed tau jets (Tau_Reco_VHad).
2.4 Producing TopViewNtuple

**TopNtupleDIYv11**: instruction for release 11

### 2.4.1 Running the Default Job

Default job creates the default Ntuple. All you need to do is

```
> athena TopView/Analysis_topOptions.py
```

the top option can be configured with an extra jobOption which sets the topOption parameters as described in this link, and you should do

```
> athena LocalOverride.py TopView/Analysis_topOptions.py
```

This can be further simplified by including the `TopView/Analysis_topOptions.py` in `LocalOverride.py`, that is to write

```
include("TopView/Analysis_topOptions.py")
```

at the end of `LocalOverride.py`. This way you can do

```
> athena LocalOverride.py
```

An example of such local setting can be found under the `run` directory where all the above is already written in.

### 2.4.2 Running TopView Job on Distributed Analysis

The setup file provided under `run` directory `GridSetupLxplus.(c)sh` will setup for grid tools (with afs access only). Source this file (or do your own grid setup) before proceeding.

For job submission, the script, `TopViewSubmission.py` can be used for PANDA job submission (details of panda submission can be found in DAonPanda) to run on dataset available through DQ2. A few submission parameters should be defined (some of them have been shown above). After the end of setting of these options, you should insert `#End`. Therefore, the first part of the submission jobOption (called it `YourSettings.py` here) would look like the following:

```
# your panda username. use "top" only for official production
sub_username="akirashibata"
# special option for production. use " --official " for using "top" user name in official production
sub_special=""
# prefix for this job
sub_version="TopView1212_Test"
# index
sub_tryNo="001"
# type of queue. Use " --long " for only build job to go to long
sub_long="--long"
# number of files per job
sub_nFiles=" --nFilesPerJob 15"
# number of files to process in total
```
CHAPTER 2. APPENDIX A: COMMON NTUPLE REFERENCE

2.4. PRODUCING TOPVIEWNTUPLE

```python
sub_nAllFiles=" --nfiles 10 "
# how many sub jobs to split into (done automatically if left empty) eg. " --
# split=100"
sub_split=
# if using precompiled library dataset, use something like " --libDS user.
akirashibata.lxplus014.lib._001189"
sub_libDS=
# your jobOption, include another if you need to specify some variables
sub_jobOp="TopView/Analysis_topOptions.py"

#End
```

YourSettings.py is now followed by additional customization settings like showed in the following sections to eg. change selection cut etc. Once YourSettings.py has been defined with submission options, all you need to do now is to put that file in the share directory and compile (do make from cmt directory), then issue this command from under your run directory:

```
./TopViewSubmission TopView/YourSettings.py
```

Now you can use Panda monitor to monitor your job. Once your job has finished, you can retrieve the output by using DQ2 commands.

2.4.3 Customizing the Parameters.

One common customization is to change the preselection cut. To fully customize the inserter options, you need to edit the module TopViewInserters_module.py but if you are just changing the parameters, then you can create a job options like the following:

```python
LocalOverride.py

# this part until "end" is the setting for distributed analysis submission
# set the dataset to run on
sub_InDataSets=[
    "csc11.005200.T1_McAtNlo_Jimmy.recotrig.AOD.v11000505",
]
# set the main job option
sub_jobOp="TopView/Analysis_topOptions.py"
# put your user name
sub_username="akirashibata"

#End

# import TopView
from TopView import *

# no screen dump there are also other options defined in Analysis_topOptions.py
doScreenDump=False
# only run Atlfast and truth, (put "FullReco" to run full reco job)
Analysis=["Atlfast", "Truth"]

FastSimInserter = TopViewInserters("Inserter", mode="FastSim")
FastSimInserter.ElectronInserter.override({ "etCut" : 30 })
FastSimInserter.ParticleJetInserter.override({ "etCut" : 40 })
```

25
2.4 PRODUCING TOPVIEWNTUPLE

2.4.4 Changing the Structure of Trees

The trees are written out using EventViewNtupleDumpers. This is done in two stages: first information is written out to UserData (variable container in EventView) using EventViewUserDataCalculators tools. This mainly involves reading information from AOD objects or calculating new ones from objects in EventView (including other variables in UserData). Once variables are in UserData, it is then simple to write it out into Ntuple/trees. Typically this involves scheduling of one EVModule (from Analysis_topOptions.py):

```
FullRecoLooperAna += AANtupleFromUserData("RecoAADumper", filename=NTupleName, sequencer=theJob, 
                           EventTree=True, CandTree=False, Prefix ="FullRecoAna", SaveParent=False)
```

Here, FullRecoLooperAna is the EV tool looper used to build the EventView. The options in this module are important if you want to reorganize the tree structure (also see EventViewNtupleDumpers):

- EventTree: enable event based tree.
- CandTree: enable candidate based tree.
- Prefix: prefix for the naming of the tree. Here, tree names will be FullRecoAna0, FullRecoAna1 and so on.
- SaveParent: specify to follow parent EV to obtain UserData variables that were not calculated for the given view.

The last option needs more explanation. Analysis in EventView may produce "child" EventViews from the first EV that you started with. This happens when eg. combinatorics tools. This is the mechanism to create different "views" of events depending on eg. different combinations. For eg, you can start with a view with selected final state objects and using combinatorics tool you will produce multiple views with different set of jets selected for W reconstruction. UserData variable calculated on the parent EventView is available through child one by following the chain (variable calculated for child only is not accessible from the parent). Here SaveParent option is used to put variable calculated in child view into a separate tree.

In TopView ntuple, basic variables such as information from AOD objects (electron kinematics, jet calorimeter info, etc) are saved in FullReco0 tree. Infromation from the result of further analysis (reconstructed top mass for eg) is saved in separate tree, FullRecoAna0. This is done by first using AANtupleFromUserData on the parent EV and then using again the same tool with SaveParent False. Therefore if you want to put all information from full reco analysis, then you can get rid of the first one of these and use only the second with SaveParent False.
Chapter 3

Appendix B: The TopView Manual

3.1 Getting Started

3.1.1 How to Get Started - 12.0.x

The latest tag is TopView-00-12-12-02 ($version = 00-12-12-02 in the following) This version is compatible with 12.0.6 (groupArea fixes some problems)

Just follow EventViewGroupArea and check out TopView in your workarea (not necessary if packman package was used):

```
cmt co -r TopView-$version PhysicsAnalysis/TopPhys/TopView
```

and set up top view:

```

cd PhysicsAnalysis/TopPhys/TopView/TopView-$version/cmt
#
# (TopView-$version directory may not exist depending on your setup)
cmt config
source setup.sh
gmake
```

To test, do athena TopView/Analysis.topOptions.py from PhysicsAnalysis/TopPhys/TopView/TopView-$version/run (you must provide alternative data if you are not on lxplus.) If it runs, you are all set.

**NOTE**: TopView is a runtime package, you can run your jobs from TopView/run directory. You don’t need to go to UserAnalysis.

3.1.2 How to Get Started - 11.0.x

The latest tag is TopView-00-11-14 ($version = 00-11-14 in the following)

Just follow InstructionsForEventViewin1105 and check out TopView in your workarea (not necessary if packman package was used):

```
cmt co -r TopView-$version PhysicsAnalysis/TopPhys/TopView
```

and set up top view:

```

cd PhysicsAnalysis/TopPhys/TopView/TopView-$version/cmt
#
# (TopView-$version directory may not exist depending on your setup)
cmt config
```
3.2 Inside TopView

3.2.1 Package Contents

You can see the code in CVS repository.

EventView aims to increase the amount of python used in analysis, making the analysis more flexible and portable. In writing analysis, the user should first consider using the tools available under the EventView-Builder package before writing his/her own tools in C++. The contents of TopView package is the following:

- **TopView/run**: The directory where you can run your job from. Also contains scripts to submit PANDA jobs.
- **TopView/python**: Consists of python modules each of which is a sequence of tools scheduled to the tool looper.
- **TopView/share**: Where jobOption files are stored. The top option can be found here.
- **TopView/src**: This is the place to implement C++ tool specific to top analysis.
- **TopView/TopView**: This is the place to place header files for the C++ tools.
- **TopView/cmt**: Manages dependencies, setup and compilation of the package.
- **TopView/root**: Contains a small root package to analyze the ntuple produced by TopView.
- **TopView/pyroot**: Contains a package written in pyroot to analyze the ntuple produced by TopView. Recommended.

Following sections explain each components in more detail.

3.2.2 Python Modules (python directory)

Modules can be found in the file `ModuleName_module.py`. Some of the modules are merged, for eg, `LeptonicW` and `HadronicW` reconstruction module can both be found in the file `WReconstruction_module.py`

- **TopViewModule**: The base class from which all TopView modules are derived from.

To test, do `athena TopView/Analysis_topOptions.py` from `PhysicsAnalysis/TopPhys/TopView/TopView-$version/run` (you must provide alternative data if you are not on lxplus.) If it runs, you are all set.

**NOTE**: You may need to change the input file to something other than the default. Please edit the line:

```
EventSelector::InputCollections = \
["/afs/cern.ch/user/a/ashibata/scratch0/data/csc11.005200.T1_McAtNlo_Jimmy_digit.RDO.v11004205..00005.pool.root.14.AOD.pool.root"]
```

to the file you have. The directory `/afs/cern.ch/user/a/ashibata/scratch0/data/` normally have some files to run some tests on.

**NOTE**: TopView is a runtime package, you can run your jobs from `TopView/run` directory. You don’t need to go to UserAnalysis.
Inserter Modules
- **TopViewInserters**: Inserts full simulation and atlfast AOD objects into EventView.
- **TopViewTruthInserters**: Inserts truth objects into EventView.

Calculator Modules
- **TruthUserdata**: Labels truth particles with "HadronicW" etc.
- **EventVarCalculator**: Calculates event variables such as HT, MTotal, sphericity, aplanarity.
- **WTopCalculator**: Calculates the kinematic information of reconstructed W and top, also does truth matching.
- **RecoUserDataDump**: Calculates kinematic and other type specific information of objects inserted into EventView.
- **ParamBTagger**: Calculates kinematic btag probability for given jets and combined event probability for specified working point.
- **EVSecVtxUserData**: Obtains secondary vertex for the given jet.

Match Module
- **MatchTruthToReco**: Matches truth objects to reconstructed objects.
- **MatchRecoToTruth**: Matches reconstructed objects to truth objects.

Reconstruction Module
- **LeptonicWReconstruction**: Leptonic W reconstruction using W mass constraint.
- **HadronicWReconstruction**: Hadronic W reconstruction using combiner tools.
- **TopReconstruction**: Top reconstruction using b-tagging.
- **TopReconstructionNoB**: Top reconstruction without b-tagging.
- **TopReconstructionFit**: Top reconstruction using b-tagging, creates all possible combinations.
- **ForwardJetFinder**: Finds forward jets and labels them.
- **TTBarReconstruction**: Lepton + jets ttbar reconstruction.
- **TTBarReconstructionNoB**: Lepton + jets ttbar reconstruction with no B tagging information.
- **WgluonSingleTopReconstruction**: Reconstruction of Wgluon single top.
- **HitFitTTBar**: Kinematic fit for ttbar reconstruction.

Calibration Module
- **MergeMuonAndJetTag**: Merges a jet with muon (for semileptonic decay of B meson).
- **SubtractElecFromJet**: Subtract electron in the cone of a jet. (see ATL-PHYS-PUB-2006-022)

Histogramming Module
- **TruthHistograms**: Create truth validation histograms.
- **RecoHistograms**: Create reconstructed object validation histograms.

Other Modules
- **TopViewSelection**: Applies event selection for ttbar ans wg single top analysis.

3.2.3 **JobOptions** *(share directory)*
- **Analysis_topOptions.py**: The top default job option. See CustomizeTopView.
- **TopDefault_jobOptions.py**: Default job setting used for ntuple production.
- **TopDefault11_jobOptions.py**: Default job setting used for ntuple production with a converter to read data produced with 11. (obsolete)
- **FastSim_jobOptions.py**: Job setting used for atlfast samples.
3.2. INSIDE TOPVIEW

3.2.4 C++ Tools (src directory)

Tools based on EventViewBaseTool

- **AddAllObjects**: calculate the vector sum of all particles in an event.
- **EventTypeCalcTtbar**: finds the type of ttbar truth events from truth objects (lepton + jets, fully hadronic etc)

Tools Based on EVUDObjCalcBase

- **EtSumCalc**: calculate the Et sum (HT) of the particles specified by Labels property.
- **ForwardObjLabeller**: labels forward objects specified with MinEta and MinPt.
- **TruthLabeller**: labels truth objects including hadronic W, leptonic top etc
- **MotherInfo**: finds the mother W and top from a given reconstructed object
- **ObjectLabeller**: puts labels to reconstructed objects according to their kinematics ("central" etc)
- **CosThetaStarCalc**: calculates cos(\theta\star) for a given object
- **WTransMassCalculator**: calculates W transverse mass for a given lepton and missing et
- **JetMCCalibration**: calculates calibration constants for light jets based on truth information

Tools Based on EVTransToolBase

- **JetTagMuonMerger**: merges (vector sum) a given jet with the lepton identified for semileptonic decay of B meson.
- **JetElecSubtract**: subtracts four vector of electron in the cone of a jet
- **TruthTransformer**: remove invalid truth and obtain relevant particles.

Non EventViewcode

- **CalibMapJetRaw**: helper tool to calculate calibration constant (written by Eric COGNERAS). Used in JetMCCalibration.

Tools Based on EVHistoBase

- **EventInfoHistogram**: Creates event information histograms.
- **KinematicsHistogram**: Creates histogram for kinematics of given objects.
- **TruRecoMatchHistogram**: Creates histogram from truth match information.

3.2.5 Root Packages (root and pyroot directory)

See TopViewRootPackages section.
3.2.6 Scripts (run directory)
- GetNtuples.py: Used to download ntuples from DQ2.
- GridSetupLxplus.(c)sh: Sets up environment for grid tools on lxplus machines (or possibly other machines with afs access).
- TopViewSubmission.py: Panda submission script.

3.3 Physics Validation Using TopView

Validation of TopView is ongoing and useful reference histograms are being defined.

3.3.1 Validation Histograms

To quickly check the performance of TopView, some validation plots are included by default. These can be found under:

RecoHistogram

for reconstructed objects and reco-to-truth match. The prefix corresponds to that of ntuple variables.

TruthHistogram

for truth objects and truth-to-reco match.

Usage Examples
- Some plots on 5200 sample can be found on SingleTop5200Validation where the histograms are used for validation purposes in single-top CSC note.
- Another example of the use of validation plot is shown here where the results of release 11 and 12 is compared.

3.3.2 Event Info Histograms

Event Type plots (for ttbar only)

Event type for ttbar events are identified. It is also usable to single top (look at semi leptonic only).
3.4. **TOPVIEWANALYSIS**  

The TopView package aims to cover a large scope of physics analysis process from preselection to top reconstruction. The following describes the procedure of this analysis. The analysis can be divided into preparation and further analysis stage. In the preparation stage, AOD objects are selected, calibrated and

1. Full Hadronic
2. Dileptonic
3. Dileptonic Elec Elec
4. Dileptonic Muon Muon
5. Dileptonic Tau Tau
6. Dileptonic Elec Muon
7. Dileptonic Muon Tau
8. Semi Leptonic
9. Semi Leptonic Elec
10. Semi Leptonic Muon
11. Semi Leptonic Tau
12. OTHER (truth problem)

**Other plots**
- **MissingEt**: using the default one which is MET_Final (in reco) and MET_Truth (in truth).
- **MissingEt_Phi**: phi of the missing et of the event.

### 3.3.3 Particle plots

**Kinematic plots**

Defines $E$ (energy), $N$ (number), charge, eta, phi, mass and $p_T$.

**Truth Match plots**

These are only available for truth histograms.
- **Reco_E_scale**: Energy scale (reco/tru)
- **Reco_p_T_scale**: Pt scale (reco/tru)
- **Reco_eta_diff**: Eta difference (reco-tru)
- **Reco_phi_diff**: Phi difference (reco-tru)
- **Reco_p_T_resolution**: Pt resolution ((reco-tru)/tru)
- **Reco_eta_(not)matched**: This is for efficiency/fake against eta. eg El_Reco_eta_matched/El_eta will plot electron reconstruction efficiency against eta.
- **Reco_p_T_(not)matched**: Same as above for pt.

Note that efficiency plots cannot be done when ntuples are created as their entry stop to make sense once histograms are added from separate ntuple files.

### 3.3.4 Top Monte Carlo Validation

Joe Foster has a [web page](http://example.com) which provides links to sets of validation plots for different Top Physics Monte Carlo generators, event selections, and Athena Releases. The intention is to spot differences and inconsistencies quickly and easily.

### 3.4 TopViewAnalysis

The TopView package aims to cover a large scope of physics analysis process from preselection to top reconstruction. The following describes the procedure of this analysis. The analysis can be divided into preparation and further analysis stage. In the preparation stage, AOD objects are selected, calibrated and
associated. Some top physics variables are also calculated here. In the second stage, combinatorics and object reconstruction are done.

### 3.4.1 Preparation Stage

The preparation stage is where objects are for the analysis and these are the objects found in the ntuple. This process is particularly important for the common definition of object in the collaborative environment.

**Preselection and Overlap Removal**

Preselection and overlap removal is the first step in the analysis which defines the physics objects from reconstructed AOD candidates. Preselection and insertion is done using EventViewInserters and the default preselection/overlap removal is defined in TopViewInserters for fully reconstructed and fast simulation objects and TopViewTruthInserters for truth objects.

**Fully Reconstructed and Fast Simulation Objects**

The specified preselection is applied and overlap is removed between the objects from different containers. Insertion order is also relevant as objects inserted first have higher precedence of survival. By default in TopView, objects are inserted in the order: Muon, Electron, Photon, TauJet, ParticleJet. BJets are not selected separately but indicated by adding a label BTagged to ParticleJet objects.

If it become necessary to define separate definitions between sub-groups, they can provide their own definition by developing their own inserter modules.

**Truth Objects**

Selection of truth objects is somewhat more complex. Saving everything will result in a large ntuple with much information not very useful for any purposes but important objects should not missed out. The truth selection is done in the following order:

1. Vertex filtering: here, objects are selected by their decay patterns. In TopView, there are top decay, W decay (both hadronic and leptonic) and Z decay. This will enable us to keep the objects of interest including the ones that are not stable (eg quarks.)
2. Adjustment to vtx filtered objects: Vtx filtering sometimes results in an undesirable consequences. For eg, immediate daughter W from top decay in MC event record may not be the right choice: generators make modifications to these objects by further letting them to decay (not in a physical sense but for internal adjustments). These self-decay should be ignored and the last object in the self-decay chain should be obtained. Also, leptons may not be in the stable state (before radiation) if the immediate daughter is selected and the stable lepton should be sought to find more physical objects. These are obtained by replacing the objects selected by the vtx filtering.
3. Stable leptons and photons: Stable leptons and photons other than the ones found by vtx filter may be a result of brem etc. These objects are selected by using EventView truth inserters by their pdgId and stableness.
4. Visible tau decay: Tau lepton decays both hadronically and leptonically and either may involve weak interaction where some energy is lost by neutrinos. Reconstruction only finds the "visible" part of the decay products and therefore, their truth equivalent has to be included in the truth tree. Visible tau decay products are summed up into a composite particle and selected by using inserters.
5. Truth jets: There may be extra jets from ISR/FSR and the truth equivalent of these is hard to find by looking at individual particles. Therefore, truth jets, the objects which results from applying
jet algorithm to stable truth particles, which do not overlap with the objects previously inserted, are inserted here using inserters. By this method, truth jets which are the results of vertex filtered objects (like u quark from W decay) are most likely to be rejected by overlap removal and therefore they are instead associated to these quarks later in the analysis.

**UserData Calculation**

Once objects are inserted into EventView, it is now possible to calculate useful quantities out of these objects. For this, EVUD calculators and some custom tools are used. These are done in `RecoUserDataDump` module for reconstructed objects and `TruthToRecoAssociators` for truth objects. Some of these are done in association module whose primary purpose is truth matching. This is to reduce the number of looping-over-objects operation since extra object looping introduces significant inefficiency.

**Truth Match**

Both "Truth to Reco" (by `MatchRecoToTruth` module) and "Reco to Truth" (by `TruthToRecoAssociators` for `TruthJet`, `TruthJetPa`, `TruthTau` objects and by `MatchTruthToReco` module for vtx filtered objects only) match are done. Truth match is based on deltaR method and the matching parameter is defined in `MatchParameters`.

**3.4.2 Analysis Stage**

The second stage is further analysis. Here, hadronic and leptonic tops are reconstructed. It’s not meant to be THE analysis but is more for reference/validation purposes right now. By default, ttbar analysis is run which assumes lepton + jet reconstruction.

**ttbar Analysis**

Here’s a brief explanation to the ttbar analysis found in TopView with the names of corresponding modules.

1. **(HadronicWReconstruction)** take two non-tagged jets and combine them to form a (or many) "hadronic" W.

2. **(TopReconstruction)** combine the above W with nearest b-jet to form a (or many) "hadronic" Top.

3. **(LeptonicWReconstruction)** reconstruct "leptonic" W using missing Et and W mass constraint, ie, all W reconstructed from this branch has exact W mass.

4. **(TopReconstruction)** do the same for leptonic side.

5. Select one combination by choosing the combination with the highest pt hadronic top.

EventView analysis might be unfamiliar at the beginning since it is quite different from the AOD analysis done in AnalysisExample etc. The following briefly illustrates the flow of a part of the ttbar analysis implemented in TopView (the configuration method is somewhat old):
3.4.3 Other analysis available in TopView

**ttbar Analysis without B-tagging**

Based on Bentvelsen, S; Cobal, M; 2005:

1. Take 3 jets and combine them (top candidates).
2. Chose the EV with highest pt top.
3. Take 2 out of the top daughters (W candidates).
4. Chose the EV with highest pt W.

**w-g Single Top Analysis**

This is mostly based on the same set of modules used in ttbar analysis by using the leptonic top reconstruction module. One of the goals of this analysis is to boost into the rest frame of the top and measure it's polarization.

3.5 Customizing TopView

3.5.1 Different Ways of Customization

There are several levels in customizing TopView, from which you can choose the most efficient one for your needs:

1. Change the options listed in the top jobOption file, Analysis_tobOptions.py: This will let you change ntuple variables, wheather to include some variables or not, wheather to include analysis or not, wheather to save ntuple or not, change the name of the input/output file etc.
2. Write your own jobOption using existing python modules, or modify the existing jobOptions: This will let you fine tune your ntuple variable selection or write your own analysis based on the standard EventView tools and modules available in TopView.
3. Write your own python modules: This will let you create your own python module that can be used in your jobOption. See below for how to write a new python module.
4 Write your own C++ tools: This will give you total freedom on what you want to do with the least constraints from what we have in the "standard" set of tools. You can write a small tool to fit in the other standard tools or you can write a large tool that processes a major part of your analysis.

3.5.2 Using existing jobOption or writing your jobOption

TopNtupleDIY have some useful information for this. Also, TopViewTutorialUsingTools explains basics of how to write your own job option files based on tools and modules available. The options available for default top option is described below:

- Previous version: TopViewTopOption11

The top jobOption for TopView is Analysis_topOptions and from here, you can switch on/off different options available to customise your job. The following are the available options. You are not recommended to modify the top JO file itself, but specify the options in a separate file and include it with (as explained in TopNtupleDIY)

```
# single top specification
doScreenDump=False
Analysis="["Reco", "Truth"]"
KinematicsOnly=False
Parameters = InsertParametersSingleTop(DoPreselection=Preselect, InsertOnOverlap =not RemoveOverlap)
```

Options Available

- name of option (default): description
- Analysis (["Truth","FullReco", "Atlfast"]) : Which jobs to do, Truth/=FullReco/=Atlfast=
- doAnalysis (True) : Do ttbar or single top analysis?
- TtbarAnalysis (["LepJets", "BTag"]) : Include ttbar analysis? LepJets for lepton + jets, BTag for use of B tagging NoBTag otherwise.
- SingleTopAnalysis ([]) : Do single top reconstruction? Leptonic and NoBTag implemented
- KinematicsOnly (False) : Only save kinematic information, no type specific info. False for light weight and faster running.
- NtupleName ("TopViewAANtuple.root") : Name of the output file.
- doScreenDump (True) : Do screen dump. Useful for testing but slow. Turn off for batch jobs. More on EventViewScreenDump.
- showUserData (False) : When doing screen dump, print value of UserData as well (can create a huge std output)
- ParamBtag (False) : Do TRF b-tagging?
- SemilepTagging (False) : Include semileptonic tagging? (temporary switch for dependencies)
- doHisto (True) : Produce validation histogram?
- doMatch (True) : Do truth Match?
- doJiveXML (False) : Dump JiveXML files?
- doTrigger (False) : Include trigger info? (temporary due to dependencies)
- readEvgen (False) : Read EVNT file rather than AOD?
- exportContainer (False) : If true, only do preselection
- removeOverlap (True) : Switch for doing overlap removal.
3.5. CUSTOMIZING TOPVIEW

- `doPrintMC (False)`: Run PrintMC algorithm instead of TopView job. Useful for debugging.
- `doDumpMC (False)`: Run DumpMC algorithm.
- `doSGDump (False)`: Dump SG contents. For debugging.
- `OutputLevel (3)`: Overall output level.

Rest of the top option

TopViewAnalysis outlines the default TopView job.

3.5.3 Writing TopView python modules

TopView uses python modules to group individual EventView tools into meaningful set of tools. Taking EventView modules as basic building block, you can construct your analysis by creating bigger modules that deal with problems such as reconstruct W, or match truth. You can use the script `newTopViewModule` (available once you’ve done `make` for TopView) to create a new module:

```
$ newTopViewModule nameOfModule
```

nameOfModule.py is created under python directory. You can now start filling in the skeleton. The methods you must fill in are:

```python
def schedule(self):
    """typically you make a list of modules to be scheduled
    and use addToolsToSelf to schedule them. The tools
    will become the attribute of the class""

    self = self.self()

    self += [
        anEVTool("EV_tool_name/instance_name"),
    ]

    self.EV_tool_name += anEVTool("EV_tool_name2/instance_name")

def setDefaults(self):
    """set parameters of the tools by passing a dictionary to set properties ""

    self.EV_tool_name.setProperties(  
        parameter1 = value1,  
        parameter2 = value2,  
    )  
    # or equivalently,
    self.EV_tool_name.setProperty(  
        {  
            "parameter1" : value1,  
            "parameter2" : value2,  
        }  
    )
```

and you can specify the set of tools you will schedule in this module and corresponding configuration of the tools. The top reconstruction module is explained in detail in the tutorial, TopViewTutorialFurtherAnalysis.
3.5.4 Writing your own C++ tools

Writing of C++ tools is also done by generating skeleton code and filling in the appropriate methods. As you can see in TopViewContents, there are several types of tools used for different purposes and once you know which kind of tool you need, choose one of the following scripts to generate your skeleton:

- `newEVTool`: for tools based on EventViewToolBase. Any tool can be based on this.
- `newEVUDAssocTool`: for tools based on EVUDobjAssocBase. For associating one object to another.
- `newEVUDobjTool`: for tools based on EVUDobjCalcBase. In the case of object calculator.
- `newEVCompTool`: for tools based on EVComparatorBase. If you want to write EventView comparator which orders multiple EventViews.

A couple of examples can be found in TopViewTutorialWritingTools (an event variable calculator and an object variable calculator) and more details can be found in the EventView wiki page. Once tools are written, you can schedule them to EVToolLooper or put them into TopViewModule and then use it as a part of your analysis.

3.6 Root Packages

TopView provides two packages to analyze the ntuple produced. One is Root based and the other is based on PyRoot. Both of them provides basic framework for event looping and supports reading multiple trees.

Disclaimer

The following packages are provided as an example to show how to start root analysis with TopView ntuple. They are general packages and can manage complex analyses but they are not meant to become a common framework for the top group. Support for these packages will be minimal and no further developments are planned. If you need help, please try to find a local expert on root: EventView hypernews is not an appropriate forum.

3.6.1 Root Package (found under root directory)

The Root package is based on MakeClass mechanism provided by Root. It has another layer of organization since MakeClass does not support reading more than one tree at once as it is.

Package Contents

- `cpplib`: Contains the base classes and MakeClass’d classes. MakeClass’d classes should be replaced once you change the format of your ntuple. Also includes some analysis tools.
- `CppAnaMain.C`: The main steering file, defines the files to read in and which analysis to do.
- `TopViewAnalysis.C (.h)`: Example analysis demonstrating the use of the base class.
- `Script.C`: Small example demonstrating the use of CINT scripting.

How to Write an Analysis

The example analysis found here (TopViewAnalysis) is derived from the class Analysis. The idea here is that THE Analysis base class will provide simple event loop as well as Initialize/Finalize and control the flow of the analysis. The package make use of automatically generated classes (MakeClass). The user only have to overload these methods. The following methods are called in the given order:

1. `Initialize`: Initialize before the event loop
2. `BookHistogram`: Book your histograms
3.6. ROOT PACKAGES

CHAPTER 3. APPENDIX B: THE TOPVIEW MANUAL

3.6.2 PyRoot Package (found under pyroot directory)

PyRoot based package is much more robust than the other one mentioned above. Python interface provides a dynamic environment much suitable for physics analysis. This page has a very good introduction to PyRoot and if you are unfamiliar with it, this is the place to start.

Package Contents

- **pylib**: Similar to cpplib above. Consists of base classes and some tools. These files are less often or hardly modified.
- **PyAnaMain.py**: Similar to CppAnaMain.C in the above package. Does the steering of the analysis and schedules input files.
• **TestAnalysis.py**: Similar to TopViewAnalysis.C above. An example analysis based on Analysis base class.

The whole package is written in python (though it is easy to use the modules written in C++ or even a single function written in C++) and that means one can benefit from the range of modules available in python. For eg, a simple example of specifying the input files, in C++ version, you had to list each files up but in PyRoot you can use the `glob` module to enable wild card for input file selection.

**How to Write an Analysis**

Writing of analysis is much like in the Root package explained above though there are small differences. Analyses are derived from `Analysis` class as defined in `Analysis.py` and the base class (or analysis scheduler called `Monster`) is responsible for calling them in the appropriate time of event loop.

1. `BookHistogram`: Book your histograms
2. `Initialize`: Initialize before the event loop
3. `< begin event loop >`
4. `InitializeEvent`: Initialize before execute
5. `PassCut`: apply event selection. Failed events will not be processed futher.
6. `ExecuteEvent`: Execute your analysis
7. `FinalizeEvent`: Finalize your analysis
8. `< / end of event loop >`
9. `Finalize`: Finalize after the event loop

Loading of objects is done automatically and you don’t need to generate `MakeClass` everytime the ntuple structure changes. However, if you decide to load objects into four vector as well (like in the example in `TopViewTutorial`), then you must specify the prefix of the objects you are interested in in the constructor of your analysis:

```python
def __init__(self, AnaName='WJetsAnalysis', RecoTree=NULL, TruthTree=NULL):
    Analysis.__init__(self, AnaName, RecoTree, TruthTree)
    self.RecoObjects=['El', 'PJet']
    self.TruthObjects=[]
    self.weightName='"' # eg "eventWeight"
```

Here you only specified to instantiate four vectors for electrons and particle jets. Of course, all the flat variables are readable no matter what is specified here. (ie `self.RecoTree.Mu_p_T` is still available but you cannot do `self.RecoTree.Mu.Pt()`)

More documentation is written inside the example analysis.

**How to Run**

To specify input files, modify the following:

```python
FileVersion5200=glob.glob('TopView1114_AnalysisMatch/005200.
    T1_McAtNlo_Jimmyv11000505/*root')
CSCSamples={
    "5200":FileList5200
}
```

Here, I’m reading in all .root files under this directory.
In case you are dealing with trees with different names, you may also need to modify:

```python
RecoName="FullSim0"
    print "define chains. Reco chain is", RecoName
    RecoChain = TChain(RecoName)
    TruthChain = TChain("Truth0")
    InfoChain = TChain("CollectionTree")
```

Now the scheduling of your analysis is done by analysis looper (like EventViewToolLooper. You can modularize your analysis this way). You can add your analysis module like the following:

```python
from TestAnalysis import TestAnalysis
    modules=[TestAnalysis("TestAnalysis")]
```

ie, just add them into `modules` variable.

To run, do

```
$ python PyAnaMain.py
# you can run on 20 events only by saying "python PyAnaMain.py 20". Useful for testing.
```