Flat NTuple Analysis

Flat NTuple Analysis

This software tool can be used to analyze flat root ntuples. The aim is to have a lightweight framework to enable quick study of different ntuples.

Description

The tool is based on so-called *Topologies*. The different topologies can reflect different analyses which are interested in different final states e.g. search for bjet+MET and long-lived particles. The idea is that while the final states are different many needs are common for the different analyses; cross-section/k-factor handling, histogramming, cut flow, statistics, etc. Also, the overhead in the use of different ntuples is minimized as these common functionalities are decoupled from the ntuple structure.

It is based on a analysis package developed by J. Sjolin (Stockholm University) and P. Hansson.

A slightly more detailed description is given below.

Short technical description

The core of the framework is in the `/core` directory.

The main idea is to have a common description of an event even though different ntuples may be used. This class is called **AtlEvent** and can be found in the `core/` directory and contains electrons, jets, muons, etc.

The `AtlEvent` is built using the `BuildEvent()` function which uses the interface function `copy_to_interface()` to get information from the ntuple. The `copy_to_interface()` is a derived function implemented in each new ntuple description class called `FileEvent` in order to handle the difference between ntuple variable names. Since all `FileEvent` derives from a base class (`EventSrc`) all subsequent steps are the same independent of the ntuple used. This is how the ntuple content is decoupled from your analysis and when you want to run your analysis on a new ntuple you only need to make sure that the `copy_to_interface()` is working properly to build your event.

The analysis is done using "Topology" classes which all derive from a base class `Topology_base` which contain many useful functions to grab content of your AtlEvent.

The loop over your events is done using `Topology_event` which derives from `Topology_base` but in addition contain the functions to looper over your events `loop_src()` and to add the input root files `add_mc_sample()` and `add_src_list`.

Since this is supposed to be a lightweight framework with focus on quick compilation and running the compilation is done automatically when you run your topology. This is handled by the "steering" macro `ana.C` which is in your topology directory (see example in `topologies/example`. The various classes are compiled, loaded, analysis classes are added and finally the event loop is executed.

Checkout

The code is in SVN and can be found by doing:

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% svn co svn+ssh://phansson@svn.cern.ch/reps/atlasgrp/Institutes/SLAC
/FlatAna
```

In order to run it you need a few packages that is used:

ROOT

Root is needed.

GRL

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% svn co svn+ssh://phansson@svn.cern.ch/reps/atlasoff/DataQuality
/GoodRunsLists/tags/GoodRunsLists-00-00-84 GoodRunsLists
```

Compile this package:

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% cd DataQuality/GoodRunList/cmt
phansson@pcphuat27]/u1/phansson/jetmetbtag/DataQuality/GoodRunList/cmt% gmake -f Makefile.Standalone
```

Copy the shared library to the topology directory where you will run your jobs (thi can be improve at some point):

```
phansson@pcphuat27]/u1/phansson/jetmetbtag/DataQuality/GoodRunList% cp ../Standalone/libGoodRunsList.so ../..
/FlatAna/trunk/topologies/jetmetbtag/
```

JES uncertainty provider package:

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% svn co svn+ssh://phansson@svn.cern.ch/reps/atlasoff/Reconstruction
/Jet/JetUncertainties/tags/JetUncertainties-00-01-02 JetUncertainties
```

This package will be compiled on the fly by FlatAna but you need to copy the root file to the topology directory:

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% cp JetUncertainty/JESUncertaintyProvider/share/JESUncertainty.root
FlatAna/trunk/topologies/jetmetbtag/
```

Trigger Navigation Tools:

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% svn co svn+ssh://phansson@svn.cern.ch/reps/atlasoff/Trigger
/TrigAnalysis/TriggerMenuNtuple/tags/TriggerMenuNtuple-00-01-32 TriggerMenuNtuple
```

This package MUST be compiled first!

```
phansson@pcphuat27]/u1/phansson/jetmetbtag% cd TriggerMenuNtuple/cmd
```

Edit the Makefile in this directory. You must remove the two instances of "-m32" lines 15,20.

```
phansson@pcphuat27]/u1/phansson/jetmetbtag/TriggerMenuNtuple/cmd% make
```

you might need to check that the path to this package are correct in ana.C (your run directory e.g. topology/jetmetbtag/), in particular check a few hard-coded paths (not very nice):

```
gSystem->AddIncludePath(" -I../../../../DataQuality/GoodRunsLists/");

gSystem->AddIncludePath(" -I../../../../JetUncertainties/");

gSystem->CompileMacro("../../../../JetUncertainties/src/JESUncertaintyProvider.cxx","k");

gSystem->AddIncludePath(" -I../../../../TriggerMenuNtuple");

gSystem->Load("../../../../TriggerMenuNtuple/lib/libTriggerMenuNtuple.so","k");
```

Run an example

In order to run on the example you need a ntuple. This example runs on an ntuple:
`user.bcbutler.susySlim11.v1.simple_Bb_B200_L100_herwigpp_susy.merge.NTUP_SUSY.e636_s933_s946_r1831_r1700_p364/`

Make sure you manipulate the filelist:

1. Open `flatana/filelist/filelist.txt` and change the path (top line) and directory/path name to match your own paths.
2. e.g.: change top line-> "/u1/data/"
3. find the line: "BbB200L100.susy011 1.0 13.0 usr/p/phansson/data/user.bcbutler.susySlim11.v1.simple_Bb_B200_L100_herwigpp_susy.merge.NTUP_SUSY.e636_s933_s946_r1831_r1700_p364/"

4. And change it to: "BbB200L100.susy011 1.0 13.0 user.bcbutler.susySlim11.v1.simple_Bb_B200_L100_herwigpp_susy.merge.NTUP_SUSY.e636_s933_s946_r1831_r17\
   BbB200L100.susy011 1.0 13.0 user.bcbutler.susySlim11.v1.simple_Bb_B200_L100_herwigpp_susy.merge.NTUP_SUSY.e636_s933_s946_r1831_r1700_p364/"

All the needed compilations are done automatically in the main root macro `ana.C`. The analysis class `Topology_cutflow` can be run with:

```
[phansson@pcphuat27]/u1/phansson/jetmetbtag% cd FlatAna/trunk/topologies/jetmetbtag

[phansson@pcphuat27]/u1/phansson/temp/flatana_test/FlatAna/trunk/topologies/jetmetbtag% root -l

root [0] .L ana.C

root [1] ana("cutflow","BbB200L100.susy011",100)
```

This should produce root files containing some basic histograms and a log file with event statistics. Have a look at the next section to get more info on what was going on and how to use a generic ntuple.

Run example over your own NTuple

In order to run over your own ntuple you need only to create a new `FileEvent`:

```
[]/u1/phansson/jetmetbtag/flatana/trunk/topologies/example% cp FileEvent_template.h FileEvent_trg.h
[]/u1/phansson/jetmetbtag/flatana/trunk/topologies/example% cp FileEvent_template.cxx FileEvent_trg.cxx
```

Then open your root file (e.g. ntuple.root) and do a makeclass:

```
[]/u1/phansson/jetmetbtag/flatana/trunk/topologies/example% root -l /u1/data/user09.DavidWilkinsMiller.trigger.v02.SethCaughron.misal1_mc12.EnhancedBias_1031.digit.RDO.v13004004_AANT_PHCOPY_SLIM/user09.DavidWilkinsMiller.trigger.v02.SethCaughron.misal1_mc12.EnhancedBias_1031.digit.RDO.v13004004.AANT._00001_slim.root
root 1 CollectionTree->MakeClass("slimclass")
```

Open `slimclass.h` and

1. Copy the declarations of variables and branches to `FileEvent_trg.h` (as private members).
2. Copy **only** the initialization of the variables and SetBranchAddress from the `Init(TTree *tree)` function to the function `SetBranches()` in `FileEvent_trg.cxx` (*Note again that only the initialization of variables and SetBranchAddress() should be copied*)
3. Open `FileEvent_trg.cxx` and `FileEvent_trg.h` and search and replace *template* with *trg*

The main macro to run is the `ana.C` which takes care of compiling the needed dependencies, loading the chosen configurations, adding the analyses /topologies and finally executing the event loop. To make sure that your new ntuple will be used you need to open `ana.C` and edit it:

1. Find the line `/***** ADD YOUR OWN NTUPLE TYPE BELOW` and make sure that your new FileEvent class is compiled.
2. Find the line `/*** ADD YOUR OWN FILEEVENT NTUPLE FORMAT BELOW ****/` and make sure that your FileEvent class is loaded (the string in the file list name given as argument is used to determine what file type should be used)

Now you are ready to run. Use the simple helper script `ana.sh` which is good to schedule several jobs at the same time as in the example above or run directly in ROOT:

```
[]/u1/phansson/jetmetbtag/flatana/trunk/topologies/example% root -l
root 0 .L ana.C
root 1 ana(14,"tt.trg",10)
```

This should produce some histograms and log files.

Adding you own analysis class

In the `topologies/example` folder there is an example class to start from `Topology_l2rate`. It is recommended to make a new directory e.g. `topologies/yourdirectory` in order to submit everything seamless to svn if needed. You can copy the `example` directory.

Edit the new topology .eg. `Topology_test` by adding new histograms, cuts, etc, see below for more details. The only change you need to do in order to run this class is to add this topology to the main macro `ana.C`:

1. Find the line `/***** ADD YOUR OWN TOPOLOGY TYPE BELOW` and make sure that `Topology_test` class is compiled and added.

The first argument in the `add_topology()` function has to be unique and will be added to all output files.

Configuration

Many things are configurable. The configuration files are rather self-explanatoty and an example can be found in `topologies/example/l2bjetrate.config`. Most of the selections are turned off if set to 0. Some example configurations are:

- Choose luminosity
- Object Dr cut definition for matching using automated functions

- Overlap removal of electron/jet/muons (See `do_overlap()` in AtlEvent.cxx))
- Electron,muon,jet,tau definitions

File list

The file list is where the samples you are using are defined. An example file list is in `flatana/filelists/filelist.txt`.. The nice thing is that scale factors and cross-sections is directly given in a straightforward way. The first line contains the base path to your data directory. The follow lines then defines a set of sub-directories for each sample:

```
name, K-factor, cross-sectionin pb, subpath/to/filedirectory
```

Note that all root files in that directory will be assumed to contain the TTree and will be used.