

CMS analysis in CMSSW_1_6_11

Dr Freya Blekman, LEPP Cornell University, Freya.Blekman@cern.ch

1. Introduction

These tutorials contain most minimal information a physicist that does not do immediate code development will need to do basic physics analysis. The tutorial is set up for CMSSW_1_6_11 but most of the web pages that are referenced are updated when new major releases occur. The CMS release mostly used for analysis is currently the CMSSW_1_6_X series¹, but this will be replaced by the CMSSW_2_0_X series in the very near future².

There are two parts to this set of tutorials. The first relates to the writing of analysis code in the framework, while the second part is more focused towards grid-based analysis. For beginning users there probably is a bit too much material for an entire day, but the collection of links will be useful in further analysis work.

Note: After completing these tutorials, it is advisable to make a tar ball and copy everything to your local user node for future reference.

2. List of nodes

The main machine to run CMSSW jobs on is:

.....

If you are running on a machine with a local linux installation and it has the CERN installation of scientific linux 4 (scl4) you probably also have everything installed, but this is beyond the scope of this tutorial.

¹ The CMSSW_1_6_X series is used because the last grid-based large Monte Carlo production was done during the Computing Software and Analysis challenge in 2007 (CSA07). In May 2008 the CSA08 will start, which will be producing large samples in CMSSW_2_0_X.

² If you are feeling courageous or are an experienced CMSSW user you can consider doing this entire tutorial in CMSSW_2_0_0, this should theoretically be possible.

3. Setting up the environment

On lxplus.cern.ch, CMSSW is already set up at login. To use the CMS grid computing facilities too, the following paths need to be set and necessary scripts sourced³:

```
source /afs/cern.ch/cms/LCG/LCG-2/UI/cms_ui_env.csh
# (for (t)c-shell) OR
source source /afs/cern.ch/cms/LCG/LCG-2/UI/cms_ui_env.sh
#(for bash)
```

The script above also sets up your account for grid use, so if you have a valid grid certificate and are registered with the CMS virtual organization (VO) you can now run grid jobs! You are now ready to run the CMSSW tutorials from the Workbook⁴ or any other source (like these tutorials).

4. Create a work area

No matter what the project is that you plan to work on, you will need a working environment that has all paths set appropriately. Our first step will be to create such a working environment. Documentation on how to do this can be found here⁵, but in brief these are the commands:

```
scramv1 project CMSSW CMSSW_1_6_11
```

this command produces a working environment in a CMSSW_1_6_11 directory.

For most normal applications you only need to work and edit in the CMSSW_1_6_11/src directory. In the workbook, there is a lot more documentation available on setting up the work environment. Now

³ These settings change per site, if there is no installation available at your particular site you might also want to consider installing the CMS software environment via AFS (ask your local administrator how to do this).

⁴ The CMS Workbook,

<https://uimon.cern.ch/twiki/bin/view/CMS/WorkBook>

⁵ <https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookSetComputerNode>

change to the appropriate directory and execute the last step of the framework setup:

```
cd CMSSW_1_6_11/src
cmsenv
project CMSSW
```

The `cmsenv` command sets up all the appropriate paths. Without it you will get unintelligible software errors when compiling and running, so executing this command usually is the first 'fix' to try when you observe errors that you did not expect or recognise. The `project CMSSW` command sets up access to the Concurrent Versions System (CVS), which you will need to access already existing code.

CMSSW uses the `scramv1` program for compilation and the workbook contains a lot of additional information on `scramv1`⁶. One important thing to bear in mind is that there also is a `scram` command, used mainly in grid computing. Mixing the two commands obviously creates software conflicts.

Project 1: Setting up the CMSSW environment

- a) Log in to your CMS machine defined in Sec. 2. Create a software environment of your own.*
- b) What happens when you change your version of `scramv1` to `scram`?*

⁶ <https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookScramV1Intro>

5. Running CMSSW

To first order, the CMSSW framework has only one single executable, called `cmsRun`. Any compiled code (*module*) needed by `cmsRun` is included using configuration steering files. The files have extension `.cfg` (direct steering file, i.e. `cmsRun mysteeringfile.cfg`), `.cff` (modify settings of modules, included in a `.cfg` file) or `.cfi` (instantiate settings of modules, included in a `.cfg` file)⁷. In the main configuration file the different modules are called and their execution order is defined. It is also possible to write aliases of series of modules called *sequences*, and these again can be chained together. The next example shows how to do a very simple framework analysis with `cmsRun`.

Project 2: Basic framework analysis

*a) Work through the introductory framework tutorial here:
<https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookWriteFrameworkModule>
You probably need to make some modifications. For one, the tutorial does not use `CMSSW_1_6_11`, so make sure you keep using the correct version. Also, you will need to find a file to run on. We have arranged for some files to be copied to your local machine and they can be found here:*

.....

⁷ Note that currently, the `.cfg/.cfi/.cff` file nomenclature is currently interchangeable and it is left to the user to do this conform to the CMS coding standards.

6. Creating an analysis in the framework using the Physics Analysis Toolkit (PAT)

In CMSSW, there are many different ways to do data analysis. Some of these are completely root-based and will not be discussed here⁸. Instead we will focus on analysis that is completely done in the framework, which has the advantage that the analysis can also easily be run on the grid. The toolkit we use is called the Physics Analysis Toolkit (PAT) and this analysis framework is intended for end-user analysis of early data.

To produce files that we can easily read in ROOT we will also use the StarterKit, a framework intended to easily book histograms from the PAT output.

Project 3: Compiling the PAT

The edanalyzer tutorial in Project 2 uses only one kind of object, tracks. We will now work through a tutorial which produces framework objects that can be used for analysis. The PAT analysis framework is heavily documented in the CMS WorkBook and Offline Software Guide⁹. Our first step will be to compile and run it. We will use the 'latest and greatest' version of cvs tags for end April/start May 2008. Note that the PAT documentation has a recipe for the impatient user which lists the current best tags, this is useful if you decide to redo/continue this tutorial at a later date.

a) Check out the appropriate packages for the PAT. The list of packages is very long for CMSSW_1_6_11 (but all of these will be in CMSSW_1_6_12 in the near future):

```
# Muon ID
cvs co -r V00-16-01-00 DataFormats/RecoCandidate
cvs co -r CMSSW_1_6_11 RecoMuon/MuonIdentification
cvs co -r V01-02-06-01 RecoMuon/MuonIdentification/src/IdGlobalFunctions.cc
```

⁸ <https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookMakeAnalysis> contains examples of all three methods. The two root-based methods have the disadvantage that in ROOT the debugging of compiler errors is a lot more difficult.

⁹ <https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePATRecipes>

```
cvs co -r V01-02-06-01 RecoMuon/MuonIdentification/interface/IdGlobalFunctions.h
cvs co -r V02-15-10-00 TrackingTools/TrackAssociator
# Jet Corrections
cvs co -r V02-04-04-06 JetMETCorrections/Type1MET
# Bug fix in c-jet flavor identification
cvs co -r B160_V00-04-06 PhysicsTools/JetMCAlgos
# Included to use new NamedCompositeCandidates
cvs co -rNamedCompositeCandidate_1611 DataFormats/Candidate
cvs co -r pat_1611_080425 PhysicsTools/CandUtils/
cvs co -r pat_1611_080425 PhysicsTools/CandAlgos/
# Includes global version of Steven Lowette's CSA07 processing tools
cvs co -rPAT_1611_080425 PhysicsTools/HepMCCandAlgos
# check out the pat source code
cvs co -r V01-02-04 PhysicsTools/PatAlgos
cvs co -r V01-02-03 PhysicsTools/PatUtils
cvs co -r V01-02-03 DataFormats/PatCandidates
# check out the TQAF prescription for demonstrations of complex Layer 2 examples
cvs co -r pat_tk_tutorial_080424 AnalysisDataFormats/TopObjects TopQuarkAnalysis
cvs co -r TQAF_1611_080415 CondFormats/PhysicsToolsObjects
cvs co -r TQAF_1611_080415 PhysicsTools/MVATrainer PhysicsTools/MVAComputer
```

Make sure you are in the correct location (which should be \$CMSSW_BASE/src) when you execute these commands)

The list of packages is long because a lot of the CMSSW_2_0_X series code has been back-ported into CMSSW_1_6_11 for use with physics analyses in this version of the framework. In the (future) release CMSSW_1_6_12 this will be fixed.

b) Compile the PAT by typing

```
scramv1 b
```

Note that the location of the files is important, as are the actual file and directory names. The compilation takes quite a while (this will be fixed in CMSSW_2_0_X and has to do with ROOT shared library technicalities) so open another shell window and start with the next exercise once you have assured yourself that the compilation is running smoothly.

Project 4: Looking at File Content and accessing data files

a) Look at the content of the MC files by dumping the first event to

screen. You have already done this in Project 2, but this time, take some time to study the different reconstructed objects that are available for your analysis.

b) Also try to open the file in ROOT. This can be done by adding the file name to the command line when starting ROOT:

```
> root -l myfile.root
```

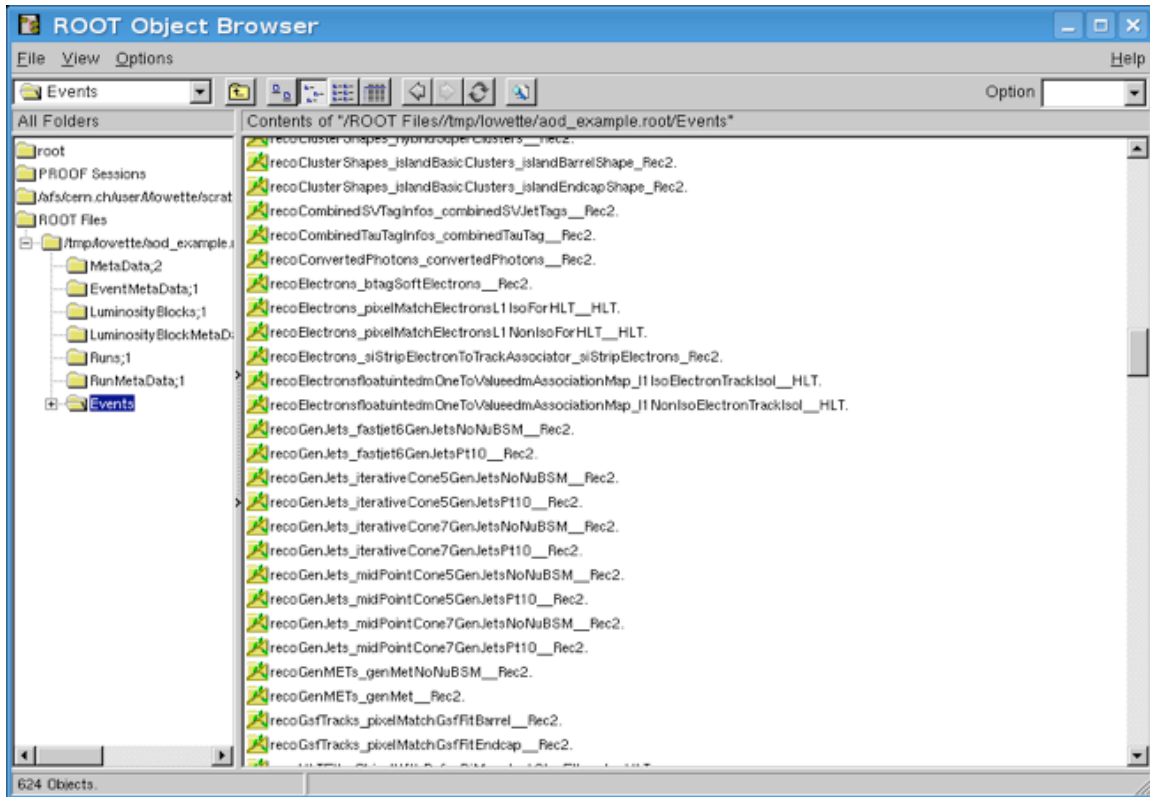
where `myfile.root` should obviously be replaced by the file you want to look at. The `-l` option is optional, it stops ROOT from showing the flash screen with root logo etc on start up. Note that you will get many errors of type

```
Warning in <TClass::TClass>: no dictionary for class  
?????????? is available
```

which is ROOT telling you that it does not understand the objects which are in the CMSSW output. However, you can still look at the file by using the ROOT browser, which starts by typing

```
root [0] TBrowser tb;
```

The window that will open will look something like what is shown in Figure 1, except that there you can already see the content of the file. Open a file by clicking on it.



Project 5: Opening a CMSSW file in fwLite

It is also possible to first load the libraries for all the available objects in your CMSSW file. The set of libraries that is supplied to work in this is called *fwLite* (you can think of it as a light-weight version of the CMSSW framework). For this you do need to have set up a CMSSW environment¹⁰.

If you include the following commands on the root command line before you open a file, the 'warning in <TClass::TClass>: no dictionary' errors in root will disappear:

- a) execute the following commands on the ROOT command line before opening a file:

```
root [0] gSystem->Load("libFWCoreFWLite.so");
root [1] AutoLibraryLoader::enable();
root [2] gSystem->Load("libDataFormatsFWLite.so");
root [3] gROOT->ProcessLine("namespace edm {typedef
```

¹⁰ Having set up a CMSSW environment means you have issued the `cmsenv` command in a `CMSSW_*_*/src` directory, this sets up all appropriate paths including the path to the `fwlite` libraries.

```

edm::Wrapper<vector<float> >
Wrapper<vector<float,allocator<float> > >; }");
root [4] gROOT->ProcessLine("namespace edm {typedef
edm::Wrapper<vector<double> >
Wrapper<vector<double,allocator<double> > >; }");
root [5] TFile *_file0 = TFile::Open("myfile.ro
ot");

```

where commands 3 and 4 are optional, and command 4 actually opens the file. Notice the fact that you no longer get errors and (even better) you now get access to all class access methods (including for example lorentz vector calculus) for the objects in your root files. Many of the examples in the workbook are based on fwLite but for now we will continue using full CMSSW.

Project 6: Running the PAT

By now your compilation of the PAT framework should be done. It is now time to run the PAT and look at the output.

a) All scripts to run the PAT are located in the following directory:

```
cd $CMSSW_BASE/src/PhysicsTools/PatAlgos/test
```

You might remember that the PAT runs in two steps: Layer 0 cleans the objects and collects additional information (like MC match, trigger match, resolution information), which in layer 1 is merged into dedicated pat objects. For most analyses, Layer 1 is what you need. Creating layer 1 objects from 'plain' reconstructed CMSSW root files can be done by running the modules defined in `patLayer1_fromAOD_full.cfg`.

Take a look at `patLayer1_fromAOD_full.cfg` and identify the various steps in the PAT analysis. For example, look where the output is defined, where the .cff fragments are that control which modules are run, etc. Where and what is the name of the output file?

b) Run the PAT from AOD/RECO to Layer 1 by executing

```
cmsRun patLayer1_fromAOD_full.cfg
```

c) If your job ran successfully you should now have an output file. Open

- the file in a ROOT session and use the TBrowser to look at the content. You should notice that the number of objects has been significantly reduced. The pat creates six basic object types. What are those six basic types?*
- d) Examine the electron object. Identify the trigger, monte carlo and resolution information. Do the same for another object of your choice. Do all objects already have resolutions associated with them?*
 - e) There already is quite a lot of documentation available for the PAT. Take a look at the following wiki pages:*

<https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePAT>

Project 7: The StarterKit and making ntuples for private analysis

The StarterKit is one of the analysis tools that the CMS collaboration supplies for end-user analysis on top of the PAT output. As usual, the StarterKit (SK) is documented in the WorkBook¹¹. It is currently heavily under development but a working version can be created in CMSSW_1_6_11 with the following CVS tag¹²:

```
cvs co -r V00-00-06 PhysicsTools/StarterKit
```

*The appropriate CMSSW configuration file is **PhysicsTools/StarterKit/test/starterKitDemo.cfg**. This file runs on PAT output, so you would need your PAT output file that you created in Project 6 and make sure you are reading this file in with your .cfg.*

- a) Run the StarterKit*
- b) look at the output file in root with a TBrowser. Notice that the output is even further compressed, you now have histograms available for physics analysis.*
- c) make a plot of the p_T (transverse momentum) and eta (pseudorapidity) of the leading and second leading muon and electron.*
- d) The sample you ran on was a Higgs to 2 Z boson sample, where both Z*

¹¹ <https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookAnalysisStarterKit>

¹² Note that the list of tags that are supplied here have to be combined with the list of tags used for the PAT.

bosons decay to leptons. Try to create a Z boson resonance peak plot for electrons & muons, using the StarterKit.

e) And combine the two Z boson candidates to create a Higgs boson peak. You have just found the so-called golden-decay channel Higgs boson! This is the Higgs production process that both ATLAS and CMS were designed to be able to reconstruct very well.

WELL DONE!

7. Grid analysis and the crab analysis submitter

NOTE: skip this section of the tutorial if you do not have a valid grid certificate and are also registered in the CMS virtual organization (VO). Section 8 does not need grid access.

For this part of the tutorial you will need to log on with your own account at your own institution's grid User Interface (UI)¹³. Also, you will need a grid certificate as for obvious reasons it is impossible to run otherwise.

Later in this tutorial we will discuss the generation of small private MC datasets. This is the 'official CMS way' to generate small samples¹⁴ but much for obvious reasons larger samples are produced centrally by the collaboration. Using the collaboration's samples has the additional advantage that the MC generator settings are usually created by the experts and thus generally accepted as the 'official' signal samples. Also, the information is organized in a database, which ensures that the location and name of the datasets is easily retrieved. We will first focus on submitting jobs on existing datasets.

Project 8: Test that your job runs locally

You will need a CMSSW program to submit. For now, choose something simple (for example the exampleDump.cfg program from Project 2. Create a CMSSW area and a configuration file in the CMSSW_1_6_11/src directory. If you're feeling courageous you can copy the entire analysis module from Project 3 and compile. Test that your job runs.

Project 9: Test your grid certificate

Make sure you have your grid certificate installed at the UI you are planning to use. Create a grid certificate by typing

¹³ If you execute the setup script at login any of the lxplus machines can be used as UI.

¹⁴ In this context, any sample containing less than 50k events should be considered small.

`grid-proxy-init`

and if everything is properly installed you will be prompted for your grid password and your grid certificate will be created.

Note that you will also have to have registered with the CMS virtual organisation (VO) to do anything more than this, this can be tested by also activating your voms grid certificate:

`voms-proxy-init`

Which works exactly the same way as the general grid certificate activation above.

Project 9a: Run on data that is hosted at your local site

If your site hosts data locally it is possible to run on this data using Logical FileNames (LFNs). You will need a grid certificate to run interactively on these files.

Locate a MC file and run your exampleDump.cfg on it.

Project 10: Installing and running CRAB

Install a version of the crab analysis submitter¹⁵. FYI Crab is not the only grid submission program, there also is an alternative analysis submitter called ASAP. In this tutorial we will only look at job submission with crab.

a) Work through the crab tutorial here:

<https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookRunningGrid>.

As a program, use the exampleDump.cfg file that you created earlier (this will only produce additional printout output).

Make a copy of the standard crab config file

¹⁵ The newest version of crab is CRAB_2_1_1. This version or one above should produce acceptable behavior. On most machines, it is easy to install crab over AFS from CERN (a local install is also possible, of course). This means running the `/afs/cern.ch/cms/ccs/wm/scripts/Crab/CRAB_2_1_1/crab.(c)sh` script. The first time you will also have to instantiate the BOSS service, by running the `configureBoss` script in the same AFS directory.

and modify it so it runs your job. The important sections to focus on are:

- The dataset that you plan to use has to be compatible with the version of CMSSW that you use (so use one of the 1_6_? datasets). Choose an appropriate dataset from the DBS/DLS web interface¹⁶. The data-tier¹⁷ matters for the dataset you select. As we are planning to run on reconstructed data you should select the "RECO" data tier.
 - Make sure you include the name of your cmsRun .cfg file.
 - The number of events you want to run on needs to be in there too!
 - The possible inclusion or exclusion of particular grid sites by means of white and black lists. Be particularly careful if you don't know the exact name of computing or storage elements.
 - Returning the data. If you plan to create an output file (for example a root file) you will have to instruct crab to copy it back once the job has completed. In this project you will only produce additional standard output, so you do not need to specify where to copy your data.
 - Note that if you plan to create large samples you will not be able to copy the files back to your user node, the maximal output file is limited by the grid OutputSandBox, and is of the order of a few tens of MB. Anything larger needs to be copied directly to a storage element (SE) and crab can be instructed to do this by flagging the return_data parameter to true and providing an appropriate storage location in the crab script.
- b) Submit a few jobs with a few (<5) events each, and experiment with the submission options. Choose your white- and black-listing in a clever way, as for instance the CERN cluster tends to be very busy so your jobs will be queued for a long time.
- c) If time: see if you can run your analysis module on the same dataset.
- d) The crab team also keeps statistics, so make sure we create a nice peak in their submission plots¹⁸

¹⁶ https://cmsweb.cern.ch/dbs_discovery/index?userMode=user

¹⁷ <https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookDataFormats>

¹⁸ <http://cmsgridweb.pg.infn.it/crab/crabmon.php>

8. Small private MC production

The last part of this tutorial deals with MC production of small samples. The tutorial for this type of work is also already present in the workbook, but for CMSSW_1_6_0¹⁹. You can just follow the instructions but use CMSSW_1_6_11 instead.

When running private production, one of the most important things is to know what software you are running and (most preferably) how this compares to the settings used for the production of large samples by the collaboration. The tutorial can be found here:

<https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookEntireRecoChain>

script does this by including the same files as the ReleaseValidation and MC production samples, in the correct version of CMSSW.

Project 11: Creating your own simulated events

- a) Try to understand the different steps of the production, look at the “run everything” workbook tutorial.*
- b) Generate 10 $H \rightarrow \mu\mu\mu\mu$ events.*
- c) Check out the $H \rightarrow \mu\mu\mu\mu$ configuration file and modify it in such a way that you generate $H \rightarrow eeee$ events instead. Generate 10 events with a different output file name.*
- d) Create a Pythia module of your favourite physics process and create your own card file. Generate 10 events²⁰.*

¹⁹ <https://uimon.cern.ch/twiki/bin/view/CMS/WorkBookEntireRecoChain>

²⁰ <https://twiki.cern.ch/twiki/bin/view/CMS/HighLevelTrigger06ExerciseMC> Samples contains many CMSSW configuration files, you should be able to extract the necessary information from the relevant file.

Project 12: Running MC production on the grid

This final project involves running a small MC production on the grid. Modify your crab configuration file so it has your exampleRunAll.cfg file as input and make sure crab knows the name of your output file too.

a) Run a small MC production that returns the files to your local UI (so not to a

SE). Make sure the output files are small enough that they can be written to the

OutputSandBox²¹. Submit at least 10 jobs, preferably to different sites, which can be done by keeping the white lists empty.

b) IF POSSIBLE²²: Modify your crab configuration file such that you write back to your local SE. Note that it is very important to create a dedicated directory where you save the files as crab will not only over-write existing files with the same name but it is also non-trivial to delete files (or particularly rename them) once they are created. Create a 10kEvent sample of your favourite MC.

²¹ Effectively this means having less than 20 events in your output file. As soon as CU has access to a Storage Element you will be able to save your grid jobs straight on to the element, and will be able to run jobs of 500 events or more per job.

²² This is not possible yet. However, if you manage to get (grid) write access to another storage element you can directly write there. Which is why being nice to grid administrators always pays off in the end ☺