Table of Contents

4.2 CMSSW Analysis Starter Kit

Complete:
Detailed Review status

See the starter kit in the news! Symmetry Magazine article on Starter Kit.

Authors (alpha order):

- Steven Lowette <steven.lowette@cernSPAMNOT.ch>
- Petar Maksimovic <petar@jhuSPAMNOT.edu>
- Salvatore Rappoccio <rappocc@fnalSPAMNOT.gov>
- Elizabeth Sexton-Kennedy <sexton@fnalSPAMNOT.gov>
- Eric Vaandering <ewv@fnalSPAMNOT.gov>

What is the StarterKit?

- **Focus**: to create and implement the StarterKit, designed to facilitate "fast" analysis, to move from conception to a "reasonable" histogram within a day.
- **Who is the target audience?**: Anyone who wants to get a reasonable start on analysis, including professors, new grad students, theorists, postdocts, those unfamiliar with C++/ROOT/CMSSW, and many more!
- **What other tools are available?**: There are many other tools available for analysis in CMSSW, and the Physics Analysis Toolkit (PAT) is an attempt to consolidate them. We utilize the PAT as an analysis model. While there are other options available, this is the recommended versions for newcomers. We strongly encourage people to use the PAT for all of their analysis efforts, to avoid duplication of effort and to use well-understood objects and algorithms that are "blessed" by the Detector Performance Groups (DPG), Physics Object Groups (POG) and Physics Analysis Groups (PAG).
- **See us in the news!**: Symmetry magazine has run a piece on the CMS Starter Kit here: http://www.symmetrymagazine.org/cms/?pid=1000590

Structure of the StarterKit

The StarterKit is a group of C++ objects along with corresponding configuration files to plot specific examples. It is based on the Physics Analysis Toolkit (PAT) described in detail here:

- https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePATOld
- https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookPAT1612 (under construction)

The prescriptions are based on the PAT prescriptions as shown here:

- https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePATRecipes

The StarterKit has the following functionality:

- Will perform "common" plots for variables in the analysis such as 4-vector quantities and matching variables for detector objects.
- A tiered structure of output, whereas the user can specify to simply make plots, perform simple skims, or write out a subset of the PAT and/or AOD.
- Can automatically make plots for CompositeCandidate objects and "drill down" to the constituents recursively.
- Provides user control for histogram binning, etc. This will eventually be generalized using the ExpressionParser functionality.

Our programming to do list includes:

- Implementation of more complex examples.
- Generic kinematic fitting of arbitrary constituents.
- A validation suite to perform CRON jobs for day-to-day monitoring.
- Extensive FWLite examples for performing analysis.
- Automatic ntuplization of composite candidates.

Class structure:

- **PhysVarHisto**: "Smart" histogram which can also ntuplize.
- **HistoGroup**: Object to plot 4-vectors using PhysVarHisto.
- **HistoMuon, HistoElectron, HistoMET, HistoJet, HistoPhoton, HistoTau**: Derived classes to implement specific objects including matching variables.
- **PhysicsHistograms**: Contains and drives Histo objects.
- **StarterKit**: EDProducer to drive PhysicsHistograms

CMSSW in 90 seconds or less

If you've been following the workbook all along, this should be "old news" to you, but here is the "quick and dirty" introduction for the impatient user. CMSSW stands for "CMS Software", and is a coding framework for the high level trigger, the reconstruction, and the analysis all at the same time. This is a novel approach for a collaboration of this size. It creates "projects" in subdirectories based on static code "tags" (from CVS). It allows us to modularize the performance so that specific tags can be studied and compared.

The StarterKit makes a new "package" in a directory in your home area. It will check out some code and compile it, and you can run a program on a predefined set of output and check against the "correct" answer to make sure you're starting from a good place.

Full details on the CMSSW framework can be found here:
https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookCMSSWFramework

A 30 second introduction to scram and some commands can be found here:
https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookScramV1Intro

Prescription

Verbose version of prescription for pedagogy

NOTE! The following section is for pedagogy, it will not include all commands, please see next section for full "cut-and-paste" list of commands!

We will now discuss how to check out and compile the StarterKit. Note that this depends on the PAT prescription linked here:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePATRecipes#Tags_for_1_6_x

It also depends on the Top Quark Analysis Framework prescription for example "Event Hypothesis" demonstrations here:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideTQAF#Checkout_procedure_with_PATLayer

If you are using FNAL machines, be sure your .cshrc file includes

```
source /uscmst1/prod/sw/cms/cshrc uaf
```

If you are using lxplus, you do not need to include anything, as it is already set up.

Once the above commands are executed, you are ready to follow the checkout procedure in 1.6.11. Be sure you have access to a machine with the 1.6.11 version of CMSSW installed. *This prescription will not work in any release less than 1.6.11*.

NOTE: If you are not at Fermilab, you do not need to execute the "unsetenv PYTHONPATH" line below. Also, if you are at Fermilab, you can open a CVS repository session in two ways (if you are working at CERN, the CVS repository is automatically set up so you don't have to worry). First, you can use the "read-only" mode as described in the following commands

```
cmscvsroot CMSSW
cvs login
```

The password for anonymous CVS access is "98passwd". This sets up the anonymous CVS server access.

Alternatively, you can set up a write-read session by typing:

```
project CMSSW
kserver_init
```

and give your kerberos authentication at CERN (take care to use your CERN kerberos password, not your Fermilab kerberos password, if you are at Fermilab). The reason for this is a version mismatch between the Fermilab kerberos version and the CERN kerberos version.

For this exercise, we will use the anonymous authentication.

To create a new area, we will execute the command

```
scramv1 p CMSSW CMSSW_1_6_11
```

Then we will use CVS (the Concurrent Version System) to "check out" some packages with the following commands:

```
# check out backports and bug fixes:
    # Muon ID
cvs co -r V00-16-01-00 DataFormats/RecoCandidate
cvs co -r CMSSW_1_6_11 RecoMuon/CMS.MuonIdentification
cvs co -r V01-02-06-01 RecoMuon/CMS.MuonIdentification/src/IdGlobalFunctions.cc
cvs co -r V01-02-06-01 RecoMuon/CMS.MuonIdentification/interface/IdGlobalFunctions.h
cvs co -r V02-15-10-00 TrackingTools/CMS.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 update -r pat_tutorial_14may08 TopQuarkAnalysis/TopEventSelection TopQuarkAnalysis/TopJetC
cvs co -r TQAF_1611_080415 CondFormats/PhysicsToolsObjects
cvs co -r TQAF_1611_080415 PhysicsTools/MVATrainer PhysicsTools/MVAComputer
# check out the starter kit source code
cvs co -r V00-00-07 PhysicsTools/StarterKit
```

We will then set up the CMSSW software environment with the command

```
eval `scramv1 runtime -csh`
```

To compile the code, we will then execute the "build" command

```
scramv1 b
```

To speed up processing it is possible to split the build over several processors by

```
scramv1 b -j 4
```

The CMS model is to have everything put together into one big executable called "cmsRun", and your module gets added as a "plugin". Thus, to run the StarterKit, we will execute the command

```
cmsRun StarterKitDemo.cfg >& skoutput.txt &
```

Detailed description of output and running

The relevant configuration file is StarterKitDemo.cfg.

Expected output will be:

- **StarterKitHistos.root** : Root file containing output histograms.
- **StarterKit_output.txt** : Text file for your convenience, if you desire you can write out debugging information, by default is empty.
- **skoutput.txt**: Text file containing program output
- **StarterKitSkim.root**: Root file containing skimmed events.

If you run over more than a few hundred events, you must edit the file StarterKitDemo.cfg to write to a place you have write access to in a scratch area. The relevant portion is

```
module out = PoolOutputModule {
  untracked string fileName = "StarterKitSkim.root"
  ...
  untracked bool verbose = true
}
```

Change this to read

```
module out = PoolOutputModule {
  untracked string fileName = "/my_scratch/username/StarterKitSkim.root"
  ...
  untracked bool verbose = true
}
```

where "/my_scratch/username/" is the name of your scratch directory. For instance, if you are using lxplus, you can use "/tmp/[username]", and at Fermilab, you can use "/uscms_data/d1/[username]" after you create those directories.

For the impatient: To set up a new area and run

For the impatient, a "cut-and-paste" version that you can directly input to start a new area and run is as follows.

```
cmscvsroot CMSSW
cvs login
mkdir StarterKit
cd StarterKit

scramv1 p CMSSW CMSSW_1_6_11
cd CMSSW_1_6_11/src/
eval `scramv1 runtime -csh`
unsetenv PYTHONPATH
cvs co -r V01-02-04 PhysicsTools/PatAlgos
cvs co -r V01-02-03 PhysicsTools/PatUtils
cvs co -r V01-02-03 DataFormats/PatCandidates
cvs co -r V00-16-01-00 DataFormats/RecoCandidate
cvs co -r CMSSW_1_6_11 RecoMuon/CMS.MuonIdentification
cvs co -r V01-02-06-01 RecoMuon/CMS.MuonIdentification/src/IdGlobalFunctions.cc
cvs co -r V01-02-06-01 RecoMuon/CMS.MuonIdentification/interface/IdGlobalFunctions.h
cvs co -r V02-15-10-00 TrackingTools/CMS.TrackAssociator
cvs co -r V02-04-04-06 JetMETCorrections/Type1MET
cvs co -r B160_V00-04-06 PhysicsTools/JetMCAlgos
cvs co -rNamedCompositeCandidate_1611 DataFormats/Candidate
cvs co -r pat_1611_080425 PhysicsTools/CandUtils/
cvs co -r pat_1611_080425 PhysicsTools/CandAlgos/
cvs co -r PAT_1611_080425 PhysicsTools/HepMCCandAlgos
cvs co -r V00-00-07 PhysicsTools/StarterKit
cvs co -r pat_tk_tutorial_080424 AnalysisDataFormats/TopObjects TopQuarkAnalysis
cvs update -r pat_tutorial_14may08 TopQuarkAnalysis/TopEventSelection TopQuarkAnalysis/TopJetC
cvs co -r TQAF_1611_080415 CondFormats/PhysicsToolsObjects
cvs co -r TQAF_1611_080415 PhysicsTools/MVATrainer PhysicsTools/MVAComputer
scramv1 b -j 4
cd PhysicsTools/StarterKit/test
cmsRun StarterKitDemo.cfg >& skoutput.txt &
tail -f skoutput.txt
```

For the impatient: To re-setup an old area and run

If you've already made a package and want to start again in the same package:

```
cmscvsroot CMSSW
cvs login
```

```
cd StarterKit/CMSSW_1_6_11/src
eval `scramv1 runtime -csh`
cd PhysicsTools/StarterKit/test
cmsRun StarterKitDemo.cfg >& skoutput.txt &
tail -f skoutput.txt
```

Histogram Output

The first thing you should check is the Histogram file StarterKit.root. If you copy the rootlogon.C file into your current directory, you can open root like this:

```
your_pc:%> root StarterKitHistos.root
```

The output will look something like this:

```
  ****************************************
  *                                      *
  *        W E L C O M E  to  R O O T    *
  *                                      *
  *   Version   5.16/00      27 June 2007  *
  *                                      *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch         *
  *                                      *
  ****************************************

Compiled on 28 June 2007 for macosx with thread support.

CINT/ROOT C/C++ Interpreter version 5.16.21, June 22, 2007
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
libraries loaded
root [0]
Attaching file StarterKitHistos.root as _file0...
root [1]
```

You are now in the command line for the ROOT program. If you wish, you can turn on the statistics box by doing the following:

```
root [1] gStyle->SetOptStat(111111);
```

You can specify between one and six "1's", which correspond to the name, number of entries, mean, standard deviation, underflow, and overflow bins.

You can now execute the following command:

```
root [1] TBrowser b;
```

This will bring up a browser so that you can examine your StarterKitHistos.root file. It should look like this:

Click on the "ROOT Files" directory to descend into the next level, which should look like this:

Next click on StarterKitHistos.root.

Next click on "StarterKitDemo"

Next click on "muon"

Finally make a plot for the muon pt by clicking on "muPt_1":

That's it! You've successfully made a plot that is sensible physics quality using the StarterKit prescription!

Exercise for the User:

We would like you to try the following things:

- Examine all the different variables and make sure you understand them from a physics perspective. This is a Z to mu mu sample, so you should understand the physics associated with that (i.e. what the pt spectrum should look like, etc). If there is something wrong, or something you don't understand, please make the plot, and send it to the authors above so that we can open a dialog.
- Attempt to change the PhysicsTools/StarterKit/test/StarterKitDemo.cfi configuration file to run over all events and produce larger statistics histograms. This will let us know how well you understood how to run things. To run over all events, **first change the output file to a scratch area as described above**. Then, change

```
untracked PSet maxEvents = { untracked int32 input = 200 }
```

to read

```
untracked PSet maxEvents = { untracked int32 input = -1 }
```

You do not need to recompile, just execute the commands

```
mv StarterKitHistos.root StarterKitHistos_smallstats.root
cmsRun StarterKitDemo.cfg >& z_mumu.txt &
```

If you drill down back to "muPt_1" again, you should obtain a plot that looks like this:

Ntuple Output

We now turn to the file in the scratch directory that you have been ignoring until now, /my_scratch/username/StarterKitSkim.root. This file (as all CMSSW files) can be opened directly in ROOT to examine its contents. First, we make sure that we have copied the [https://twiki.cern.ch/twiki/pub/CMSPublic/WorkBookAnalysisStarterKit1611/rootlogon.C][rootlogon.C] file to your home directory, and then type the following commands:

```
your_pc:%> root /my_scratch/username/StarterKitSkim.root
```

Be sure to substitute your scratch area that you wrote to before instead of "/my_scratch/username/StarterKitSkim.root".

Within ROOT, we type

```
root [1] TBrowser b;
```

You will see another TBrowser window. If you click on "ROOT files" again and click on "/my_scratch/username/StarterKitSkim.root", you will see a window that looks like this:

Click on "Events" and you will see this:

Each of the branches in this directory correspond to a variable that you have requested to be ntuplized. It also contains the PAT objects and some selected objects from the AOD. The variables we have booked are examples of EDMNtuples:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideEDMNtuples

If you click on one of the leafs (say, muon pt) you see:

The object that you're interested in if you want to plot the variable is the leaf that ends in ".obj", which stores the actual object. If you click on the ".obj" file in the "muon pt" leaf, you will get a plot that looks like this:

Command Line Analysis

At this point, we can go back to the ROOT command line and do some more interesting things. If you type

```
root [2] Events
(class TTree*)0x9df8de0
root [3] Events->SetAlias("muPt", "doubles_StarterKitDemo_muPt_StarterKit.obj");
```

you can create an "alias" so you don't have to type out "doubles_blablabla" all the time. Instead, you can now type

```
root [4] Events->Scan("muPt");
***********************************
*    Row   * Instance *      muPt *
***********************************
*        0 *        0 * 32.312796 *
*        1 *        0 * 45.212468 *
*        1 *        1 * 28.393791 *
*        2 *        0 * 68.051852 *
*        2 *        1 * 16.254379 *
*        3 *        0 *           *
*        4 *        0 * 45.218302 *
*        4 *        1 * 44.077946 *
*        5 *        0 *           *
*        6 *        0 * 49.040454 *
*        6 *        1 * 41.983508 *
*        7 *        0 *           *
*        8 *        0 * 39.459028 *
*        8 *        1 * 36.914808 *
*        9 *        0 * 16.723143 *
*       10 *        0 * 35.300636 *
*       11 *        0 * 43.026732 *
*       11 *        1 * 27.839282 *
*       12 *        0 * 55.072839 *
*       12 *        1 * 30.628422 *
*       13 *        0 * 39.528696 *
*       13 *        1 * 36.673263 *
*       14 *        0 * 13.156261 *
*       15 *        0 * 27.869177 *
*       16 *        0 *           *
Type <CR> to continue or q to quit ==>
```

This starts to scan through the objects in the "muPt" branch. You can also start making more advanced plots in principle, but we'll just plot the muon pt of the first muon. Hit "q" to "quit" the scan over the muon pt, and

type

```
root [5] Events->Draw("muPt[0]");
```

You should obtain a plot that looks like this:

Now, you can also make cuts on your ntuple as follows:

```
root [6] Events->Draw("muPt[0]", "muPt[0] > 30");
```

This cuts on the muon pt to be greater than 30 GeV/c. You should obtain something that looks like this:

This should give you the flavor of the types of things that can be done with the command line interface. This behaves exactly like a standard ROOT TTree after you have made these alias commands, and you should be able to do whatever you like with the TTree functionality on the command line.

We now turn to using derivatives of TSelectors, the FWLite::Event model.

Using FWLite::Event

To do any sort of "meaningfully complicated" analysis, it will be necessary to use a macro instead of a command line argument. To do so, Chris Jones and others have developed utilities to do this, which you can read about here:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideEDMWithRoot

The StarterKit is (in principle) fully compatible with FWLite but at the moment we have not yet implemented a test suite for those tools. However there are easily available tools for running on the "ntuple" output of the StarterKit which can be derived simply from the above link.

For the following example, be sure that you use the attached rootlogon.C file. This will set up the environment. Then, you can open the file sk_fwlite.C, and edit it to point to the proper skim file (where you renamed /my_scratch/username/StarterKitSkim.root to go).

Then, you can type

```
root[1] .x sk_fwlite.C++
```

To run our example. You should obtain a plot that looks like this:

NamedCompositeCandidate Example

CAVEAT: At the moment, the ntupling of composite variables is not yet implemented in a very elegant way, so please be patient as this comes in future versions. Only the collections are ntupled at this time, and a hard-coded ntuplization of the Z mass. With that limitation, we will now start examining an example which demonstrates the capability of the StarterKit to handle composite candidates outlined here:

https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookParticleCandidates#NamedCompositeCandidate

The class NamedCompositeCandidate is designed to represent a hierarchical event structure along with names for the candidates, and "role" strings for the daughters. This is a model of the PAT "Layer 2" implementation that is primarily useful for hierarchical "true" decays.

There are several ways to use NamedCompositeCandidates.

- The first way is to use generic tools in a selector interface as described here:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCandCombiner#CandCombiner_Framework_modules

- The second way is to incorporate NamedCompositeCandidates in existing "Layer 2" event hypothesis models. We have developed a TQAF class (TtSemiEvtSolution) that implements this functionality.

We now consider these two examples of "Layer 2" hypotheses in turn.

Using Generic Candidate Combiners

In this example we will consider Z to mumu events where we can automatically histogram various 4-vector quantities by simply specifying the process via composite candidates. We will therefore be examining the HZZKit also provided in Release 1.0 of the StarterKit. To do so, please return to the "test" directory in StarterKit and execute with the HZZKit config file as follows:

```
cd ~/StarterKit/CMSSW_1_6_11/src/CMS.PhysicsTools/StarterKit/test
nice +19 cmsRun HZZKitDemo.cfg > & hzz_demo.txt &
```

If you'd like, you can change the output skim file in this example as well, which is set to "HZZKitSkim.root". You can change the file "HZZKitDemo.cfg" in the same way as previously.

The meat of this example is provided in a configuration file using the Generic Candidate Tools shown here:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCandCombiner

We use the "named" analogue of CandViewShallowCloneCombiner as follows:

```
### H->ZZ ###
module zToMuMu = NamedCandViewShallowCloneCombiner {
  string decay = "selectedLayer1Muons@+ selectedLayer1Muons@-"
  string cut = "0.0 < mass < 20000.0"
  string name = "zToMuMu"
  vstring roles = { 'muon1', 'muon2' }
}

module hToZZ = NamedCandViewCombiner {
  string decay = "zToMuMu zToMuMu"
  string cut = "0.0 < mass < 20000.0"
  string name = "hToZZ"
  vstring roles = { 'Z1', 'Z2' }
```

```
  }
```

In the first part, we create Z candidates with roles "muon1" and "muon2". We make the output collection named "zToMuMu", which is then input into the Higgs candidate creation, "hToZZ". In this, there are also two roles, "Z1" and "Z2". The output collection is named "hToZZ" and can be directly input into the HZZKit.

We can then examine the output of HZZKitDemo (HZZKit.root) as follows:

```
root HZZKit.root
  ******************************************
  *                                        *
  *        W E L C O M E  to  R O O T      *
  *                                        *
  *   Version  5.14/00f       29 May 2007  *
  *                                        *
  *  You are welcome to visit our Web site *
  *          http://root.cern.ch           *
  *                                        *
  ******************************************

FreeType Engine v2.1.9 used to render TrueType fonts.
Compiled on 23 September 2007 for linux with thread support.

CINT/ROOT C/C++ Interpreter version 5.16.16, November 24, 2006
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
TDR Style initialized
root [0]
Attaching file HZZKit.root as _file0...
root [1] TBrowser b;
```

I'll be less pedantic for this part, since you should already be familiar with the TBrowser interface at this point after the previous examples. In this case, simply drill down to "ROOT Files/HZZKit.root/HZZKitDemo" in the TBrowser and you should get something that looks like this:

Notice that there are now two additional folders, "hCand" and "zmumuCand". If you click on "hCand", you get a window that looks like this:

If you click on "Z1Mass_1", you see something that looks like this:

Congratulations! You've already made your first invariant mass peak at CMS!

Now, in the same directory, clicking on "hMass_1", you'll see

Congratulations! You've already made your second invariant mass peak at CMS!

Quick Plotting of any

In the previous example we had a dedicated module to make plots from `H->ZZ->4mu`. In this example we will use the same concept but a different tool that is more generic. You will run into different circumstances where you want either of these two options, so you'll want to be aware of how to use both. The automatic composite candidate plotting kit is fantastically useful for getting quick plots out of any given input composite candidate collection, but if you want to do more and more complicated things you'll want the flexibility of coding everything yourself.

At this point I'll assume you're familiar with the basics so I won't go into much detail.

The code snippet to do the generation is similar to that of the previous section, with the modification being the plotting kit is the generic CompositeKit.

```
### Starter Kit ###
include "CMS.PhysicsTools/StarterKit/test/CompositeKitDemo.cfi"

service = TFileService {
  string fileName = "CompositeKit.root"
}

### H->ZZ ###
module zToMuMu = NamedCandViewShallowCloneCombiner {
  string decay = "selectedLayer1Muons@+ selectedLayer1Muons@-"
  string cut = "0.0 < mass < 20000.0"
  string name = "zToMuMu"
  vstring roles = { 'muon1', 'muon2' }
}

module hToZZ = NamedCandViewCombiner {
  string decay = "zToMuMu zToMuMu"
  string cut = "0.0 < mass < 20000.0"
  string name = "hToZZ"
  vstring roles = { 'Z1', 'Z2' }
}

# Filter on number of  desired composite candidates
module compositeFilter = CandViewCountFilter{
  InputTag src = hToZZ
  uint32 minNumber = 1
}

### Output ###
path p = {
  patLayer0,
  patLayer1,
  zToMuMu,
  hToZZ,
  compositeFilter,
  CompositeKitDemo
}
```

You can execute this with the command

```
cmsRun CompositeKitDemo.cfg
```

The one small addition to this package is that it has a separate folder and invariant mass for the topmost level of the CompositeCandidate tree. This is to work around the temporary situation before we have the full capability of the ExpressionHistogram-enabled histogram objects for the Starter Kit. It is stored in the folder marked `resonance`.

We will make use of this later in the tutorial during the question and answer session.

Using more complicated Layer 2 classes

In the previous example we looked at automatically generated hypotheses. In this example, we will use a ttbar sample to demonstrate the power of using a Layer 2 module such as NamedCompositeCandidates in another class to interface with the Starter Kit framework.

To do so, we use the TQAF described here:

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideTQAF

We have modified one Layer 2 hypothesis in TtSemiEvtSolution in the TQAF to have event hypotheses worked in to provide an interface for common tools. The implementation is

```
  reco::NamedCompositeCandidate mcHyp_;
  reco::NamedCompositeCandidate recoHyp_;
  reco::NamedCompositeCandidate fitHyp_;
```

In the corresponding cc file, the relevant implementation is

```
void TtSemiEvtSolution::setupHyp()
{

  AddFourMomenta addFourMomenta;

  recoHyp_.clearDaughters();
  recoHyp_.clearRoles();

  // Setup transient references
  reco::NamedCompositeCandidate recHadt;
  reco::NamedCompositeCandidate recLept;
  reco::NamedCompositeCandidate recHadW;
  reco::NamedCompositeCandidate recLepW;

  // Get refs to leaf nodes
  JetCandRef hadp( hadp_->p4(), hadp_->charge(), hadp_->vertex());  hadp.setRef( hadp_ );
  JetCandRef hadq( hadq_->p4(), hadq_->charge(), hadq_->vertex());  hadq.setRef( hadq_ );
  JetCandRef hadb( hadb_->p4(), hadb_->charge(), hadb_->vertex());  hadb.setRef( hadb_ );
  JetCandRef lepb( lepb_->p4(), lepb_->charge(), lepb_->vertex());  lepb.setRef( lepb_ );

  METCandRef neutrino  ( neutrino_->p4(), neutrino_->charge(), neutrino_->vertex() ); neutrino.se

  recHadW.addDaughter( hadp,    "hadp" );
  recHadW.addDaughter( hadq,    "hadq" );

  addFourMomenta.set( recHadW );

  recHadt.addDaughter( hadb,    "hadb" );
  recHadt.addDaughter( recHadW, "hadW" );

  addFourMomenta.set( recHadt );

  recLepW.addDaughter( neutrino,"neutrino" );
  if ( getDecay() == "electron" ) {
    ElectronCandRef electron ( electron_->p4(), electron_->charge(), electron_->vertex() ); elect
    recLepW.addDaughter ( electron, "electron" );
  } else if ( getDecay() == "muon" ) {
    MuonCandRef muon ( muon_->p4(), muon_->charge(), muon_->vertex() ); muon.setRef( muon_ );
    recLepW.addDaughter ( muon, "muon" );
  }

  addFourMomenta.set( recLepW );

  recLept.addDaughter( lepb,    "lepb" );
  recLept.addDaughter( recLepW,    "lepW" );

  addFourMomenta.set( recLept );

  recoHyp_.addDaughter( recHadt, "hadt" );
  recoHyp_.addDaughter( recLept, "lept" );

  addFourMomenta.set( recoHyp_ );
}
```

This is a demonstration of how to build up an event hypothesis by hand to be used in generic tools downstream. It also demonstrates how to properly utilize the CandidateWithRef class to avoid duplication of large objects, and instead stores Ref's in the NamedCompositeCandidate to save space. The Starter Kit framework automatically detects this case and handles it for you with no further effort on your part.

To run over this hypothesis, we will use the TtSemiEvtKit as follows:

```
cd ~/StarterKit/CMSSW_1_6_11/src/CMS.PhysicsTools/StarterKit/test
nice +19 cmsRun TtSemiEvtKit.cfg > & ttsemimu.txt &
```

The output will be TtSemiEvtKitHistos_1611.root.

If you drill down to the object browser window, it should look something like this:

If you click on "ttSemiEvt", you will see

If you click on "hadtMass_1", you will see the hadronic top invariant mass.

Congratulations! You have made your third invariant mass at CMS!

Now it's time to treat yourself to a coffee. And hey, why not something sweet to go with it!

Common Errors

Here we discuss some common errors from users.

PYTHON errors.

If you see something like

```
Traceback (most recent call last):
  File "/uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/root/5.14.00f-CMS3q/bin/../lib/python/genreflex
    import sys, os, gendict, selclass, gencapa, genrootmap, string, getopt
  File "/uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/root/5.14.00f-CMS3q/lib/python/genreflex/gendic
    import xml.parsers.expat
  File "/build1/162p1/slc4_ia32_gcc345/external/python/2.4.2-CMS3q/lib/python2.4/xml/parsers/expa
ImportError: /opt/gLite/external/usr/lib/python2.3/lib-dynload/pyexpat.so: undefined symbol: PyUn
```

Then you need to type

```
unsetenv PYTHONPATH
```

and run again.

Output file errors.

If you see something like

```
StorageMaker::open() File '///my_scratch/username/StarterKitSkim.root' is not found or could not
File operation open() failed (because of System error: No such file or directory (<#2,#3>))cms::E
```

Then you need to edit the config file StarterKitDemo.cfg to point to a valid storage location that you have access to, as described above.

scram project errors

If you see something like

```
Can't locate XML/Parser/Expat.pm in @INC (@INC contains: /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lc
BEGIN failed--compilation aborted at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/
Compilation failed in require at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/src/
BEGIN failed--compilation aborted at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/
Compilation failed in require at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/src/
BEGIN failed--compilation aborted at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/
Compilation failed in require at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/src/
BEGIN failed--compilation aborted at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/
Compilation failed in require at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/src/
BEGIN failed--compilation aborted at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/
Compilation failed in require at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/bin/
BEGIN failed--compilation aborted at /uscmst1/prod/sw/cms/slc4_ia32_gcc345/lcg/SCRAMV1/V1_0_3-p2/
```

Thanks to Patrick Gartung and Catalin Dumitrescu, we have a solution for this.

Ensure that the following lines are **not** in your .cshrc file:

```
if ( -f "/afs/fnal.gov/ups/etc/setups.csh" ) then
 source "/afs/fnal.gov/ups/etc/setups.csh"
endif
```

```
if ( { ups exist shrc } ) then
 setup shrc
endif
```

Feedback

We have developed a questionnaire for you to fill out to help us help you! Please copy and paste the following into an email to the authors above when you are finished with the Requests, and give us your feedback. This is text-only, so be patient with us!

```
These are exclusive categories, so please select the one that most describes you:

1) How familiar are you with C++

[ ] a) I could code a std::map<std::string,std::vector<int> > and have no problem.
[ ] b) I know what public inheritance means but I can't figure out what you're talking
         about in (a).
[ ] c) I'm still mad we went away from F77.

2) How familiar are you with ROOT?

[ ] a) I've successfully added my own classes to ROOT.
[ ] b) I can book my own histograms and plot them without trouble.
[ ] c) I still use PAW.

3) Have you performed CMSSW analyses before?

[ ] a) Yes, and I probably could actually have written this better than you.
[ ] b) Yes, but not at the expert level.
[ ] c) Are you kidding? You're supposed to be doing this for me!

4) Have you used FWLite?

[ ] a) Yes, I've used it and I love it!
[ ] b) Yes, I've used it but it needs some work.
[ ] c) No.

5) How long did it take you to get from cvs checkout procedure to a plot?

[ ] a) Less than an hour.
[ ] b) Between an hour and a day.
[ ] c) More than a day.

6) What analysis are you planning on doing in CMS?

Comment:

7) Did you like this type of framework?

[ ] a) This will be great when you get it working. Suggestions below.
[ ] b) This is too complicated, make it simpler. Suggestions below.
[ ] c) I thought it was awful and didn't accomplish any of the goals I needed it to do.
         Suggestions below.

Suggestions:

8) Any further comments or questions:

Comment:
```

#CMS.ToDoList

Starter Kit To-Do List

As you can see, we have a fairly functional Starter Kit to start you off with. The last pieces are coming together in the coming weeks. They include

- Generic histogramming utilities. These will include the following capabilities:
    - Config file booking of simple and more complex variables (pt(jet1) / pt(jet2), for instance).
    - More user control over histogram binning and axis limits.
    - Automatic ntuplization (already present, but will retain this feature).
    - "Before and after" plots given some cuts.
- Full list of POG-blessed plots for each physics object.
- Automated nightly validation scripts for monitoring purposes.
- (Not specific to starter kit): event selection on methods that are in derived classes to `Candidate`.
    - b-tagging
    - lepton isolation
    - number of components in jets
    - etc

Trial Cases for Tutorials

I have called for a request from users who would like to see specific use cases in the tutorial session on 15May08. Here is the list so far.

- Usage of jet energy corrections (Ia Iashvili)
- Dileptons from Z (Dimitri Bourilkov)
- `H->ZZ->ll + jj` (Youn Roh)
- Z+jet and Gamma+jet (`Pt-Jet1 / Pt(Z->mu+mu) and Pt-Jet1/Pt-photon`) (Anwar Bhatti)
- Accessing b-tagging information and flavor ID (Jochen Cammin)
- Accessing trigger information (Francisco Yumiceva)
- ttbar resonance production (Salvatore Rappoccio... hey, I'm allowed to request things too!)
- MC Truth matching (Dmitry Hits)
- 2-d histograms (Dmitry Hits): **This feature is not yet available, but is coming soon.**
- lepton + jets with btagging (Cecilia Gerber): **This feature will be made simpler with the upcoming features of the Expression Histograms**.

I have answered these case studies in the WorkBookAnalysisStarterKitPatTutorial14May08 TWiki.

Review status

Reviewer/Editor and Date	Comments
Main.Aresh - 21 Feb 2008	changes in verbatim elements because of some lines too long for printable version
SalvatoreRappoccio - 03 Dec 2007	page author

Responsible: SalvatoreRappoccio
Last reviewed by: *Never reviewed*

This topic: CMSPublic > WorkBookAnalysisStarterKit1611
Topic revision: r4 - 2009-05-12 - RogerWolf