* Code structure:
  + header files are in: package/include
  + source codes are in: package/src/common
  + python scripts are in: package/python
* Important packages:
  + **framework** → **very basic package**→ all the other packages can use the classes and methods defined in it
  + hadronic → hadronic analysis basically but several classes and methods can be used as basis as well.
  + WPol2, onelepton … → go into them when you use them
* Instruction:
  + AlphaTPlots is Bryn’s analysis package, so it’s better to create **your own one**.
    - Just copy the package to another name and have a look at the WeeklyUpdatePlots.hh/.cc, you will have some ideas what he is doing for writing the code
  + Creating your own classes similarly as in WeeklyUpdatePlots, you will found it is composed of several parts mainly:
    - Constructor/de-constructor of course
    - **StandardPlots**() define your histograms
    - **StandardPlots(ev)** fill your histograms
  + Important running scripts under python would be **batchGolden.py** (basic running script and called by McSmeared.py) and **McSmeared.py**(just an example, you can change accordingly)
* Running structure:
  + When source code compiled (“make –f Debug” under each used package), lib is produced and modules in python is produced and linked to the lib through your definition in Python.cc. And the modules can be used in **your own package** in the running script under python.
  + Data accessing (i.e. initialized of the c++ code) is done in Manage.cc under framework.

**Statements below will give some code examples (codes normally under AlphaTPlots ) for your understanding easily**

* Go into the packages

1. framework:
   1. Jet.cc/.hh, Lepton.cc/.hh, Photon.cc/.hh → **define jets and letpons**:
      1. What’s a jet or a lepton, i.e. passing identification….
      2. What’s the basic kinematics, isolation….
   2. JetData.cc/.hh, LeptonData.cc/.hh, PhotonData.cc/.hh → **define common jets and leptons** given the options of identification criteria definition,  cut value, isolation requirement, η requirement and overlap removal or not with other ojects…
      1. e.g. in WeeklyUpdatePlots, ev.JD\_CommonJets(). JD\_CommonJets() is redefined in EventData.cc/.hh and is the same as CommonJets()
   3. Compute\_Helper.cc/.hh → **further define event based variables**, like the HT, MHT, alphaT, MET…..
      1. e.g. in WeeklyUpdatePlots, ev.CommonHT()
   4. EventData.cc/.hh → connects all the classes defined in Jet.cc/.hh, Compute\_Helper.cc/.hh to an event in CMS.
      1. e.g. in WeeklyUpdatePlots, ev.JD\_CommonJets()
   5. CommonOps.cc/.hh → define the **cut class and methods** which will be used in the analysis. E.g. alpha\_tCut (cut on alphaT).
   6. Python.cc → define the lib and compile the cut classes into modules which will be used in the analysis
      1. e.g. in batchGolden.py OP\_CommonMHTCut, OP\_CommonAlphaTCut …..
   7. Other source code are just the templates or basic classes which will be derived by Compute\_Helper and EventData
   8. **python/icf/core.py**
      1. It defines basic running controls
         1. what’s the running model, batch or terminal?
         2. if batch→ how many jobs are splitted, where to write the output… (we have discussed a lot before during the work of making code running on BlueCrystal)
      2. It control the running of the code
         1. Under class Analysis, a method Run(): under McSmeared.py

anal\_ak5\_caloMC=Analysis("AK5Calo")

addCutFlowMC(anal\_ak5\_caloMC)

…..

anal\_ak5\_caloMC.Run(outdir,conf\_ak5\_caloMC,MC)

* + 1. It defines how a PSet is read. And the PSet is a directory. See python/icf/config.py and SUSYv2/samples/python/montecarlo/Summer11 as examples to understanding
  1. **python/icf/config.py**
     1. a list a PSet defined in the source codes

1. hadronic:
   1. The most important would be the muon/electron/jet filter that will be used in the event selection of your analysis.
      1. E.g. in AlphaTPlots/python/ McSmeared.py:

vbtfMuonId\_cff = Muon\_IDFilter( vbtfmuonidps.ps() )

vbtfElectronIdFilter = Electron\_IDFilter( vbtfelectronidWP95ps.ps() )

ra3PhotonIdFilter = Photon\_IDFilter( ra3photonidps.ps() )

def addCutFlowMC(b) :

b.AddMuonFilter("PreCC",vbtfMuonId\_cff)

b.AddPhotonFilter("PreCC",ra3PhotonIdFilter)

b.AddElectronFilter("PreCC",vbtfElectronIdFilter)

* + 1. They are defined under hadronic/python/ra1objectid and they are based on the modules produced in framework and you can see under framework/python/icf/config.py

1. AlphaTPlots:
   1. Your analysis package (you can create your own one)
   2. Read **WeeklyUpdatePlots.cc/.hh, Python.cc and Ops.cc/.hh**
   3. Read batchGolden.py
      1. Add cut flows in MakeDataTree/ MakeMCTree
      2. Objects (jets, muon, electron…) selection is defined in default\_common
      3. Overlap checks is default\_cc
2. **Extra scripts:**
   1. During my analysis, I add some classes in framework and onelepton packages. But haven’t committed
   2. Since framework/python/icf/core.py relies on your running file name when submit different jobs, I write a script to set different parameters and creating different running files to submit, it’s quite useful. Also not committed.