Event View

G. Watts (UW)
O. Harris (UW)

Philosophy

EventView Goals

Object Identification & Interpretation

Is that a jet or an electron?
Is that jet a light quark or a b-quark jet?

} Each is a *view* of an event

Declarative Analysis

Give me all events with two good electrons with $p_T>15$ GeV/c^2.
Give me the invariant mass of the two electrons

Don't have to write code for most analyses (or build!): just author up a jobOptions file.
Produces a large ntuple with all requested information
Very attractive way to start

Dump Everything…

```
include("EventViewConfiguration/EventViewFullInclude_jobOptions.py")
from EventViewConfiguration import *

theJob = AlgSequence()

defaultEVLooper = EVMultipleOutputToolLooper("defaultEVLooper")
defaultEVLooper.EventViewCollectionOutputName="defaultEventView"
theJob += defaultEVLooper

# define insertion order, you may want to play with this order
toInsert=["Electron", "Photon", "Muon","TauJet","JetTag", "ParticleJet", "MissingEt",
"EventInfo"]

defaultEVLooper += DefaultEventView("Inserters", toDo=toInsert)
defaultEVLooper.Inserters.MuonInserter.setProperties( etCut=0.1*GeV)
```

Dump Everything…

1

```
include("EventViewConfiguration/EventViewFullInclude_jobOptions.py")
from EventViewConfiguration import *

theJob = AlgSequence()

defaultEVLooper = E
defaultEVLooper.Eve
theJob += defaultEVL

# define insertion order, you may want to play with this order
toInsert=["Electron", "Photon", "Muon","TauJet","JetTag", "ParticleJet", "MissingEt",
"EventInfo"]

defaultEVLooper += DefaultEventView("Inserters", toDo=toInsert)
defaultEVLooper.Inserters.MuonInserter.setProperties( etCut=0.1*GeV)
```

Basic Setup. Include all the various default EventView tools.

Dump Everything…

1

```
include("EventViewConfiguration/EventViewFullInclude_jobOptions.py")
from EventViewConfiguration import *

theJob = AlgSequence()

defaultEVLooper = EVMultipleOutputToolLooper("defaultEVLooper")
defaultEVLooper.EventViewCollectionOutputName="defaultEventView"
theJob += defaultEVLooper

# define inse
toInsert=["E
"EventInfo"]

defaultEVLooper += DefaultEventView("Inserters", toDo=toInsert)
defaultEVLooper.Inserters.MuonInserter.setProperties( etCut=0.1*GeV)
```

The EventView host tool. All EV tools are made to be *sub-tools* of these guys. It will loop over all the objects that get inserted (next step).

Dump Everything…

1

include("EventViewConfiguration/EventViewFullInclude_jobOptions.py")
from EventVi

theJob = Alg

defaultEVLo
defaultEVLo
theJob += d

Insert the list of objects you are interested in looking at. You have to know the names (see source code for DefaultEventView). Objects that can have dual labels are inserted in order (i.e. good electron before jet).

You can adjust the selection cuts after that (default p_T is 15 GeV).

# define insertion order, you may want to play with this order
toInsert=["Electron", "Photon", "Muon","TauJet","JetTag", "ParticleJet", "MissingEt", "EventInfo"]

defaultEVLooper += DefaultEventView("Inserters", toDo=toInsert)
defaultEVLooper.Inserters.MuonInserter.setProperties(etCut=0.1*GeV)

Dump Everything…

```
include("EventViewConfiguration/EventViewFullInclude_jobOptions.py")
from EventVi

theJob = Alg

defaultEVLo
defaultEVLo
theJob += de
```

Insert the list of objects you are interested in looking at. You have to know the names (see source code for DefaultEventView). Objects that can have dual labels are inserted in order (i.e. good electron before jet).

You can adjust the selection cuts after that (default p_T is 15 GeV).

```
# define insertion order, you may want to play with this order
toInsert=["Electron", "Photon", "Muon","TauJet","JetTag", "ParticleJet", "MissingEt",
"EventInfo"]

defaultEVLooper += DefaultEventView("Inserters", toDo=toInsert)
defaultEVLooper.Inserters.MuonInserter.setProperties( etCut=0.1*GeV)
```

Dump Everything…

```
#schedule the module
defaultEVLooper += DefaultEventViewLabels("Labeler", defaultEVLooper.Inserters)

# Write UserData to AANtuple
defaultEVLooper+=AANtupleFromUserData("defaultAADumper",
filename="test.AAN.root",sequencer = theJob)

defaultEVLooper += UserDataDump("UserDataDump",
Labels=defaultEVLooper['Labeler']._Labels)

# Dump everything to screen
#defaultEVLooper+=EVScreenDump("defaultScreenDumper", printUD=False)

# print the whole job schedule
print theJob

EventSelector.InputCollections = [ "AOD.pool.root"]
```

Dump Everything…

II

```
#schedule the module
defaultEVLooper += DefaultEventViewLabels("Labeler", defaultEVLooper.Inserters)

#Write UserData to AANtuple
defaultEVL
filename="t
```

Exports the Labels that are given to the various objects (not sure why this isn't automatic).

```
defaultEVLooper += UserDataDump("UserDataDump",
Labels=defaultEVLooper['Labeler']._Labels)

# Dump everything to screen
#defaultEVLooper+=EVScreenDump("defaultScreenDumper", printUD=False)

# print the whole job schedule
print theJob

EventSelector.InputCollections = [ "AOD.pool.root"]
```

Dump Everything…

```
#schedule the module
defaultEVLooper += DefaultEventViewLabels("Labeler", defaultEVLooper.Inserters)

# Write UserData to AANtuple
defaultEVLooper+=AANtupleFromUserData("defaultAADumper",
filename="test.AAN.root",sequencer = theJob)

defaultEVLooper += UserDataDump("UserDataDump",
Labels=defaultEVLooper['Labeler']._Labels)

# Dump e
#defaultEV

# print the whole job schedule
print theJob

EventSelector.InputCollections = [ “AOD.pool.root”]
```

Create an output file, and hook up the code that knows how to translate the objects to root ntuple leaves.

Dump Everything…

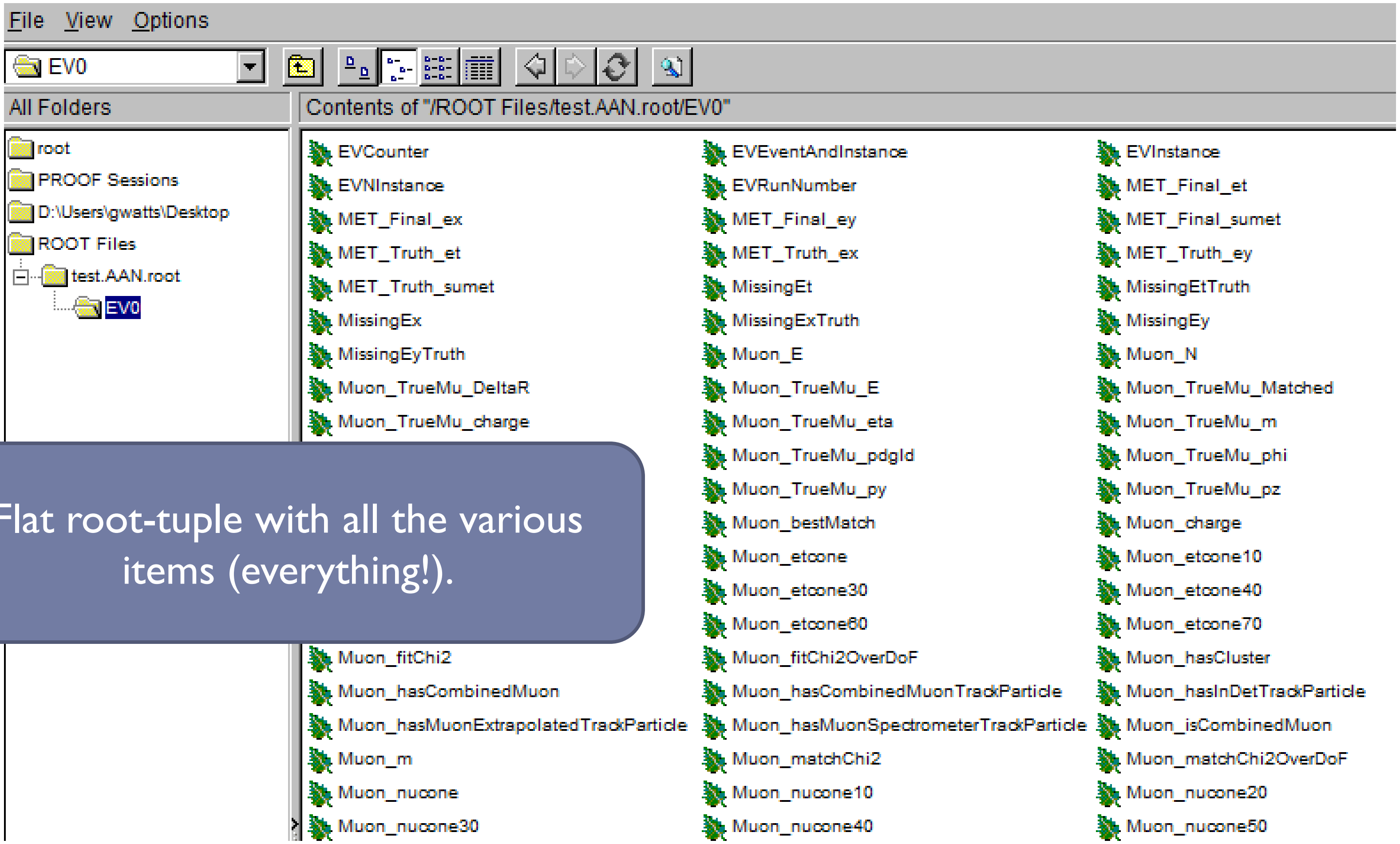
II

```
#schedule the module
defaultEVLooper += DefaultEventViewLabels("Labeler", defaultEVLooper.Inserters)

#Write UserData to AANtuple
defaultEVLooper+=AANtupleFromUserData("defaultAADumper",
filename="test.AAN.root",sequencer = theJob)

defaultEVLooper += UserDataDump("UserDataDump",
Labels=defaultEVLooper['Labeler']._Labels)

# Dump everything to scre
#defaultEVLooper+=EVScr                                    tUD=False)

# print the whole job schedule
print theJob

EventSelector.InputCollections = [ "AOD.pool.root"]
```

A KEY debugging tool. Just do it!

Usual File Specification…

What You Get…



Contents of "/ROOT Files/test.AAN.root/EV0"

EVCounter	EVEventAndInstance	EVInstance
EVNInstance	EVRunNumber	MET_Final_et
MET_Final_ex	MET_Final_ey	MET_Final_sumet
MET_Truth_et	MET_Truth_ex	MET_Truth_ey
MET_Truth_sumet	MissingEt	MissingEtTruth
MissingEx	MissingExTruth	MissingEy
MissingEyTruth	Muon_E	Muon_N
Muon_TrueMu_DeltaR	Muon_TrueMu_E	Muon_TrueMu_Matched
Muon_TrueMu_charge	Muon_TrueMu_eta	Muon_TrueMu_m
	Muon_TrueMu_pdgId	Muon_TrueMu_phi
	Muon_TrueMu_py	Muon_TrueMu_pz
	Muon_bestMatch	Muon_charge
	Muon_etcone	Muon_etcone10
	Muon_etcone30	Muon_etcone40
	Muon_etcone60	Muon_etcone70
Muon_fitChi2	Muon_fitChi2OverDoF	Muon_hasCluster
Muon_hasCombinedMuon	Muon_hasCombinedMuonTrackParticle	Muon_hasInDetTrackParticle
Muon_hasMuonExtrapolatedTrackParticle	Muon_hasMuonSpectrometerTrackParticle	Muon_isCombinedMuon
Muon_m	Muon_matchChi2	Muon_matchChi2OverDoF
Muon_nucone	Muon_nucone10	Muon_nucone20
Muon_nucone30	Muon_nucone40	Muon_nucone50

Flat root-tuple with all the various items (everything!).

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Add in a muon dumper
defaultEVLooper += anEVTool("EVUDFinalStateLooper/EVUDFSLMuon")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDKinCalc")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDMuonAll")
defaultEVLooper.EVUDFSLMuon.setProperties(
        Labels = ["Muon"],
        Prefix = "Muon_",
        SortParticles = True)
```

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Add in a muon dumper
defaultEVLooper += anEVTool("EVUDFinalStateLooper/EVUDFSLMuon")
                                                UDKinCalc")
                                                UDMuonAll")
```

A EV tool that loops over all final state objects in your event (muons, electrons, jets, etc…). Note that we are only going to be interested in muons, however! The source code object is EVUDFinalStateLooper, and we are giving it the name EVUDFSLMuon.

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Add in a muon dumper
defaultEVLooper += anEVTool("EVUDFinalStateLooper/EVUDFSLMuon")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDKinCalc")
                                                   "UDMuonAll")
```

An unamed tool that will calculate all the properties of the object (p_T, eta, etc.). It also correctly exposes its results so that they are written out to the root-tuple file.

Declarative!

I

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Add in a muon dumper
defaultEVLooper += anEVTool("EVUDFinalStateLooper/EVUDFSLMuon")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDKinCalc")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDMuonAll")
[illegible]
```

Like KinCalc – but it will process and expose properties that are specific to muons (quality, etc.). These can then also be written out to the root tuple.

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Add in a muon dumper
defaultEVLooper += anEVTool("EVUDFinalStateLooper/EVUDFSLMuon")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDKinCalc")
defaultEVLooper.EVUDFSLMuon += anEVTool("EVUDMuonAll")
defaultEVLooper.EVUDFSLMuon.setProperties(
        Labels = ["Muon"],
        Prefix = "Muon_",
        SortParticles = True)
```

Sets properties of the final state looper tool. First, we only want to loop over muons – all other objects are ignored. When we write them out, make sure to add a "Muon" to their name, and sort the output by p_T.

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Try to find a match with a muon...
defaultEVLooper.EVUDFSLMuon+=
anEVTool("EVUDToTruthParticleAssociator/TrueEAssoc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.Prefix= "TrueMu_"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc+=anEVTool( "EVUDKinCalc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.ContainerKey =  "SpclMC"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.deltaRmatch =  1
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.etCut =  0.0
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.usePdgID =  True
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.pdgID =  13
```

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Try to find a match with a muon...
defaultEVLooper.EVUDFSLMuon+=
anEVTool("EVUDToTruthParticleAssociator/TrueEAssoc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.Prefix= "TrueMu_"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc+=anEVTool( "EVUDKinCalc" )
erKey =  "SpclMC"
atch =  1
0.0
D =  True
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.pdgID =  13
```

Pre-written code to do MC to Data matching. We only need to configure it correctly.

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Try to find a match with a muon...
defaultEVLooper.EVUDFSLMuon+=
anEVTool("EVUDToTruthParticleAssociator/TrueEAssoc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.Prefix= "TrueMu_"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc+=anEVTool( "EVUDKinCalc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.ContainerKey =  "SpclMC"
                                                     match =  1
                                                         =  0.0
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.usePdgID =  True
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.pdgID =  13
```

Whatever this tool writes out to the ntuple, make sure to put the TrueMC in front of it.

Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Try to find a match with a muon...
defaultEVLooper.EVUDFSLMuon+=
anEVTool("EVUDToTruthParticleAssociator/TrueEAssoc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.Prefix= "TrueMu_"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc+=anEVTool( "EVUDKinCalc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.ContainerKey =  "SpclMC"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.deltaRmatch =  1
                                                   =  0.0
                                                  ID =  True
                                                   =  13
```

And write out the full kinematic variables for the matched MC. Note this is the same tool as what we saw for the actual muon above. This is a key feature of EV – everything looks *the same* and so you get broad reuse of the tools.

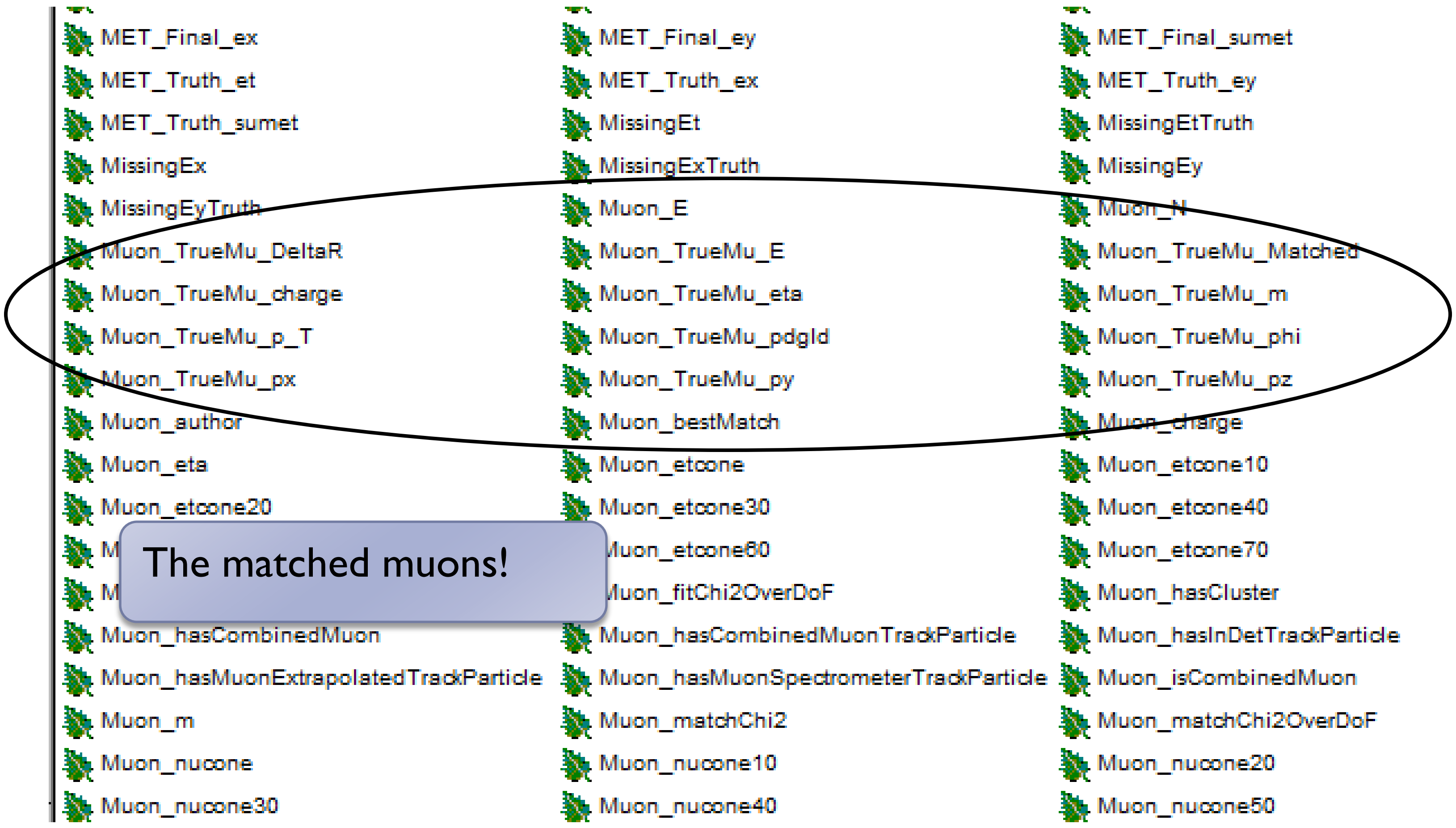
Declarative!

You can do lots more: calculate invariant masses.
Example: Associate a truth muon with a found muon

```
# Try to find a match with a muon...
defaultEVLooper.EVUDFSLMuon+=
anEVTool("EVUDToTruthParticleAssociator/TrueEAssoc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.Prefix= "TrueMu_"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc+=anEVTool( "EVUDKinCalc" )
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.ContainerKey =  "SpclMC"
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.deltaRmatch =  1
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.etCut =  0.0
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.usePdgID =  True
defaultEVLooper.EVUDFSLMuon.TrueEAssoc.pdgID =  13
```

Configure the associator. Match to a MC truth particle with pdg ID of |13| (muons), no lower energy cut, and $\Delta R<1$.

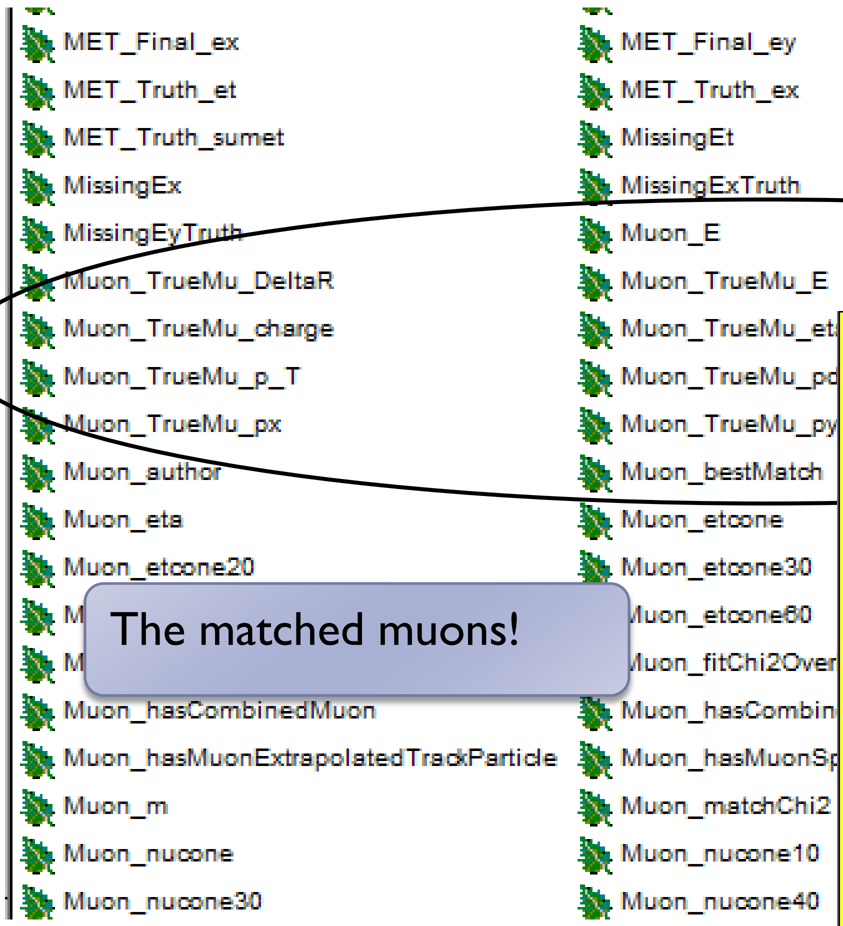
And the Association…



MET_Final_ex	MET_Final_ey	MET_Final_sumet
MET_Truth_et	MET_Truth_ex	MET_Truth_ey
MET_Truth_sumet	MissingEt	MissingEtTruth
MissingEx	MissingExTruth	MissingEy
MissingEyTruth	Muon_E	Muon_N
Muon_TrueMu_DeltaR	Muon_TrueMu_E	Muon_TrueMu_Matched
Muon_TrueMu_charge	Muon_TrueMu_eta	Muon_TrueMu_m
Muon_TrueMu_p_T	Muon_TrueMu_pdgId	Muon_TrueMu_phi
Muon_TrueMu_px	Muon_TrueMu_py	Muon_TrueMu_pz
Muon_author	Muon_bestMatch	Muon_charge
Muon_eta	Muon_etcone	Muon_etcone10
Muon_etcone20	Muon_etcone30	Muon_etcone40
M	uon_etcone60	Muon_etcone70
M	uon_fitChi2OverDoF	Muon_hasCluster
Muon_hasCombinedMuon	Muon_hasCombinedMuonTrackParticle	Muon_hasInDetTrackParticle
Muon_hasMuonExtrapolatedTrackParticle	Muon_hasMuonSpectrometerTrackParticle	Muon_isCombinedMuon
Muon_m	Muon_matchChi2	Muon_matchChi2OverDoF
Muon_nucone	Muon_nucone10	Muon_nucone20
Muon_nucone30	Muon_nucone40	Muon_nucone50

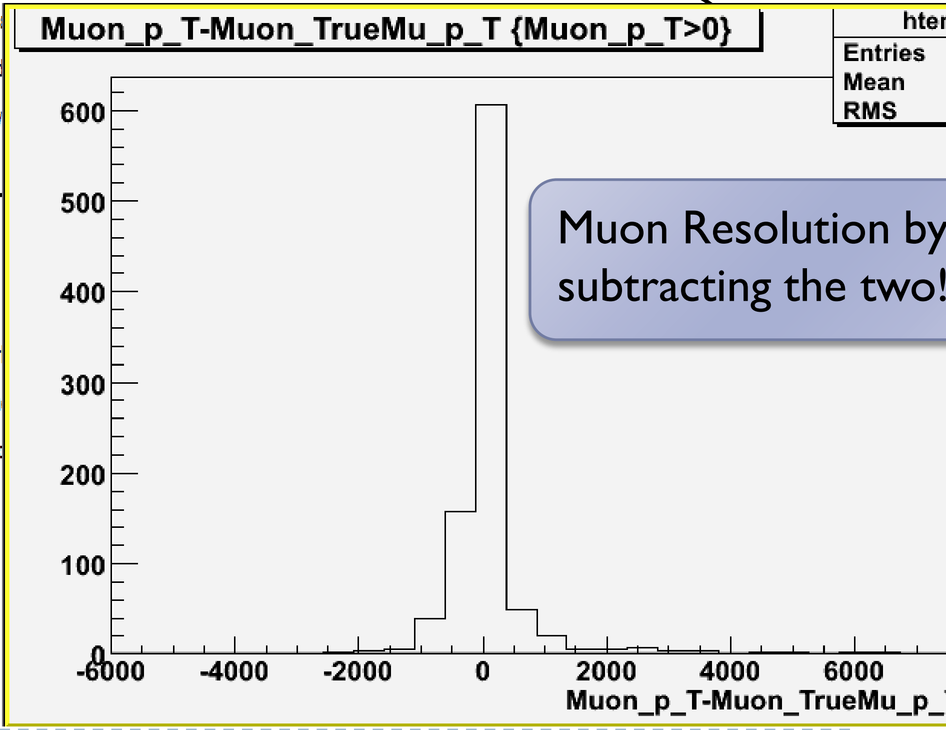
The matched muons!

And the Association…



MET_Final_ex	MET_Final_ey	MET_Final_sumet
MET_Truth_et	MET_Truth_ex	MET_Truth_ey
MET_Truth_sumet	MissingEt	MissingEtTruth
MissingEx	MissingExTruth	MissingEy
MissingEyTruth	Muon_E	Muon_N
Muon_TrueMu_DeltaR	Muon_TrueMu_E	Muon_TrueMu_Matched
Muon_TrueMu_charge	Muon_TrueMu_et	
Muon_TrueMu_p_T	Muon_TrueMu_pc	
Muon_TrueMu_px	Muon_TrueMu_py	
Muon_author	Muon_bestMatch	
Muon_eta	Muon_etcone	
Muon_etcone20	Muon_etcone30	
M	uon_etcone60	
M	uon_fitChi2Over	
Muon_hasCombinedMuon	Muon_hasCombin	
Muon_hasMuonExtrapolatedTrackParticle	Muon_hasMuonSp	
Muon_m	Muon_matchChi2	
Muon_nucone	Muon_nucone10	
Muon_nucone30	Muon_nucone40	

The matched muons!



Muon Resolution by subtracting the two!

What if what you want isn’t there?

- Adding new associators, or anything else.
- Not that hard (according to Orin)
  - And is part of the various tutorials.

Problems With The approach

- Flat Ntuple outputs
  - Get HUGE very quickly.
  - Should improve when pSAN and EV get combined (or whatever it is called this week)
    - A more object oriented format.
- Run-times.
  - EV Philosophy has you calculating everything in the ATHENA framework with minimal work afterwards
    - For example: hard to use ROOT to calculate final state particle matching
  - This works as long as it is fast to run ATHENA
    - What happens when we move to the GRID?
    - Long latencies each time you make a change?
    - Fastest way to kill off an analysis method.
  - This might be addressed when we can run ATHENA on pSAN output.
- Needs a DSL (Domain Specific Language)
  - Job options files are just not well suited to this – to much extra infrastructure has to be entered.
  - Similar to BaBar's fitting filter??

Problems with the Approach

- Output system seems to need a bunch of non-obvious connections (theJob, etc.).
  - I suspect they could clean this up.
- Any good when electron is not well understood?
  - When we first start up the detector will be quite dirty
  - EV's electrons (and other objects) are built with standard cuts.
    - 1. Write out all data with loose cuts on your electrons
    - 2. Determine best set of cuts.
    - 3. Program them into the job options file
    - 4. Re-run and do your analysis

More Information

- EventView Main Web Site
  - https://twiki.cern.ch/twiki/bin/view/Atlas/EventView
  - In particular look at the tutorial projects – very useful
- LXR Code Browser
- EventView HyperNews Site
  - https://hypernews.cern.ch/HyperNews/Atlas/get/eventview.html
  - Most Beginner Questions already answered.
- EventView User/Beginner's Manual
  - Useful the same way the ROOT user manual is useful for beginners.
  - Doesn't Exist yet
    - Should probably be written by a beginner, not one of the authors.

Conclusions

- Very Very easy way to get a root tuple out if you've never done it before.
- Not clear how this scales beyond something simple.
- Analysis is re-run frequently
  - Not obvious that this supports that well