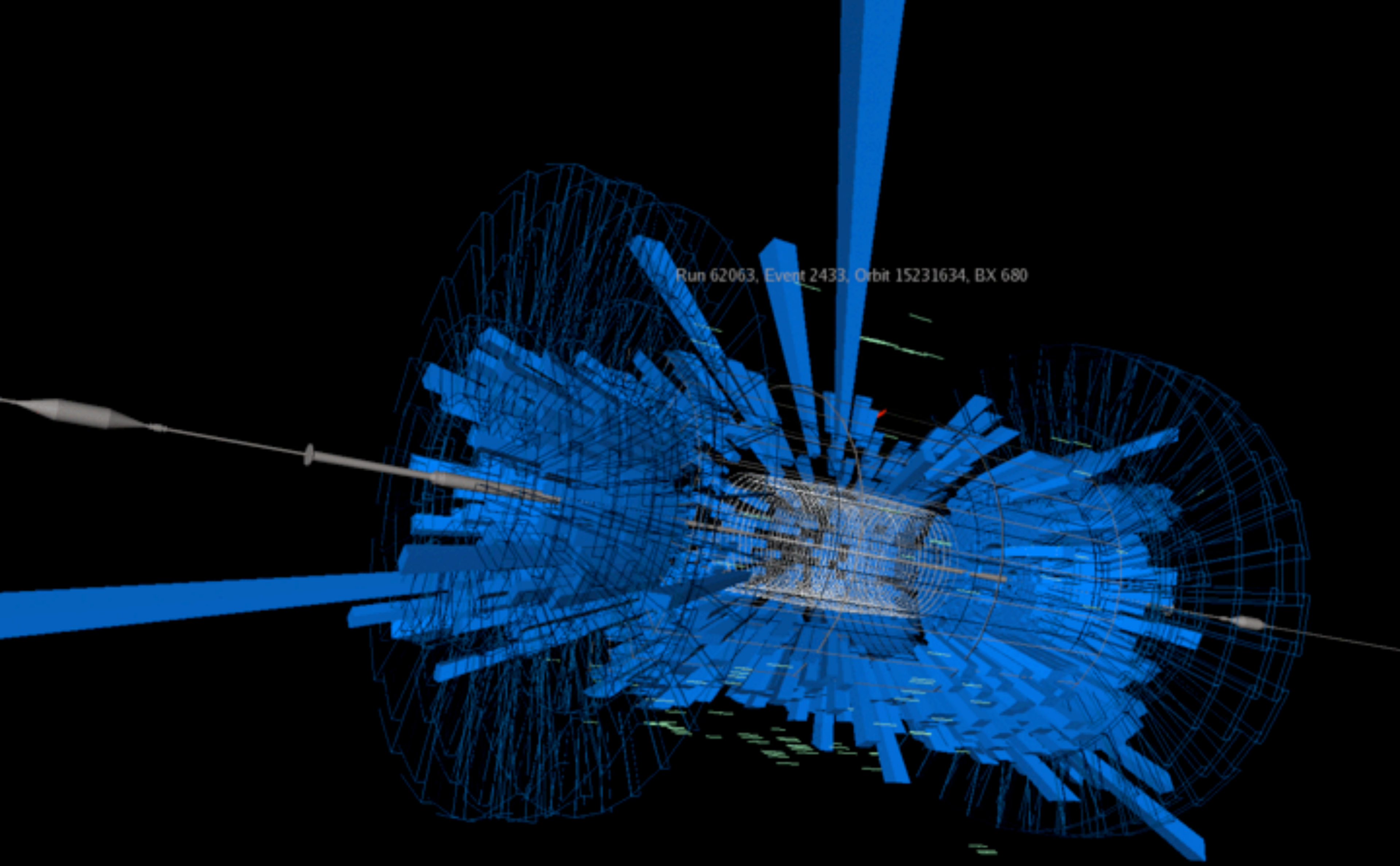


PAT Tutorial - Session V

Looking back at Exercise 4

- Data files can be used in FWLite, ROOT, cmsRun
- Basic structure of PAT objects
  - Looking at source code
  - Looking at doxygen
- Products created by PAT
  - Data types you will use for most of your analysis

- Isolation information
  - You should have a good starting point for your analysis (especially if the settings are proposed by POGs as you don’t need to defend them all)
- Embedding objects
  - Facilitating definition of event content
  - Reducing event size

Using Candidates

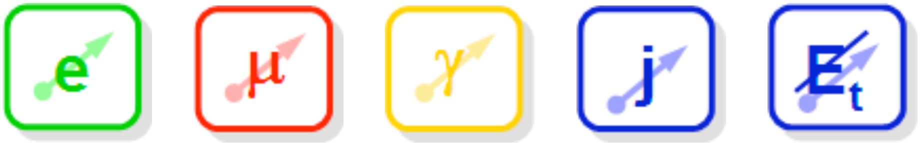
- Looking through doxygen and docs you probably stumbled across the terms “Candidate” and “Candidate Model”
- Candidates build an abstract layer to make the handling of physics objects easier
- All high level physics objects share the same interface
- Combinatorics and other tools are provided
- Most of the CMSSW parts use this model
- PAT is based on this model and your analysis code will too

candidate symbol

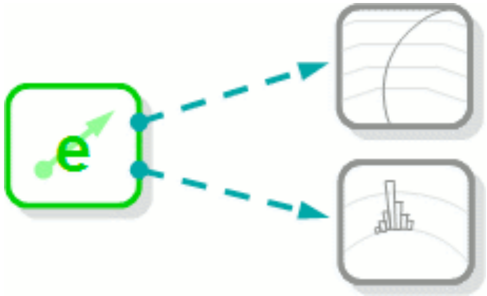
https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookParticleCandidates

Using Candidates

many types of objects are candidates



they can store references to their underlying components

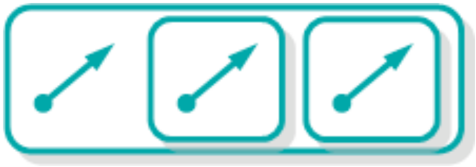


```
TrackRef trk = cand->get<TrackRef>();
```

Using Candidates

Candidates can be combined to new candidates (e.g. for Jets)

```
CompositeCandidate comp;
comp.addDaughter( dau1 );
comp.addDaughter( dau1 );
AddFourMomenta addP4;
addP4.set( comp );
```



Overlap checking

```
OverlapChecker overlap;
const Candidate & c1 = ..., & c2 = ...;

if ( overlap(c1,c2) ) { ... }
```

boost candidate plus daughters

```
Candidate * boostedZ = Z->clone();
CenterOfMassBooster boost;
boost.set( *boostedZ );
```

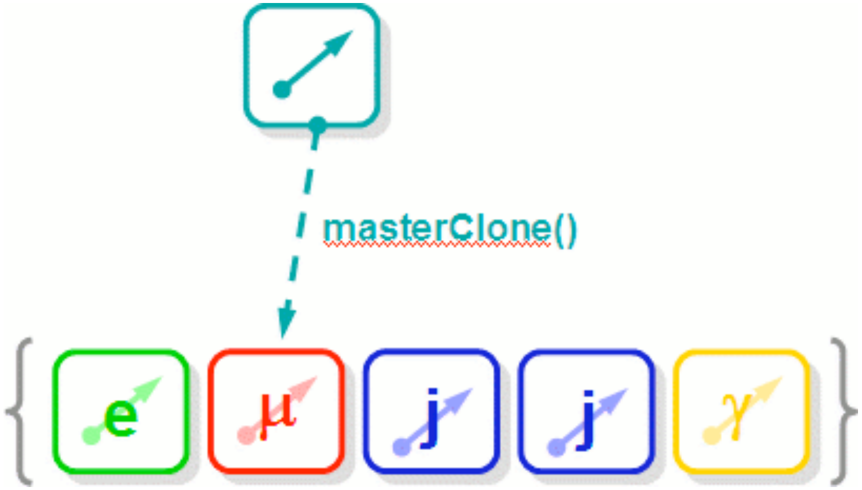
Shallow Clones

A shallow clone is a candidate forwarding access to the master clone. Kinematical variables can be changed.

`reco::ShallowCloneCandidate`

`reco::CandidateBaseRef`

`reco::CandidateCollection`



**use case:
combinatorial analysis with applied corrections**

PAT and RECO objects

- PAT objects don’t replace what is provided by reconstruction, but put some additional value on top.

```
template <class ObjectType>
class PATObject : public ObjectType {

  public:
    ...
    const std::vector<TriggerPrimitive> & triggerMatches() const;
    ...
    reco::GenParticleRef      genParticleRef(size_t idx=0) const;
    ...

};
```

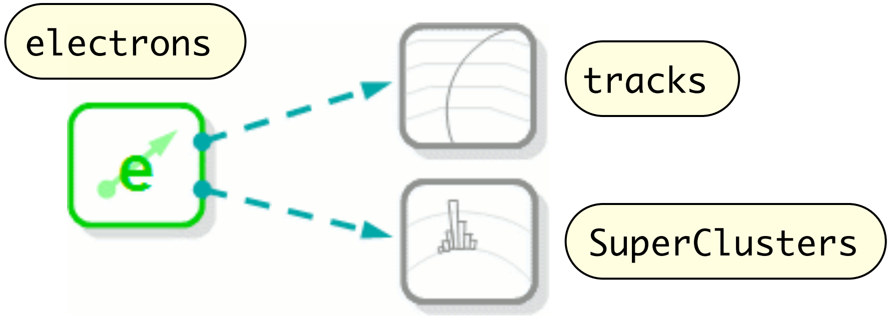
http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/DataFormats/PatCandidates/interface/PATObject.h

Embedding Objects

not embedded

=

data distributed over many products



electrons

tracks

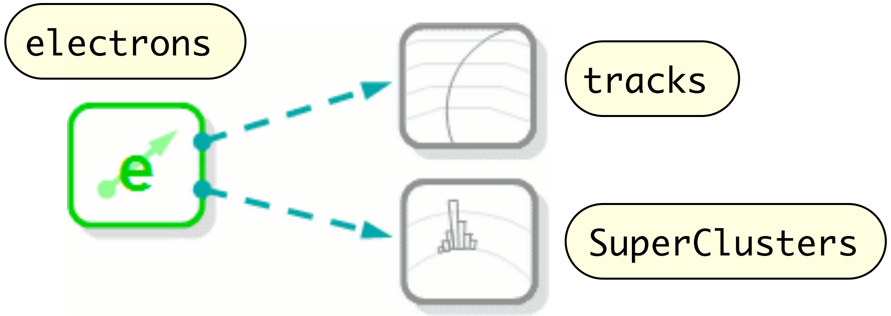
SuperClusters

Embedding Objects

not embedded

=

data distributed over many products



electrons

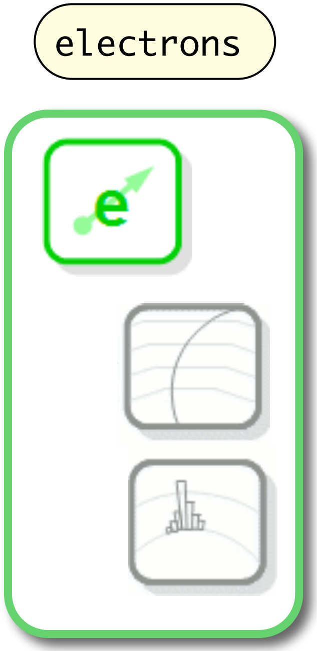
tracks

SuperClusters

embedded

=

data contained in a single product



electrons

Candidate Modules

- Many standard problems are already covered by existing candidate modules
- A few are quite general, others PAT specific
- You’ve seen already a few
- There are Selectors, Matchers, Combiner, Counters...

```
bestMuons = cms.EDProducer("CandSelector",
                           InputTag src = cms.InputTag(allMuons),
                           cut = cms.string("pt > 10 & abs( eta ) < 2")
                          )
```

- Your tutor will guide you picking the right ones

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

TFileService

Opening a ROOT file inside cmsRun is very error prone.
But there is an alternative...

```
// access the TFileService
edm::Service<TFileService> fs;

// create your histogram
TH1F * h_pt = fs->make<TH1F>( "pt"  , "p_{t}", 100,  0., 100. );

// fill it
h_pt->Fill( pt );

// create subdirectories if you like
TFileDirectory subDir = fs->mkdir( "mySubDirectory" );
```

```
# make the TFileService known to the config
process.TFileService = cms.Service("TFileService",
                                   fileName = cms.string("histo.root")
                                  )
```

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

HistoAnalyzer

Getting histograms without writing C++ code

```
plotJets = cms.EDAnalyzer("CandViewHistoAnalyzer",
    src = cms.InputTag("iterativeCone5CaloJets"),
    histograms = cms.VPSet(
      cms.PSet(
        itemsToPlot = cms.untracked.int32(5),  # plots the first 5 jets
        min = cms.untracked.double(0.0),
        max = cms.untracked.double(200),
        nbins = cms.untracked.int32(50),
        name = cms.untracked.string("jet %d E_{T} [GeV/c]"),
        description  = cms.untracked.string("jet_%d_et"),
        plotquantity = cms.untracked.string("et")
      )
    )
)
```

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

Documentation

What is described where?

- WorkBook
  - Entry point for the beginner
  - If you want to start with a topic, a part of the reconstruction, etc., do it here
  - Kati ensures a basic level of quality

- SWGuide
  - Much more detailed information
  - Very heterogenous
  - Highly fluctuating

- Doxygen
  - Describes the code and class interfaces
  - A good reference for the programmer
  - It explains the *how*, but not the *why* of a design

Homework

- Please complete your logbook so that we know if there are any problems you face
- Write an EDAnalyzer that plots the quantities you are interested in for your selection
- Sketch if there are any use cases for existing candidate modules in your analysis.
- Continue giving feedback! :-)

And now there is time for your questions!