Getting Started with PAT

Benedikt Hegner

PAT Tutorial Week, March 2010

Outline

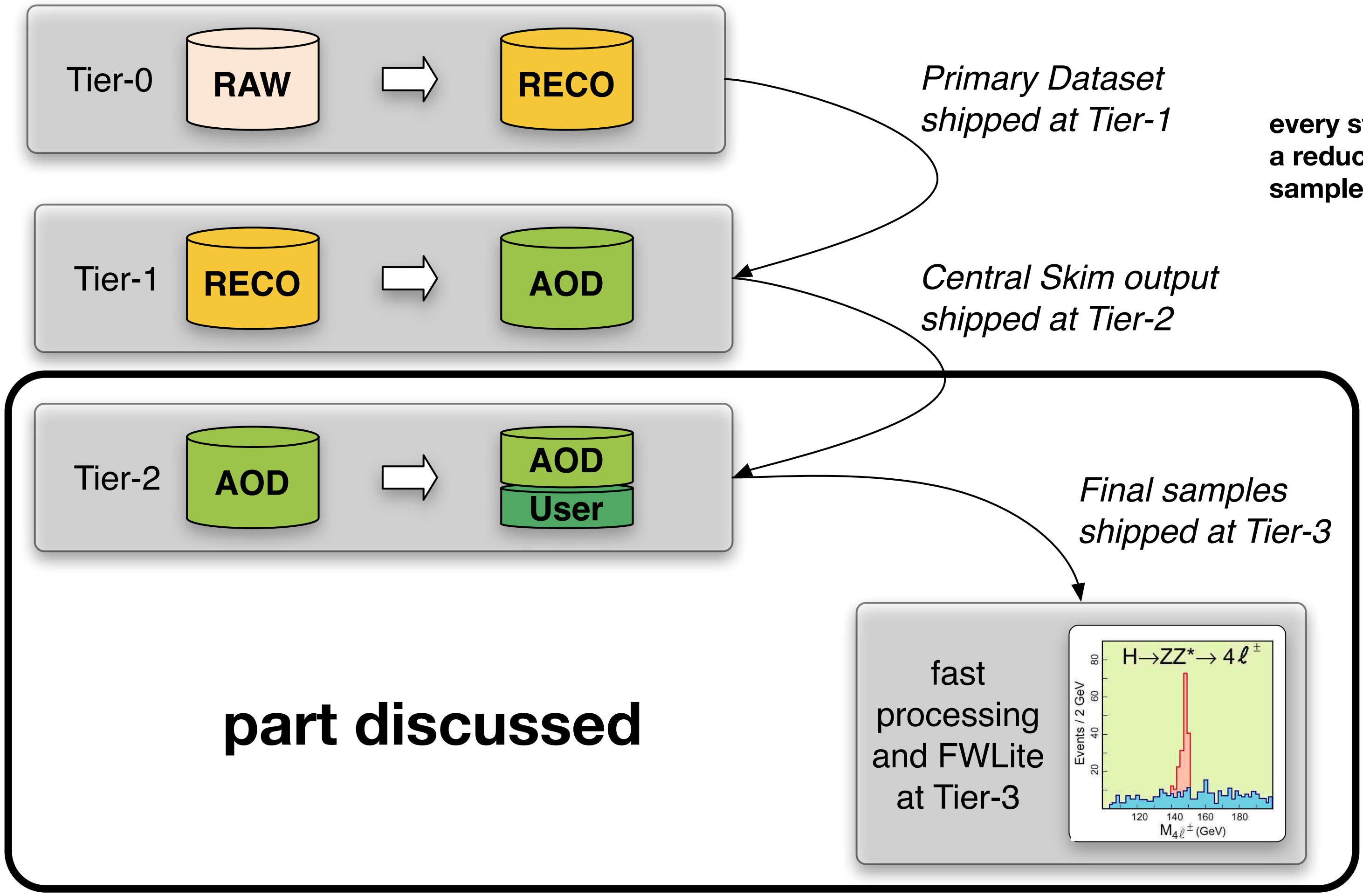
- Analysis Model in CMS
- What is PAT?
- PAT data formats
- PAT workflow
- PAT content

- How to use PAT
  - How to run it
  - How to use the output

- Support and Documentation

Introduction - Analysis Flow

Simplified picture


Tier-0
RAW
RECO
Primary Dataset shipped at Tier-1
Tier-1
RECO
AOD
Central Skim output shipped at Tier-2
Tier-2
AOD
AOD
User
Final samples shipped at Tier-3
fast processing and FWLite at Tier-3
H→ZZ*→ 4ℓ±
Events / 2 GeV
M4ℓ± (GeV)
part discussed

every step means a reduction in sample size

What is the use case?

- Groups or individuals run on AOD/RECO on Tier-2 using cmsRun to
  - perform higher level reconstruction, access calibrations
  - preselect objects and events
- Output is saved on some user space on Tier-2/3
- Above step possibly repeated several times
- In the last step copy data to your favorite machine and perform interactive analysis on it as you are used to

- The PAT is designed with this use case in mind

What is PAT?

- The PAT is a ***toolkit*** as part of this CMSSW framework
- It serves as well tested and supported common ground for group and user analysis
- It facilitates reproducibility and comprehensibility of analyses
- A ***common language***

- If another person describes you a PAT analysis you can easily know what he/she is talking about.

Concepts I

Interface

- b/w reconstruction and analysis
- simplifies access via `DataFormats`

Common Format

- facilitates transfer and comparisons
- PAG common configurations
- sustained provenance

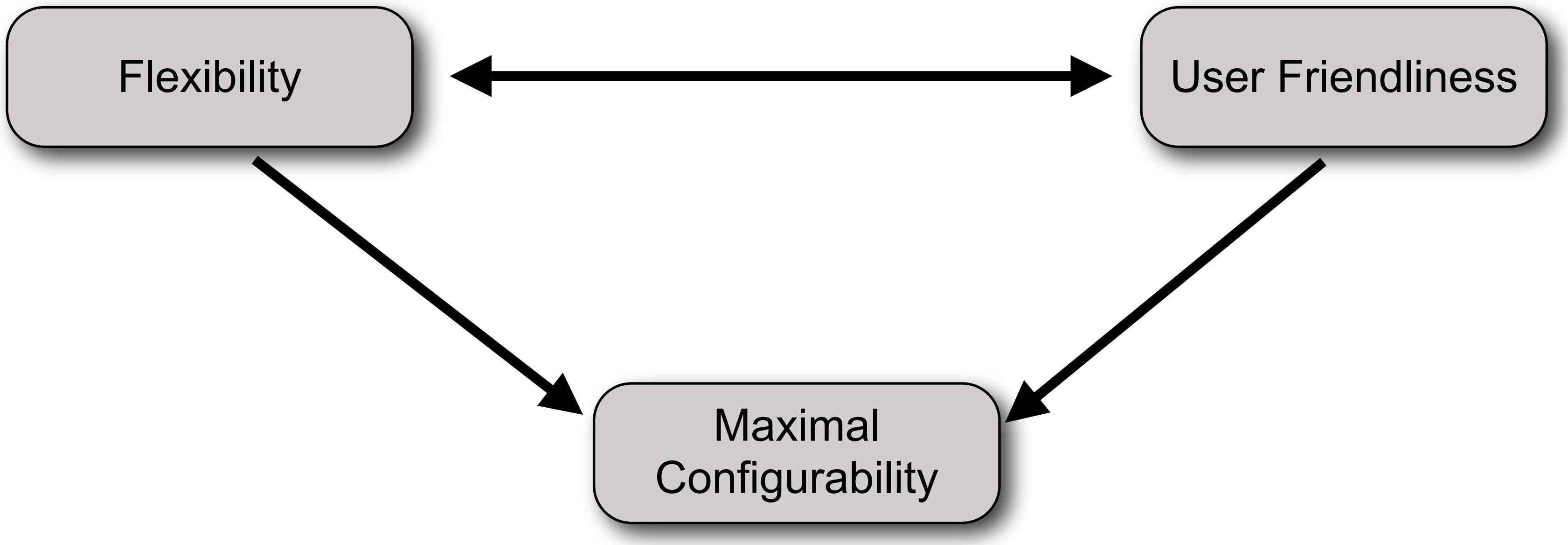
Common Tool

- approved algorithms and ~sensible defaults
- synergy (everybody can profit from recent developments)
- quick start into analysis for beginners

It canalizes expertise as it is the technical crossing point between all PAGs and POGs and is a natural extension of the data model

Concepts II

- Make use of the modular structure of CMSSW
- Provide easy access via member functions in `DataFormats`
- Serve 80% of all analyses in CMS

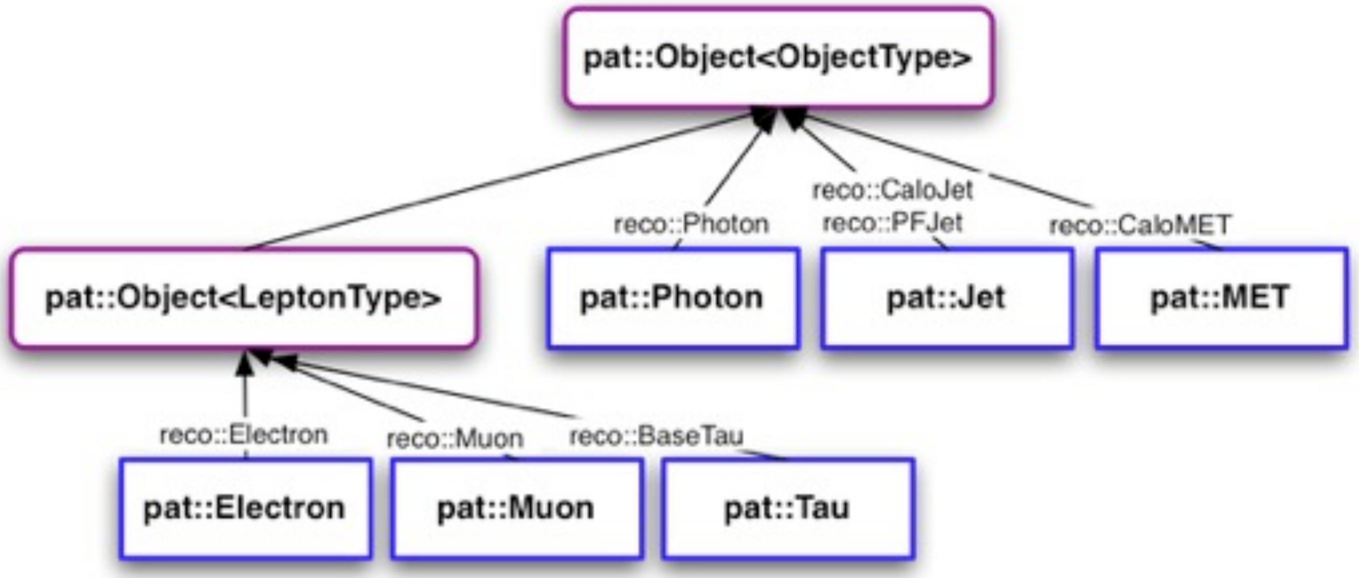


PAT DataFormats

- PAT extends reco::Candidates so that they store information useful for analysis:

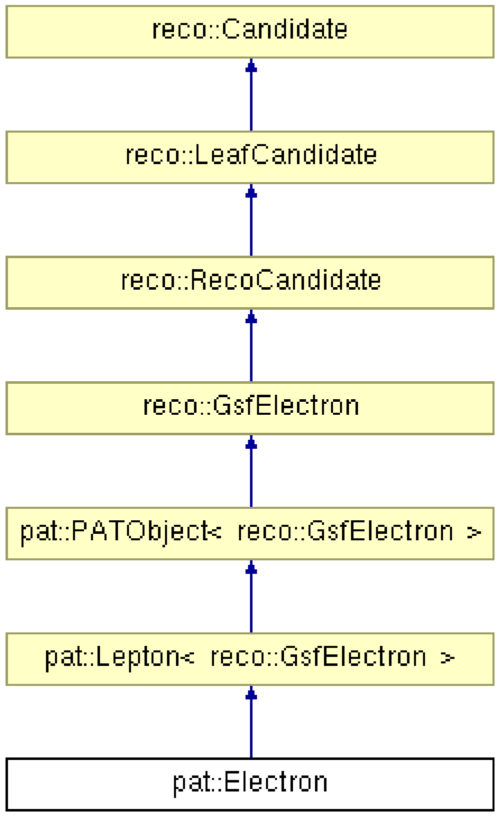
  PAT object = RECO object + more

- Code of these data types can be found in
  `DataFormats/PatCandidates`



Example - pat::Electron

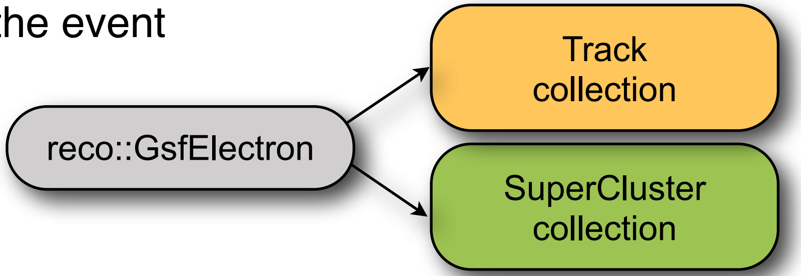
- pat::Electron inherits all properties of reco::GsfElectron
- It also inherits some properties from PATObject:



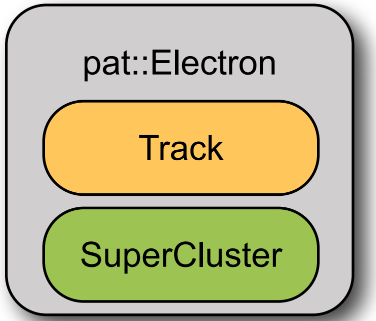
- Information about efficiency is accessible via efficiency()
- The best-matched MC electron is accessible via genParticle()
- Resolution on energy, pt, position,... are accessible via resolE(), resolPt(), ...
- Information about matched trigger objects via triggerObjectMatches()
- In addition to the isolation variables from reco::GsfElectron, one can add and access userIso
- Identification variables like electronID()

PAT DataFormats - embedding

- By default, reco objects keep only a reference to their main constituents.
- For example `reco::GsfElectron` stores only a reference to its track and SuperCluster.
  - They are not independent from other parts of the event
  - Their sizes are optimized



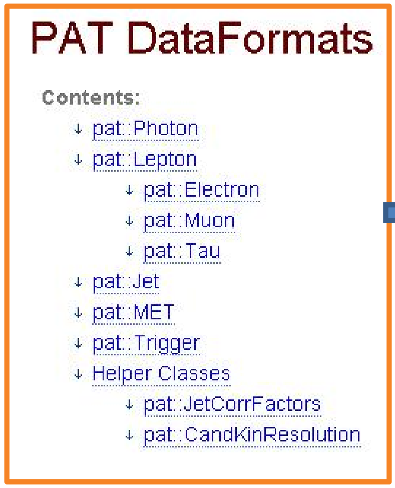
- In PAT objects, information can be kept as reference or as alternative embedded into the object
  - Being independent from other parts of the event
  - Makes EventContents more flexible



- More details tomorrow

PAT DataFormats - documentation

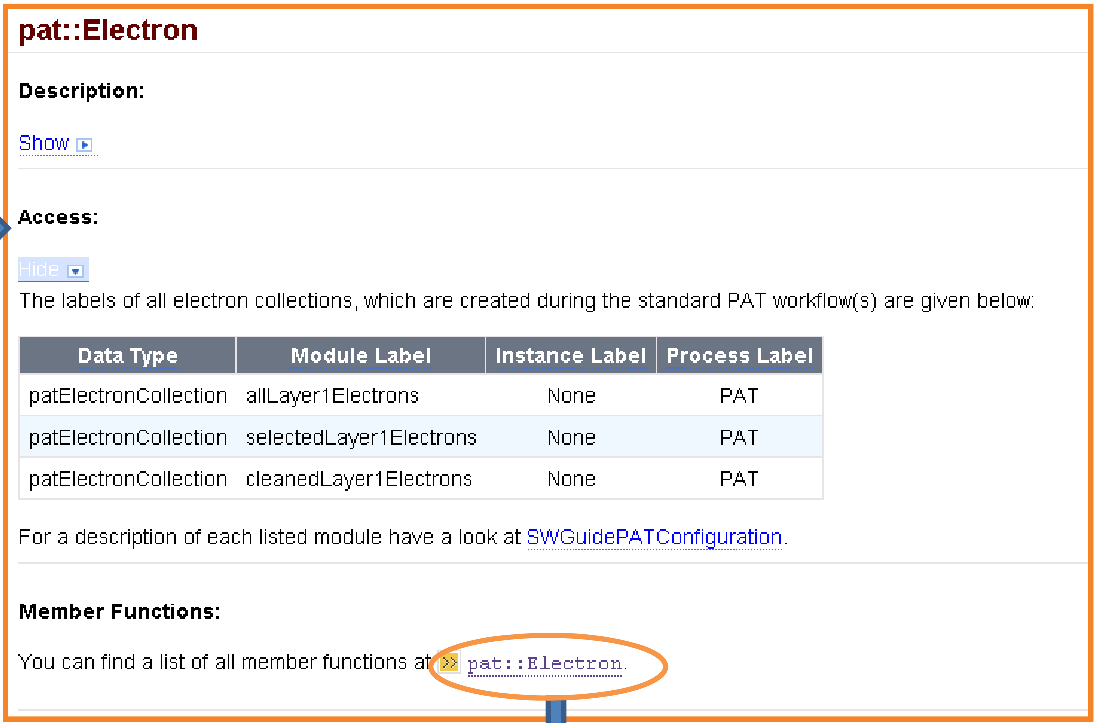
- Further documentation on **SWGuidePATDataFormats**



PAT DataFormats

Contents:

- pat::Photon
- pat::Lepton
  - pat::Electron
  - pat::Muon
  - pat::Tau
- pat::Jet
- pat::MET
- pat::Trigger
- Helper Classes
  - pat::JetCorrFactors
  - pat::CandKinResolution



pat::Electron

Description:

Show

Access:

Hide

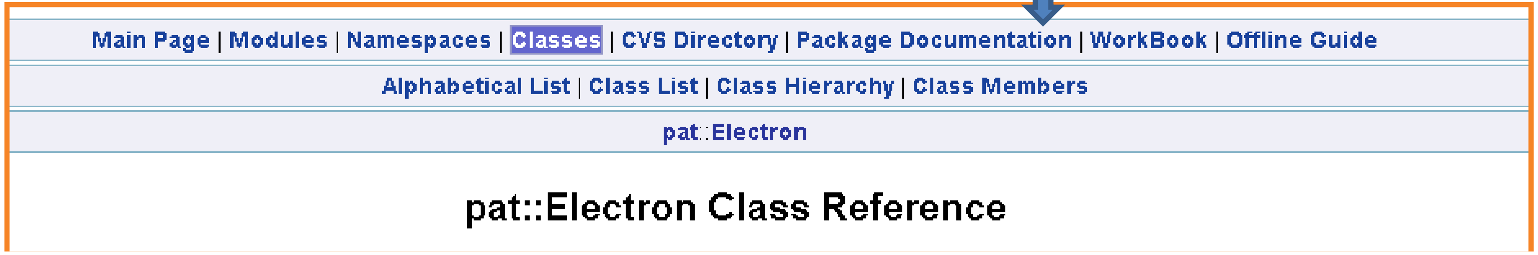
The labels of all electron collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Module Label	Instance Label	Process Label
patElectronCollection	allLayer1Electrons	None	PAT
patElectronCollection	selectedLayer1Electrons	None	PAT
patElectronCollection	cleanedLayer1Electrons	None	PAT

For a description of each listed module have a look at SWGuidePATConfiguration.

Member Functions:

You can find a list of all member functions at pat::Electron.



Main Page | Modules | Namespaces | Classes | CVS Directory | Package Documentation | WorkBook | Offline Guide

Alphabetical List | Class List | Class Hierarchy | Class Members

pat::Electron

pat::Electron Class Reference

pat[Candidates]

- The first step in PAT is making pat candidates from each reco candidate
- In this step, PAT imports all of the standard modules by their default values from different POGs, runs all of them and combines the associated results with each object to make PAT objects.
- MC matching is part of this step as well

To see what the configs look like, have a look at these directories

- MC matching
  PhysicsTools/PatAlgos/python/mcMatchLayer0/
- Sequences from POGs
  PhysicsTools/PatAlgos/python/recoLayer0/
- Final commands to create allLayer1 (one module each type)
  PhysicsTools/PatAlgos/python/producersLayer1/

selectedPat[Candidates]

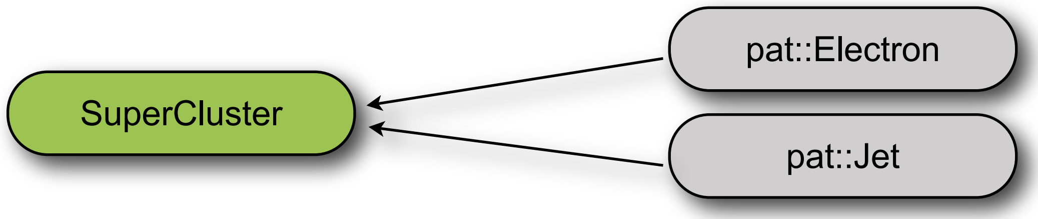
- Usually not all of the PAT objects are needed for your analysis and a simple cut can decrease the number and save more space
- Results are stored in selectedLayer1Candidates
- Default cuts can be read in **SWGuidePATConfiguration** or the config files
  PhysicsTools/PatAlgos/python/selectionLayer1
- Using the cut parser makes them very readable and easy to change

```
selectedPatElectrons = cms.EDFilter(“PATElectronSelector”,
    src = cms.InputTag(“allPatElectrons”),
    cut = cms.string(“pt > 0. & abs(eta) < 12.”)
)
```

- Further documentation on the parser in the twiki:
  SWGuidePhysicsCutParser

cleanPat[Candidates]

- After selecting reco objects, each collection can be cleaned utilizing other collections

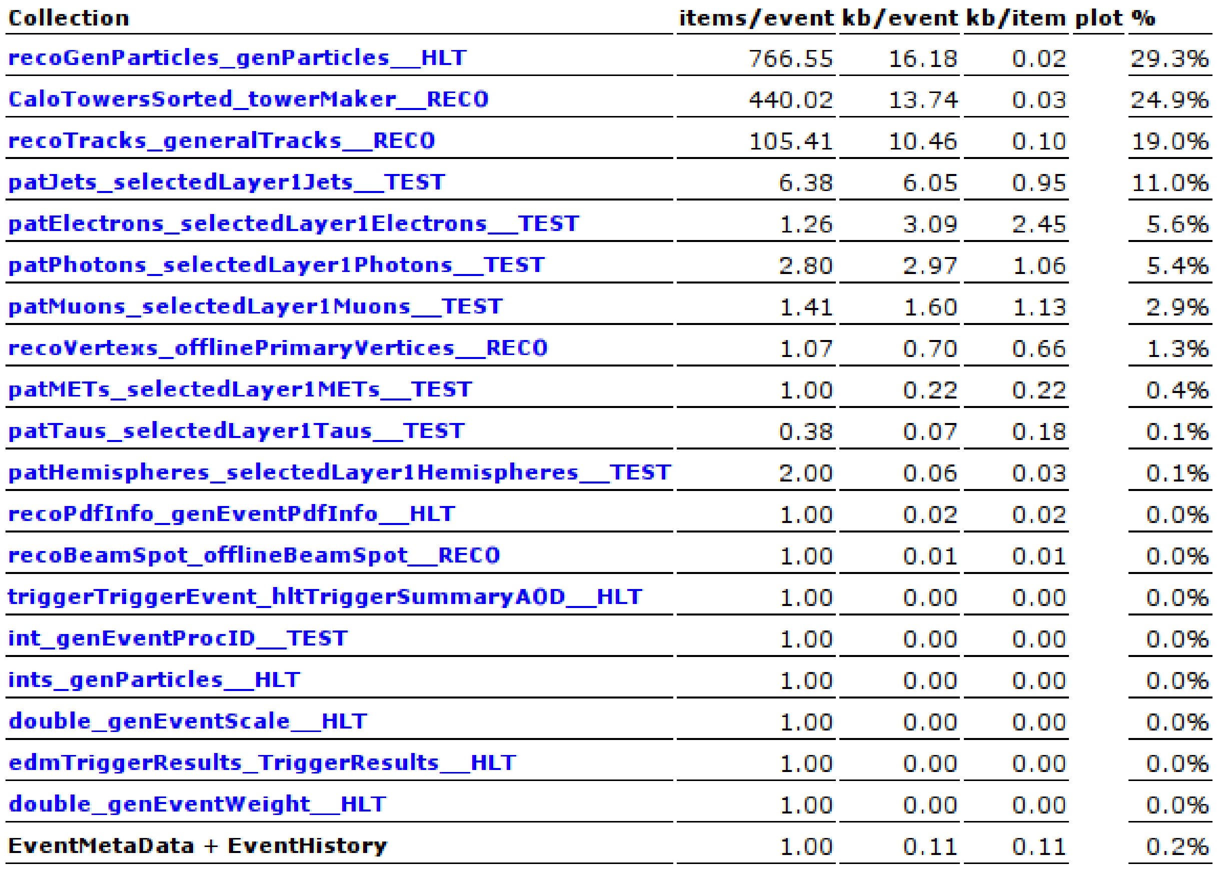


- For example, a jet collection should be cleaned from electrons
- A set of PATCandidateCleaners can help to resolve this double counting
- This cross cleaning is the last step of the PAT workflow
- Results are stored in `cleanPat[Candidates]`
- The config files for the cleaning can be found at
  `PhysicsTools/PatAlgos/python/cleaningLayer1`

EventContent

- Which part of the produced data to store is completely configurable
- Default is `patEventContent` which stores cleaned objects
- Typical untuned sizes/event are around 30-40kB
- Some groups or individuals however use sizes more in the region of ~5kB/event
- There are tools to help with reducing event sizes
- More details on

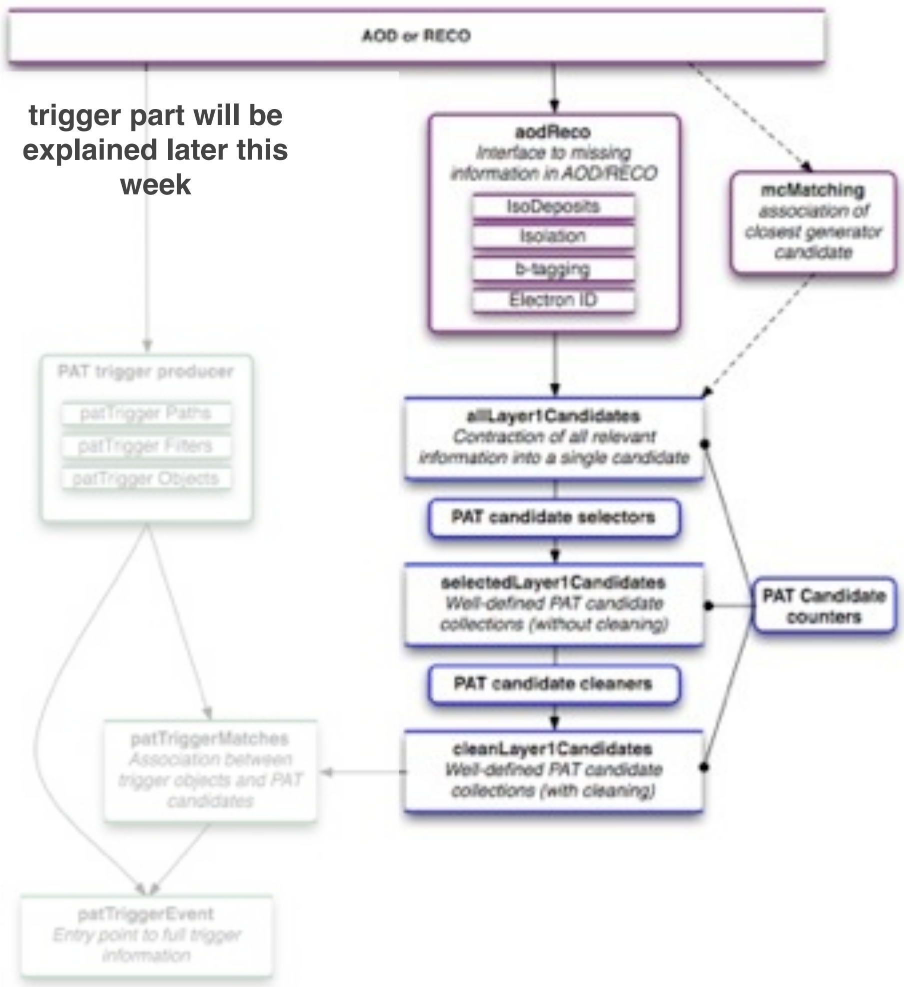
  SWGuidePATConfiguration
  SWGuidePATEventSize



Collection	items/event	kb/event	kb/item	plot	%
recoGenParticles_genParticles__HLT	766.55	16.18	0.02		29.3%
CaloTowersSorted_towerMaker__RECO	440.02	13.74	0.03		24.9%
recoTracks_generalTracks__RECO	105.41	10.46	0.10		19.0%
patJets_selectedLayer1Jets__TEST	6.38	6.05	0.95		11.0%
patElectrons_selectedLayer1Electrons__TEST	1.26	3.09	2.45		5.6%
patPhotons_selectedLayer1Photons__TEST	2.80	2.97	1.06		5.4%
patMuons_selectedLayer1Muons__TEST	1.41	1.60	1.13		2.9%
recoVertexs_offlinePrimaryVertices__RECO	1.07	0.70	0.66		1.3%
patMETs_selectedLayer1METs__TEST	1.00	0.22	0.22		0.4%
patTaus_selectedLayer1Taus__TEST	0.38	0.07	0.18		0.1%
patHemispheres_selectedLayer1Hemispheres__TEST	2.00	0.06	0.03		0.1%
recoPdfInfo_genEventPdfInfo__HLT	1.00	0.02	0.02		0.0%
recoBeamSpot_offlineBeamSpot__RECO	1.00	0.01	0.01		0.0%
triggerTriggerEvent_hltTriggerSummaryAOD__HLT	1.00	0.00	0.00		0.0%
int_genEventProcID__TEST	1.00	0.00	0.00		0.0%
ints_genParticles__HLT	1.00	0.00	0.00		0.0%
double_genEventScale__HLT	1.00	0.00	0.00		0.0%
edmTriggerResults_TriggerResults__HLT	1.00	0.00	0.00		0.0%
double_genEventWeight__HLT	1.00	0.00	0.00		0.0%
EventMetaData + EventHistory	1.00	0.11	0.11		0.2%

Combining all previous steps

- Have a look at **SWGuidePATWorkflow**



AOD or RECO
trigger part will be explained later this week
aodReco
Interface to missing information in AOD/RECO
IsoDeposits
Isolation
b-tagging
Electron ID
mcMatching
association of closest generator candidate
PAT trigger producer
patTrigger Paths
patTrigger Filters
patTrigger Objects
allLayer1Candidates
Contraction of all relevant information into a single candidate
PAT candidate selectors
selectedLayer1Candidates
Well-defined PAT candidate collections (without cleaning)
PAT Candidate counters
PAT candidate cleaners
cleanLayer1Candidates
Well-defined PAT candidate collections (with cleaning)
patTriggerMatches
Association between trigger objects and PAT candidates
patTriggerEvent
Entry point to full trigger information

POG pre-production steps

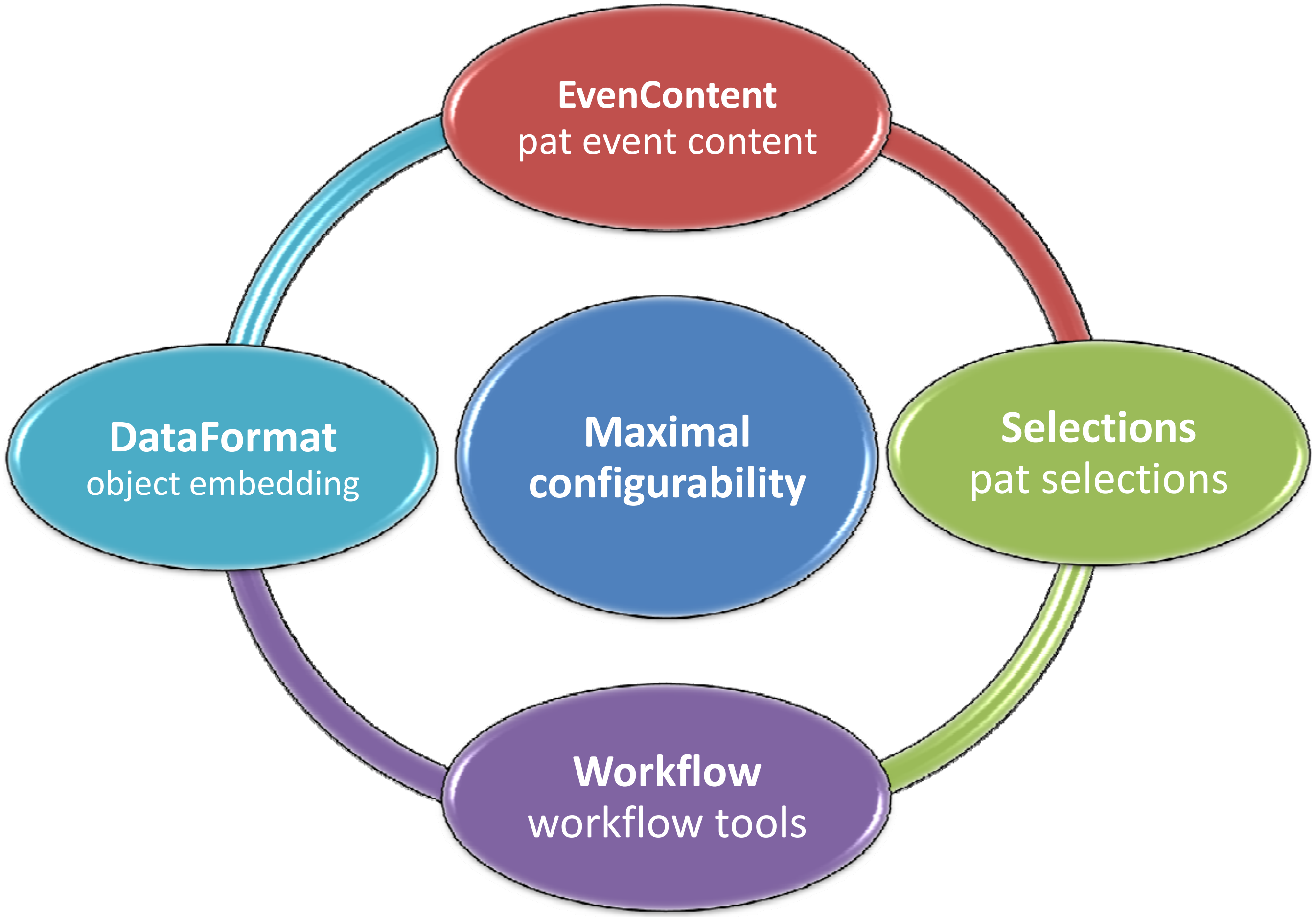
Basic collections

Selected collection

Cleaned collection
(the only one stored per default)

Maximal Configurability

- Sustain flexibility and user friendliness by maximized configurability without writing C++ code



How to get started

- Enough theory. How to get PAT started?
- It is part of the release. Unfortunately the release is broken for features outside PAT. So for this tutorial we will use a special version:

```
cmsrel CMSSW_3_5_X_2010-03-08-0200
cd CMSSW_3_5_X_2010-03-08-0200/src
cmsenv
```

- For each version there are release notes (and sometimes hot fixes) on **SWGuidePATRecipes**
- There are example configurations in PhysicsTools/PatAlgos/test
- To get them, do

```
addpkg PhysicsTools/PatAlgos
cd PhysicsTools/PatAlgos/test
```

One example

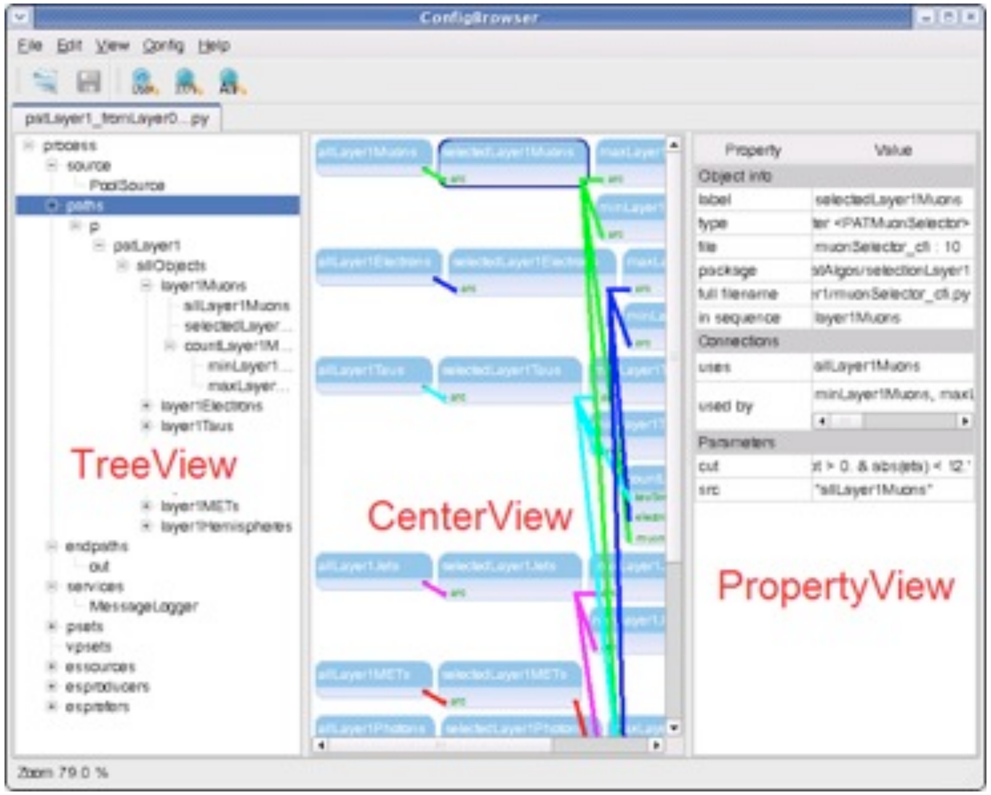
- The simples config file test_cfg.py you can have is

```
from PhysicsTools.PatAlgos.patTemplate_cfg. import *

process.source.fileNames = [“file:INPUT.root”]
process.maxEvents.input = 100

process.out.fileName = “PAT.root”

process.load(“PhysicsTools.PatAlgos.patSequences_cff”)

process.p = cms.Path( process.patDefaultSequence )
```

- In this simple file, you can set the input file and the number of events to be translated into PAT format and also the name of the output
- Default settings of PAT can now be changed by adding new lines
- Given you ran this file with cmsRun, you can use the created PAT.root as input for the next examples

How to change PAT defaults

- There are various ways to learn about parameters in a configuration. Here the three most important ones.
- Using the edmConfigBrowser



Property	Value
Object info	
label	selectedLayer1Muons
type	ter <PATMuonSelector>
file	muonSelector_cfi : 10
package	atAlgos/selectionLayer1
full filename	r1/muonSelector_cfi.py
in sequence	layer1Muons
Connections	
uses	allLayer1Muons
used by	minLayer1Muons, maxL
Parameters	
cut	t > 0. & abs(eta) < 12.'
src	"allLayer1Muons"

- More in the presentation by A. Hinzmann

How to change PAT defaults

- Another way is using the Python interpreter directly

```
$ python -i test_cfg.py
>>> process.selectedPatElectrons
cms.EDFilter(“PATElectronSelector”,
    src = cms.InputTag(“allPatElectrons”),
    cut = cms.string(“pt > 0. & abs(eta) < 12.”)
)

>> process.selectedPatMuons
cms.EDFilter(“PATMuonSelector”,
    src = cms.InputTag(“allPatMuons”),
    cut = cms.string(“pt > 0. & abs(eta) < 12.”)
)

>>>
```

the file from previous slide

full name of the module

output of the tool -
the module configuration

Ctrl-d to exit

- To change the settings, edit the config file :

```
process.selectedPatElectrons.cut = “pt > 10”
```

Provenance

- The third way of learning about configs works after the fact
- The history of how an event was created is called *provenance*
- It consists of two parts
  - where does my data come from?
  - what was done to the data?
- The origin of data sets can be traced using the data bookkeeping system (DBS)
- What was done in each job with these is stored in the file provenance
  - used CMSSW version
  - used parameters

- This per file information can help you to find out which PAT configurations were used to create a file. The tool to look at it is called `edmProvDump`

Reminder - data model

- Up to now we were discussing what to do to create a PAT file
- Most of your time though you will be busy with reading (analyzing) such files
- Let’s move to that now!

- In CMS there are two ways of reading and analyzing data
  - full framework (cmsRun)
  - FWLite

- The full framework gives you full access to low level reconstruction, alignment and so on and allows to store new data into the edm file
- FWLite on the other hand is an application which just reads already created objects and gives you a much *faster* and *easier* way of doing analysis with *the same objects as in the full framework*

Using FWLite with PAT objects

- A full blown example for FWLite can be found at
  WorkbookPATExampleFWLite
- A summary of the relevant parts:

```
// load framework libraries
gSystem->Load("libFWCoreFWLite");
AutoLibraryLoad::enable();

// loop the events
unsigned int iEvent = 0;
fwlite::Event ev(inFile);
for(ev.toBegin(); !ev.atEnd(); ++ev, ++iEvent){
    edm::EventBase const & event = ev;
    // break loop after end of file is reached
    // or after 1000 events have been processed
    if( iEvent==1000 ) break;

    // simple event counter
    if(iEvent>0 && iEvent%1==0){
      std::cout << "  processing event: " << iEvent << std::endl;
    }

    // handle to to muon collection
    edm::Handle<std::vector<pat::Muon> > muons;
    edm::InputTag muonLabel("cleanPatMuons");
    event.getByLabel(event, "cleanPatMuons");

    // loop muon collection and fill histograms
    for(unsigned i=0; i<muons->size(); ++i){
      muonPt_ ->Fill( (*muons)[i].pt()  );
    }
  }

  inFile->Close();
```

Never forget to enable the AutoLibraryLoader

Loop the events of an input file

Receive a muon collection by label

Using cmsRun with PAT objects

- A full blown example for cmsRun can be found at
  WorkBookPATExampleBasic
- Again a short summary here
- Every time you want to access a PAT data format you need to include the relevant header file, to your analyzer e.g.

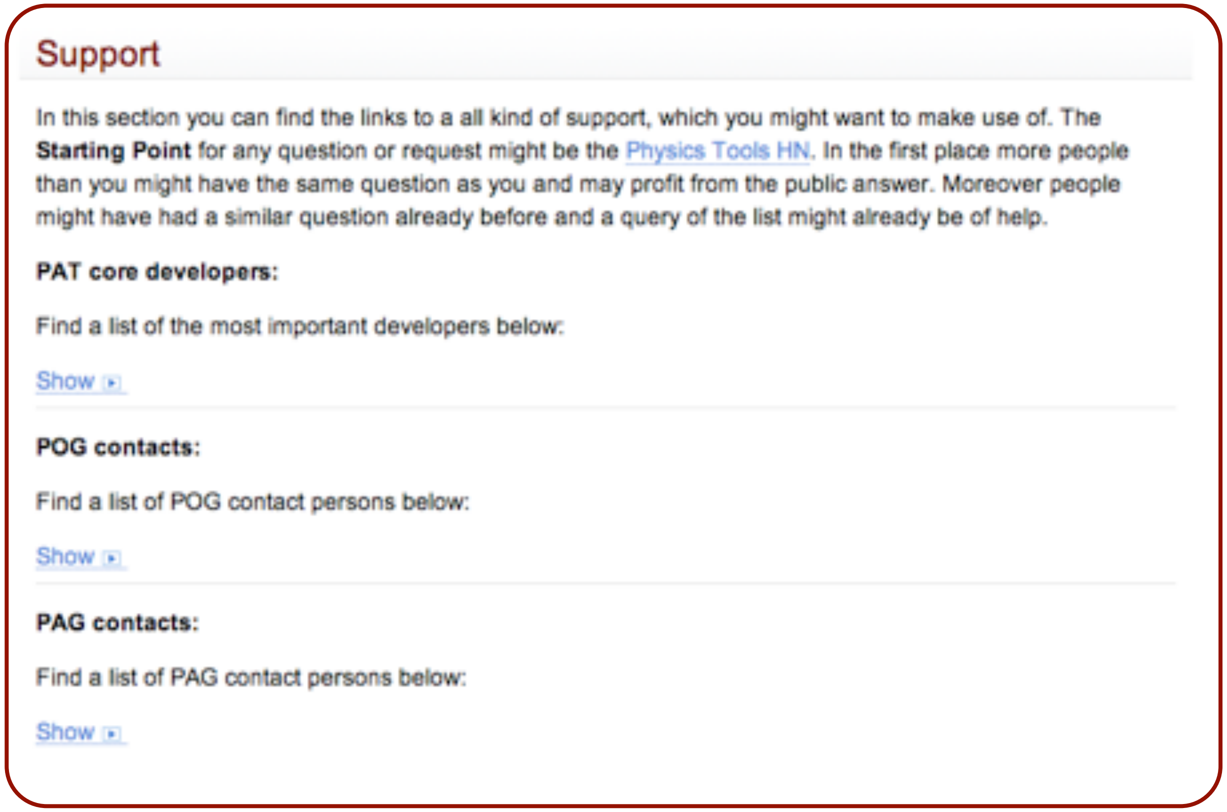
```
#inlucde “DataFormats/PatCandidates/interface/Muon.h”
```

- Make sure your BuildFile contains <use name=DataFormats/PatCandidates>
- Once this is done, you are free to add your analysis code to the analyze method of your module

```
void
PatBasicAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
{
  // get muon collection
  edm::Handle<edm::View<pat::Muon> > muons;
  iEvent.getByLabel(“cleanPatMuons”,muons);

  // loop over muons
  size_t acceptedMuons=0;
  for(edm::View<pat::Muon>::const_iterator muon=muons->begin(); muon!=muons->end(); ++muon){
    if(muon->pt()>50)
      ++acceptedMuons;
  }

}
```

Support

- For information on support have a look at **SWGuidePAT**



Support

In this section you can find the links to a all kind of support, which you might want to make use of. The **Starting Point** for any question or request might be the Physics Tools HN. In the first place more people than you might have the same question as you and may profit from the public answer. Moreover people might have had a similar question already before and a query of the list might already be of help.

PAT core developers:

Find a list of the most important developers below:

Show

POG contacts:

Find a list of POG contact persons below:

Show

PAG contacts:

Find a list of PAG contact persons below:

Show

- Tutorials
- Hypernews
- Community
- POG/PAG contacts
- Developers

Documentation

- **SWGuidePAT** Main documentation page
- **SWGuidePATRecipes** Information about releases
- **SWGuidePATExamples** Tutorials and examples
- **SWGuidePATDataFormats** pat::Candidate description
- **SWGuidePATConfiguration** Module configuration
- **SWGuidePATEventSize** Tools for event size estimate
- **SWGuidePATWorkflow** PAT workflow description
- **SWGuidePATTools** Description of workflow tools