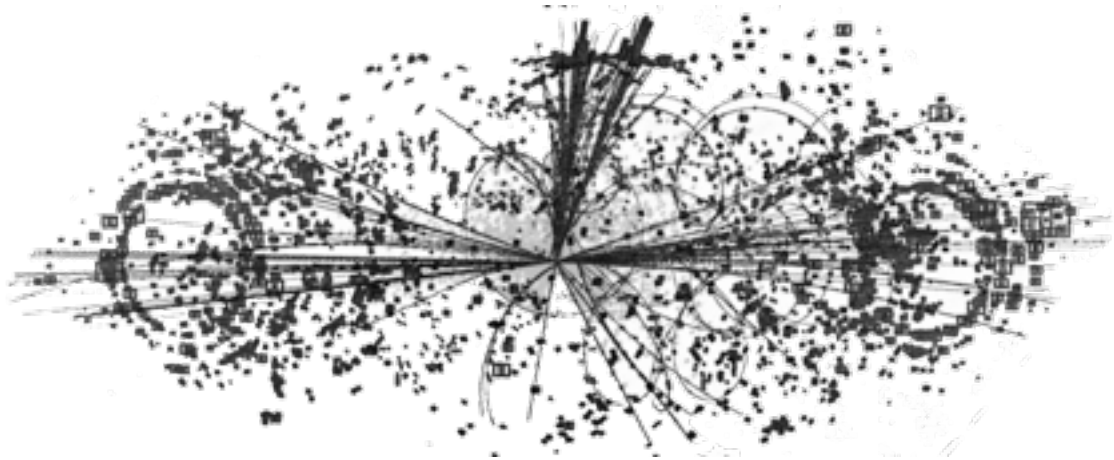


PAT Data & Cleaning

Benedikt Hegner

Mumbai, 27.10.2009

Outline

- PAT data
- PAT config tools
- PAT cleaning

PAT data

PAT vs. AOD

- A few remarks on the role of PAT and PAT data
- AOD and RECO are data tiers with a defined *static* content
- PAT has a *flexible* content
- PAT has a *flexible* workflow
- So once you reach the PAT step you make *choices* about what you consider important *for your physics* use case

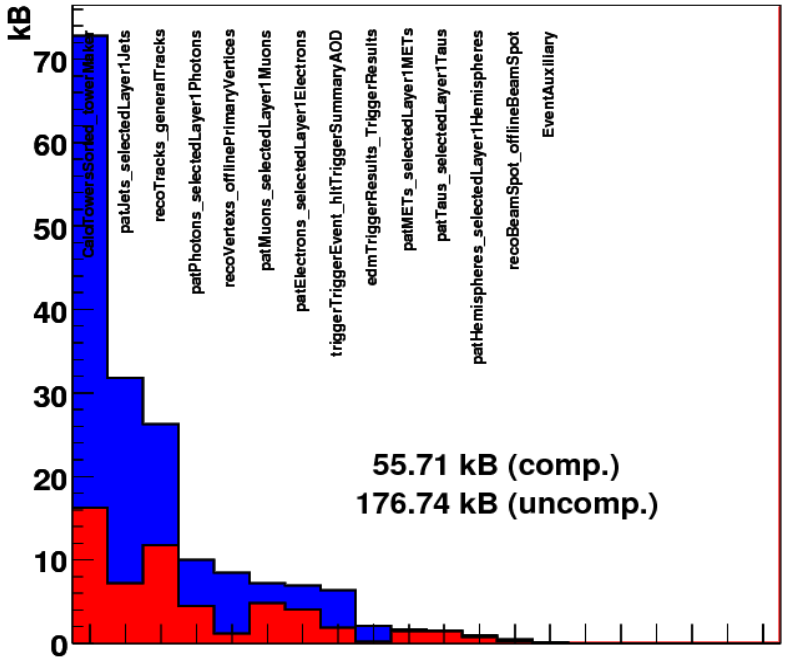
or in CMS slang:
PAT is not a data tier

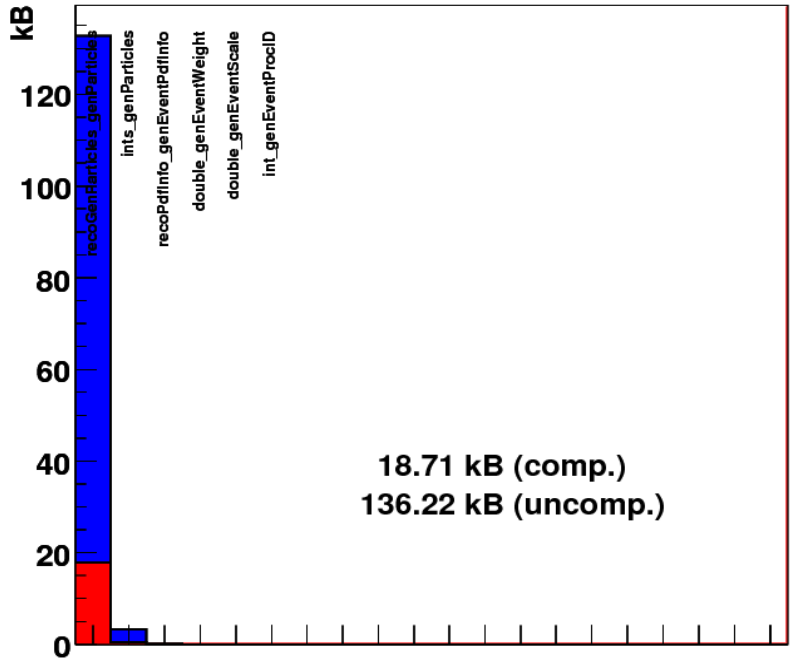
- Today we learn about how to adapt PAT to your physics and what the choices are you can make.
- Making the choices is still up to you though! :-)

PAT Samples

- One choice is to only store what is relevant for your (or your group's) analyses
- To adapt the output of PAT to what you actually need there exist many tools
- One important benchmark is always the output size

> edmEventSize -a -v -p myEventSize.ps -s myEvtSize.root inputFile.root





SWGuidePATEventSize

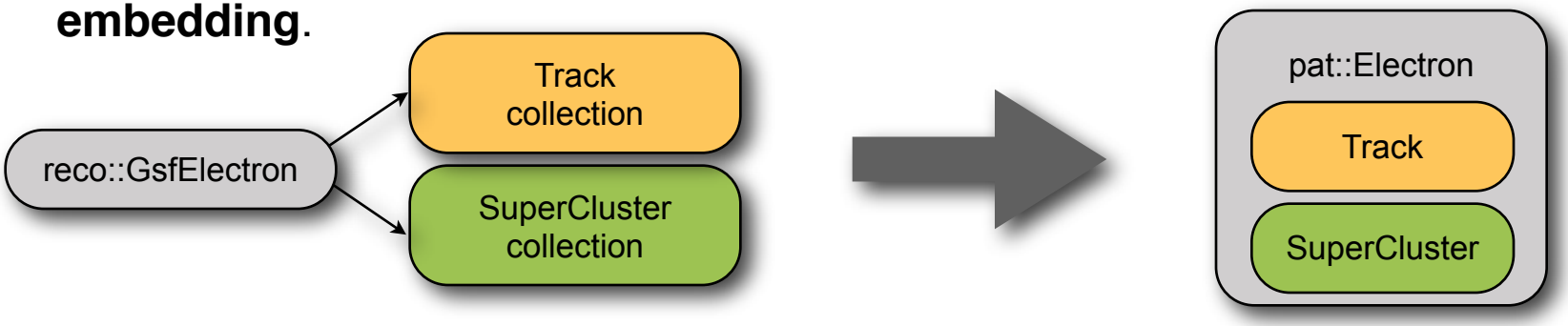
EventContent

- A more detailed breakdown is available via `diskSize.pl`
- Give it a try! It is however an expert tool so the output is not always easy to interpret

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%

Embedding can help you on size

- Let's consider the following use case:
  - You are interested in the super cluster and the track of the few electrons that pass your quality criteria
  - To do this you need to keep the full collection of super clusters and tracks
  - Your event size is dominated by this
  - What now?
- PAT solution: on creating the PAT object you *can* copy over super cluster and track *into* the electron and throw the rest away. That's called **embedding**.



- Whether the `pat::Electron` still has a reference or a copy of the object is transparent to the analysis code

Tools to change the PAT workflow

Remove MC Matching

- There are many standard cases where people want to add/drop/ change certain parts from the default PAT. We support it with a lot of config tools : **SWGuidePATTools**
- I will only give some examples here, e.g.:

```
def removeMCMatching(process,
                     name
                    ):
    """
    ------------------------------------------------------------------
    remove monte carlo matching from a given collection or all PAT
    candidate collections:

    process : process
    name    : collection name; supported are 'Photons', 'Electrons',
              'Muons', 'Taus', 'Jets', 'METs', 'All'
    ------------------------------------------------------------------
    """
```

- **Note:** all other MC related parts are also taken out from the sequence. *Essential* for real data.

Adapt to AOD vs. RECO Data

- To remove all parts which rely on information only present in RECO and not in AOD

```
def restrictInputToAOD(process,
                       names
                       ):
    """
    ----------------------------------------------------------------
    remove pat object production steps which rely on RECO event
    content:

    process : process
    name    : list of collection names; supported are 'Photons',
              'Electrons','Muons', 'Taus', 'Jets', 'METs', 'All'
    ----------------------------------------------------------------
    """
```

- **Note:** add this tool to your config when running on AOD/AODSIM

Adding more Jet Collections

- Tool to add alternative jet collections to the event output:

```
def addJetCollection(process,
                     jetCollection,
                     postfixLabel,
                     doJTA        = True,
                     doBTagging   = True,
                     jetCorrLabel = None,
                     doType1MET   = True,
                     doL1Cleaning = True,
                     doL1Counters = False,
                     genJetCollection = cms.InputTag("aColl")
                    ):
```

- All algorithms by JetMET are supported
- All types like calo, PF, JPT are supported
- Each new jet collection can be accompanied by a new MET collection at the user's will

Switch Jet Collection

- Tool to change the default jet collection in the event output

```
def switchJetCollection(process,
                        jetCollection,
                        doJTA        = True,
                        doBTagging   = True,
                        jetCorrLabel = None,
                        doType1MET   = True,
                        genJetCollection = cms.InputTag("aColl")
                       ):
```

- **Note:** when using addJetCollection and switchJetCollection in one cfg file use addJetCollection first.

Switch to Particle Flow

- To change the input collections for PAT from classic reco to particle flow objects:

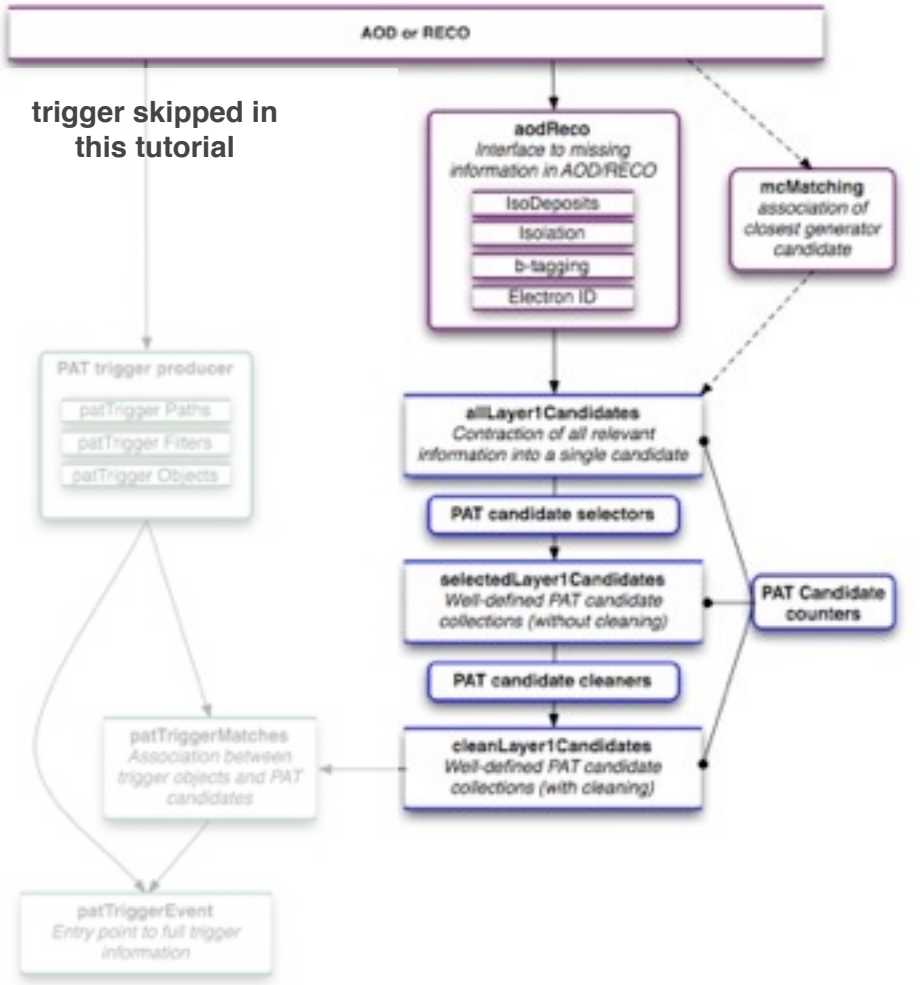
```
# Configure PAT to use PF2PAT instead of AOD sources
from PhysicsTools.PatAlgos.tools.pfTools import *

usePF2PAT(process)

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

- **Note:** Objects, which are not yet supported by particle flow will remain as reco objects.

Cleaning

The PAT workflow

- Have a look at **SWGuidePATWorkflow**



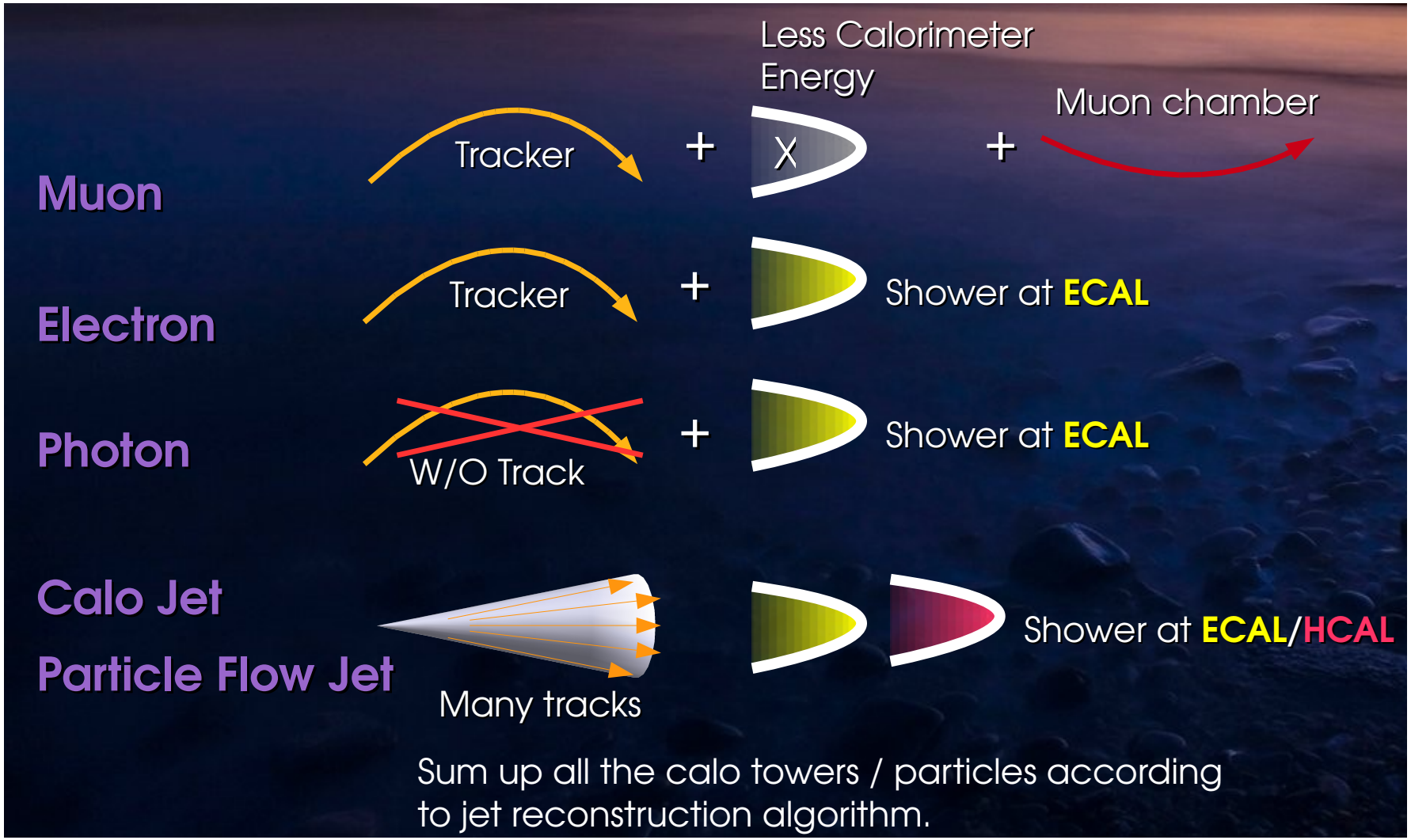
POG pre-production steps

Basic collections

Selected collection

Cleaned collection

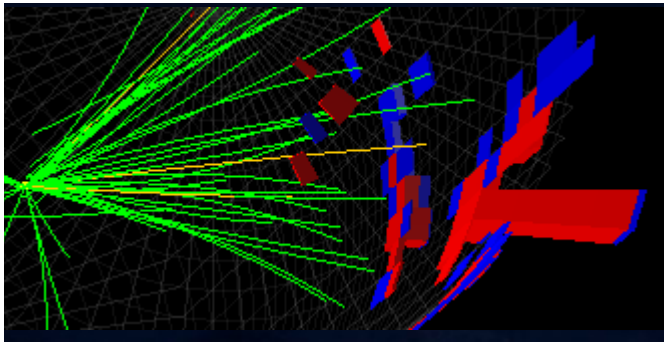
Overlap in Reconstruction



(by Kai-Feng Chen)

Cleaning as Disambiguation

Simple example: energy deposit in ECAL



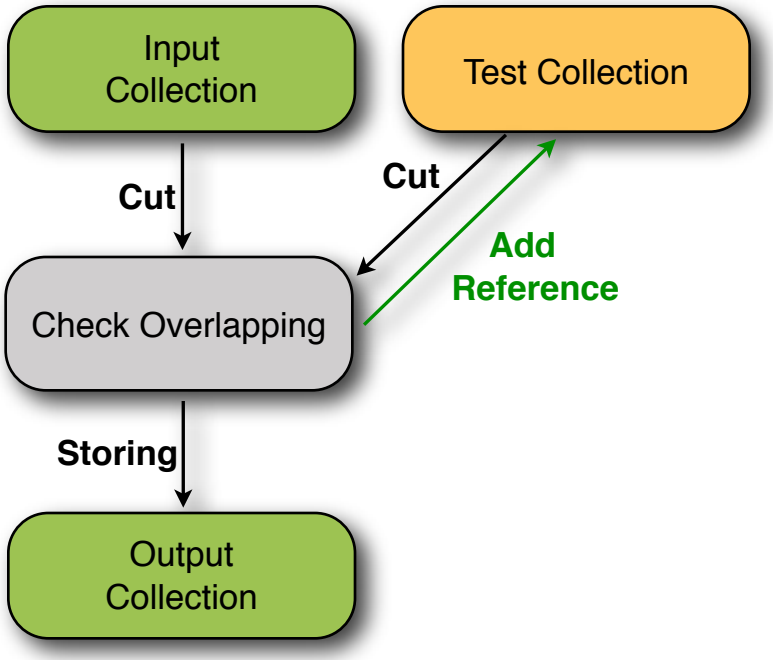
- Can form a photon
- If it matched to a track - forms an electron
- run through jet algorithms - forms a jet
- ...

Cleaning consists of disambiguation

- It is *analysis dependent*; depends on what is considered as electron, jet, ...
- The PAT provides *tools* to disambiguate
  it does not choose for you!

General Cleaning Layout

- Basically the steps in our cleaning modules are the following:
  - read in the input collection and the collections to test against
  - Apply generic cuts to both collections
  - Check for overlaps
  - Overlapping objects can be *kept* or *discarded*
  - Store reference to overlapping objects in test collection to input collection (*flagging*)
  - Apply generic cuts to input collection before storing the result



Default Cleaning

- Default configurations in PAT
  (**note:** the right settings are very analysis specific)
  - *In general* - no pre-selection nor final cut
  - **Muons***:* considered clean (no selection, no x-cleaning)
  - **Electrons***:* clean against muons that overlap by deltaR <0.3 *(add reference to electrons)*
  - **Photons***:* clean against electrons with the same supercluster seed *(discard photons)*
  - **Taus**: only taus that pass the discriminator by isolation; check overlaps in deltaR (<0.3) with electrons and muons *(add reference to taus)*
  - **Jets***:* clean against other collections by deltaR (<0.5); also against tracker isolated electrons [deltaR < 0.3; track-Iso<3, pT> 10 GeV] *(reference with label "tkIsoElectrons")*.

Example - Electron Cleaning

- From PhysicsTools/PatAlgos/python/cleaningLayer1/electronCleaner_cfi.py

```
cleanLayer1Electrons = cms.EDFilter("PATElectronCleaner",
    ## pat electron input source
    src = cms.InputTag("selectedLayer1Electrons"),
    # preselection (any string-based cut for pat::Electron)
    preselection = cms.string(''),
    # overlap checking configurables
    checkOverlaps = cms.PSet(
        muons = cms.PSet(
           src = cms.InputTag("cleanLayer1Muons"),
           algorithm = cms.string("byDeltaR"),
           preselection = cms.string(""),
           deltaR = cms.double(0.3),
           checkRecoComponents = cms.bool(False),
           pairCut = cms.string(""),
           requireNoOverlaps = cms.bool(False),
       ),
   ),
  finalCut = cms.string('')
)
```

Example - Electron Cleaning

- From PhysicsTools/PatAlgos/python/cleaningLayer1/electronCleaner_cfi.py

```
cleanLayer1Electrons = cms.EDFilter("PATElectronCleaner",
    ## pat electron input source
    src = cms.InputTag("selectedLayer1Electrons"),
    # preselection (any string-based cut for pat::Electron)
    preselection = cms.string(''),
    # overlap checking configurables
    checkOverlaps = cms.PSet(
        muons = cms.PSet(
          src = cms.InputTag("cleanLayer1Muons"),
          algorithm = cms.string("byDeltaR"),
          preselection = cms.string(""),
          deltaR = cms.double(0.3),
          checkRecoComponents = cms.bool(False),
          pairCut = cms.string(""),
          requireNoOverlaps = cms.bool(False),
        ),
    ),
  finalCut = cms.string('')
)
```

Look at the electrons that passed the previous step

Clean against muons that passed the previous cleaning

Example - Electron Cleaning

- From PhysicsTools/PatAlgos/python/cleaningLayer1/electronCleaner_cfi.py

```
cleanLayer1Electrons = cms.EDFilter("PATElectronCleaner",
    ## pat electron input source
    src = cms.InputTag("selectedLayer1Electrons"),
    # preselection (any string-based cut for pat::Electron)
    preselection = cms.string(''),
    # overlap checking configurables
    checkOverlaps = cms.PSet(
        muons = cms.PSet(
           src = cms.InputTag("cleanLayer1Muons"),
           algorithm = cms.string("byDeltaR"),
           preselection = cms.string(""),
           deltaR = cms.double(0.3),
           checkRecoComponents = cms.bool(False),
           pairCut = cms.string(""),
           requireNoOverlaps = cms.bool(False),
        ),
    ),
  finalCut = cms.string('')
)
```

No pre-selection

Set the cut value to 0.3

No further relative cut between electron/muon

Example - Electron Cleaning

- From PhysicsTools/PatAlgos/python/cleaningLayer1/electronCleaner_cfi.py

```
cleanLayer1Electrons = cms.EDFilter("PATElectronCleaner",
    ## pat electron input source
    src = cms.InputTag("selectedLayer1Electrons"),
    # preselection (any string-based cut for pat::Electron)
    preselection = cms.string(''),
    # overlap checking configurables
    checkOverlaps = cms.PSet(
        muons = cms.PSet(
           src = cms.InputTag("cleanLayer1Muons"),
           algorithm = cms.string("byDeltaR"),
           preselection = cms.string(""),
           deltaR = cms.double(0.3),
           checkRecoComponents = cms.bool(False),
           pairCut = cms.string(""),
           requireNoOverlaps = cms.bool(False),
        ),
    ),
  finalCut = cms.string('')
)
```

No need to check overlap on component level (e.g. superclusters)

Do not discard the overlapping candidates; Keep the reference to the muon

How to handle overlap in your code

- Result of cleaning is available directly from objects:

  `bool hasOverlap(string& label)`

- Checks if there was any overlap with collection named *label*., e.g.:

  `myObject.hasOverlap("muons")`

- Full list of collection labels that found in "overlap checks":

  `vector<string>& overlapLabels()`

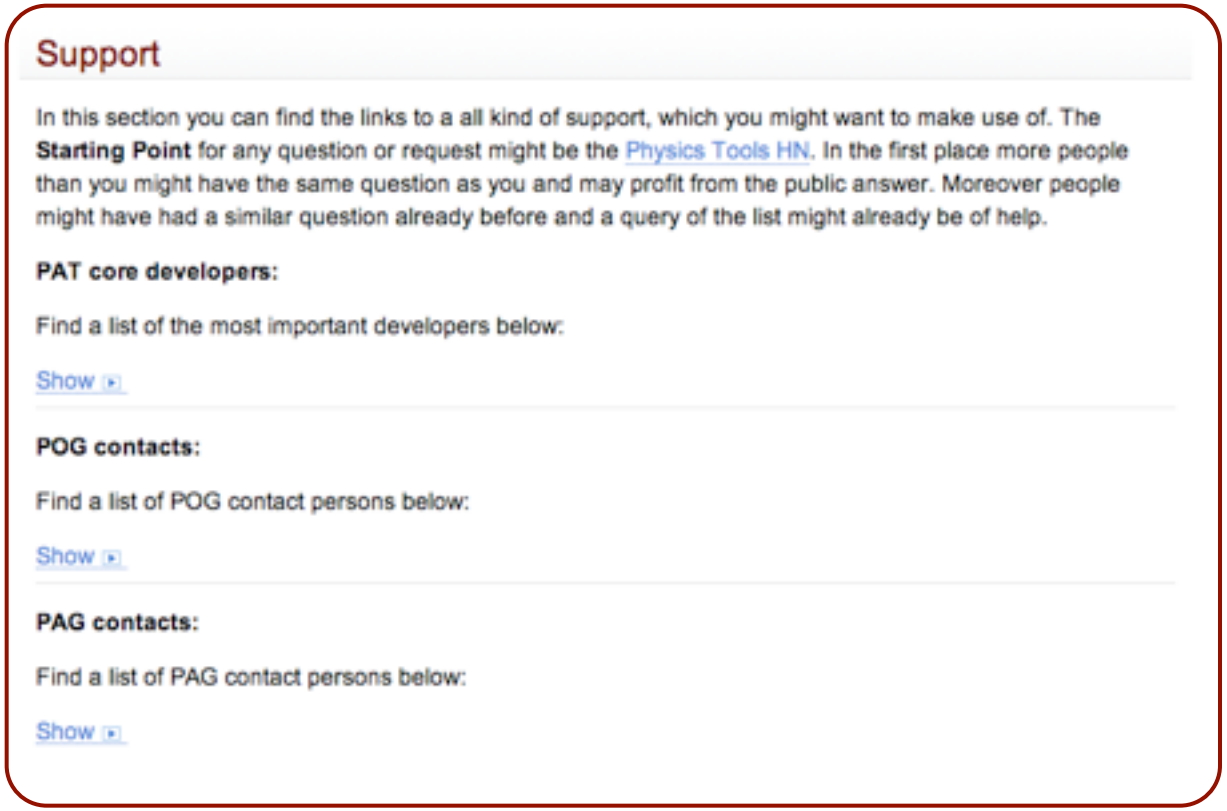
- Get the list of items which overlap:

  `CandidatePtrVector& overlaps(string& label)`

- Something to try out yourself can be found at
  SWGuidePATCleaningExercises

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

Conclusions

- PAT is not a static instance of objects. It is flexible in many aspects and supports modifications/adaptions to content and workflow
- The cleaning layer is a versatile and highly configurable tool
- PAT config tools is a field which will get extended more in the next few months.
- PAT does not relieve you from using your brain on what you want to do, but it makes it easier once you know what you want.
- All developers are happy to receive feedback/requests/hints for further developments - in the end we create the tools for **you**

That's all :-)