Parametrized simulation and analysis

Clement Helsens (CERN)

On behalf of the FCCSW team

FCC Week 2016, Rome

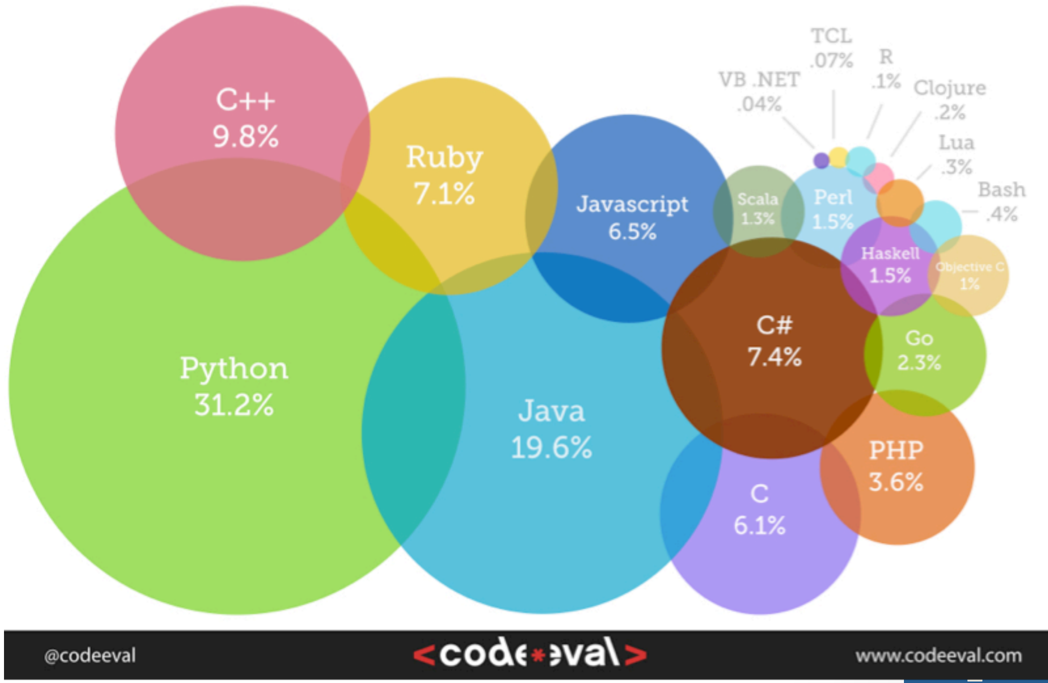
Outline

1. Why Python
2. Heppy
3. Papas example for FCC-ee
4. Delphes example for FCC-hh
5. Summary/Next steps

Why python

- Super easy to learn
- Light & short code
- Good performance
  - usually wraps C or C++ modules
- « Batteries included »
  - massive and easy-to-use standard library

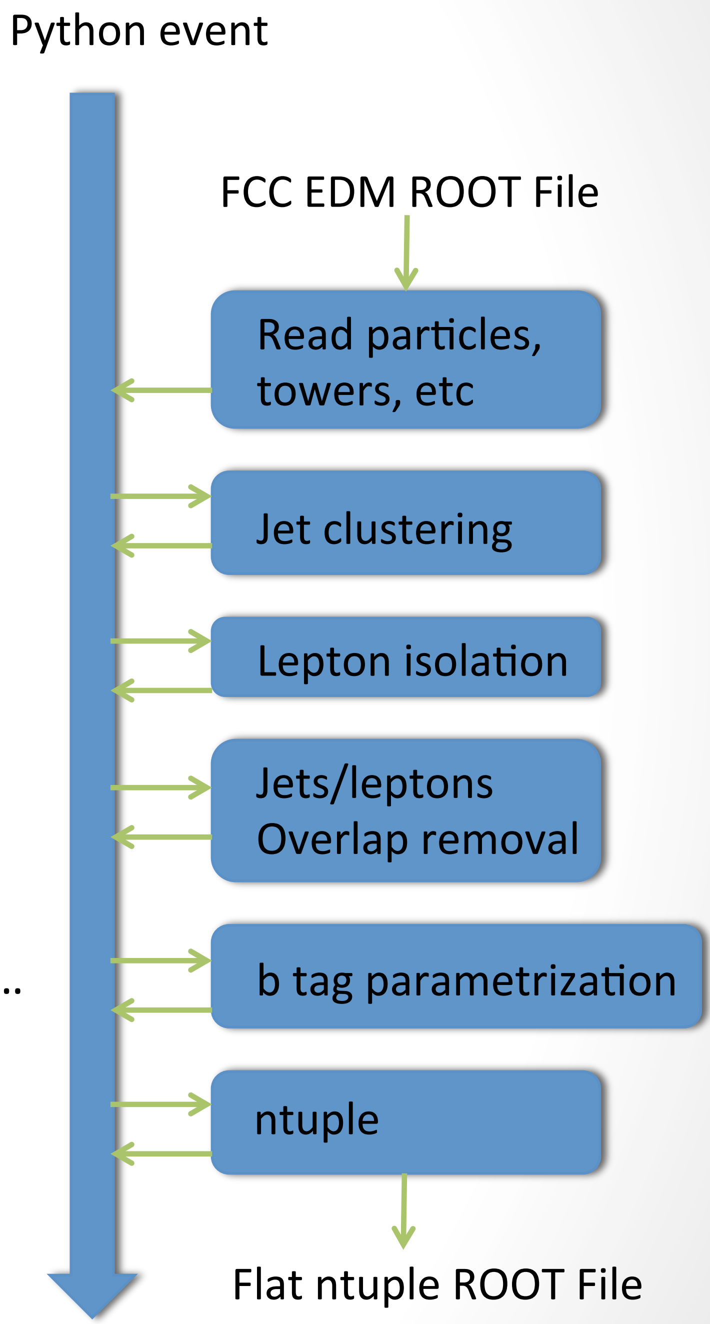
Most Popular Coding Languages of 2015



- Dynamic typing
  - good for multi-channel analyses
  - code highly reusable
- Dynamic object modification
  - Attach new attributes (or methods) to an existing object
- Productivity x 5-10 w/r C++
- A lot of fun ☺

Heppy

- Advanced framework PyROOT macros
- Design ~ Athena, CMSSW, Gaudi, Marlin
- Goals:
  - high-level reco & selection
  - write out flat ntuple or histograms for statistical analysis
- Modules are shared with FCC, CMS…
  - particle gun
  - isolation
  - M3, MET / missing energy
  - Recoil, resonance
  - filter, matcher, masker
- Take any kind of object in input from FCC, CMS…
- Modules easy to write
  - Python is terse
  - Large library of tools

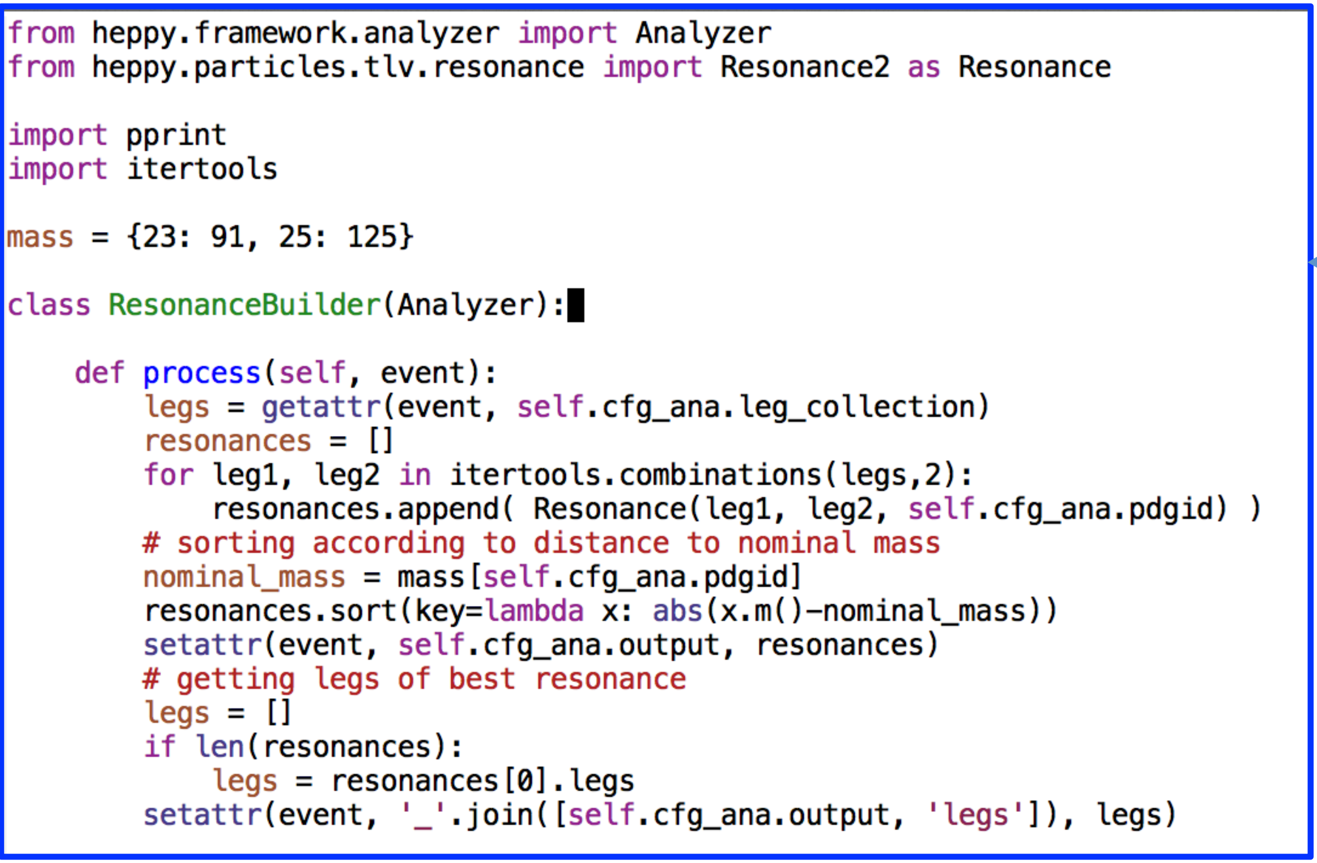


Heppy usage

- ~50 users
- CMS analyses using Heppy
  - Higgs: H→ZZ→4l, H→tau tau, ttH→multileptons, W/Z H→bb
  - Susy fully hadronic , 1 lepton, multilepton
  - W mass
- Can read any event format
  - CMS EDM
  - FCC EDM
  - Plain ROOT (pheno studies)
  - LCIO for ILC/CLIC
  - Soon: ATLAS
- Write transparent code for several experiments

https://github.com/HEP-FCC/heppy

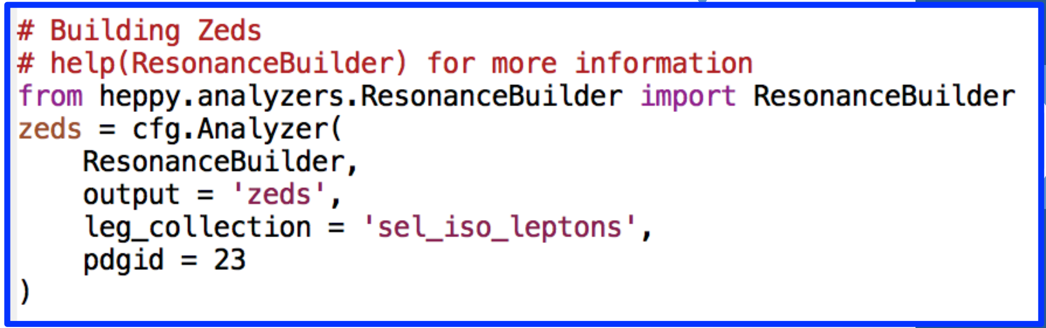
Example



```
from heppy.framework.analyzer import Analyzer
from heppy.particles.tlv.resonance import Resonance2 as Resonance

import pprint
import itertools

mass = {23: 91, 25: 125}

class ResonanceBuilder(Analyzer):

    def process(self, event):
        legs = getattr(event, self.cfg_ana.leg_collection)
        resonances = []
        for leg1, leg2 in itertools.combinations(legs,2):
            resonances.append( Resonance(leg1, leg2, self.cfg_ana.pdgid) )
        # sorting according to distance to nominal mass
        nominal_mass = mass[self.cfg_ana.pdgid]
        resonances.sort(key=lambda x: abs(x.m()-nominal_mass))
        setattr(event, self.cfg_ana.output, resonances)
        # getting legs of best resonance
        legs = []
        if len(resonances):
            legs = resonances[0].legs
        setattr(event, '_'.join([self.cfg_ana.output, 'legs']), legs)
```

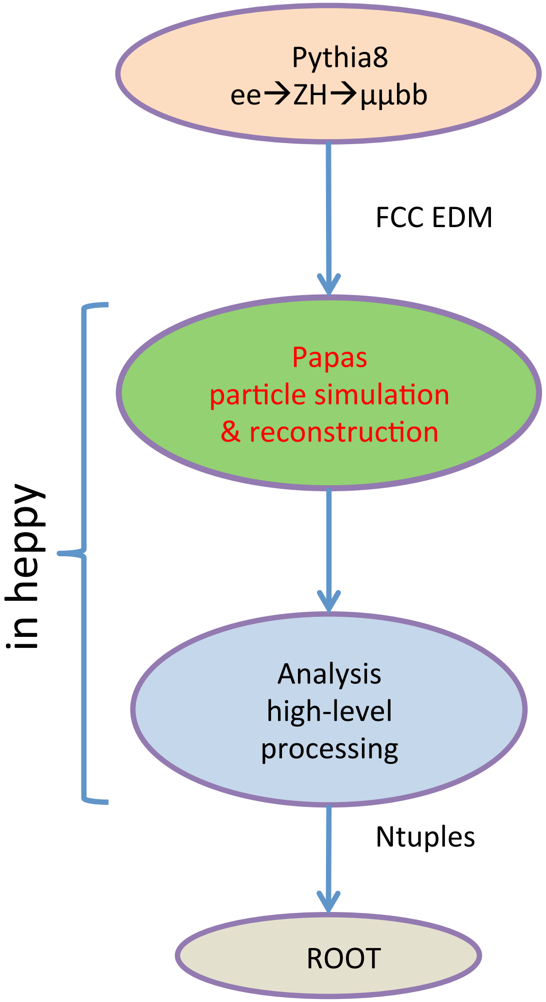
Typical analyzer code…

…and configuration

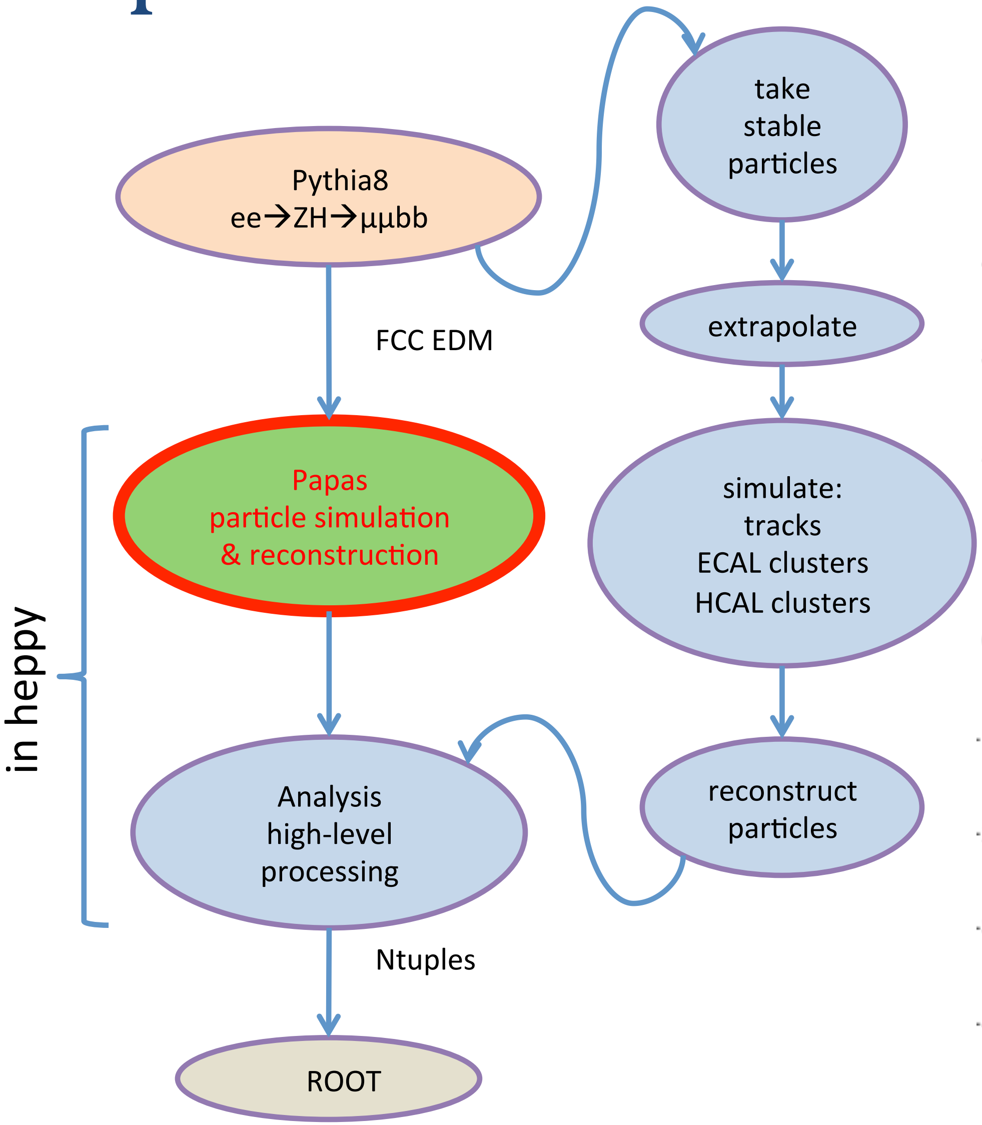


```
# Building Zeds
# help(ResonanceBuilder) for more information
from heppy.analyzers.ResonanceBuilder import ResonanceBuilder
zeds = cfg.Analyzer(
    ResonanceBuilder,
    output = 'zeds',
    leg_collection = 'sel_iso_leptons',
    pdgid = 23
)
```

Example: ZH analysis



Papas



Pythia8
ee→ZH→μμbb

FCC EDM

take
stable
particles

extrapolate

simulate:
tracks
ECAL clusters
HCAL clusters

reconstruct
particles

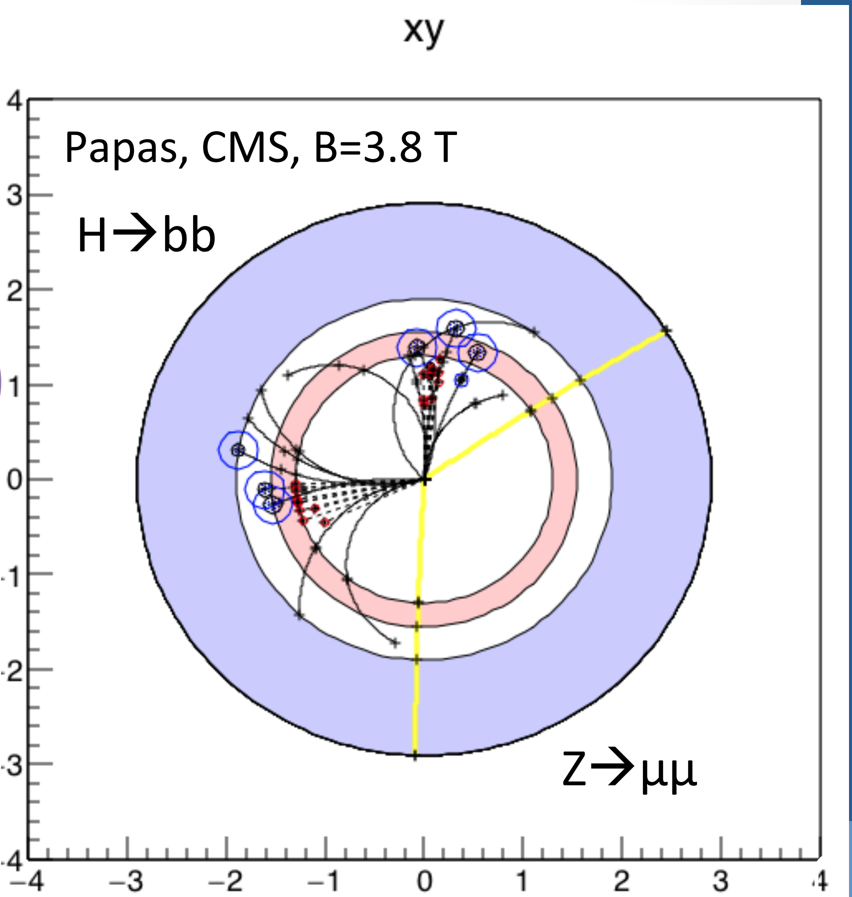
in heppy

Papas
particle simulation
& reconstruction

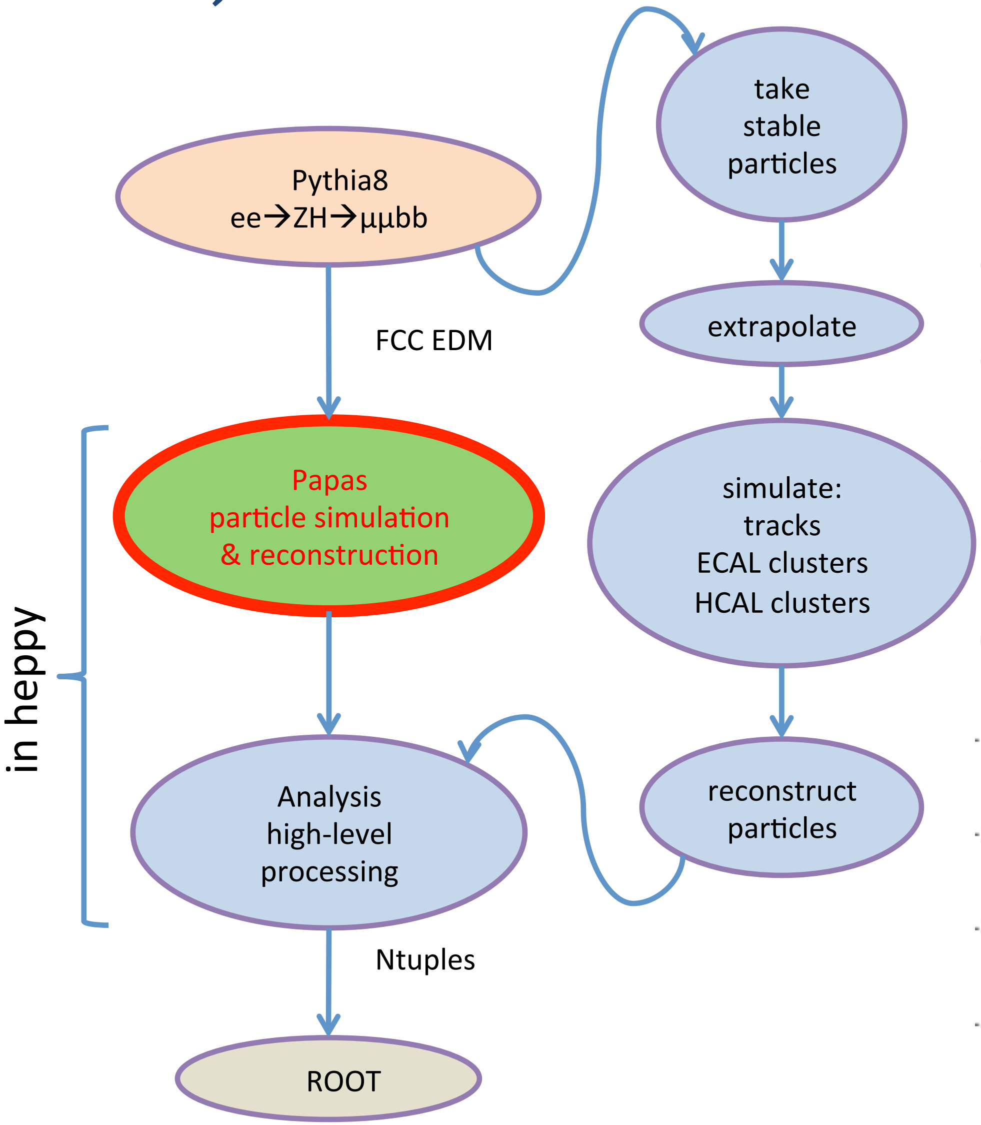
Analysis
high-level
processing

Ntuples

ROOT



CMS, B = 1.5 T



Pythia8
ee→ZH→μμbb

FCC EDM

take
stable
particles

extrapolate

simulate:
tracks
ECAL clusters
HCAL clusters

reconstruct
particles

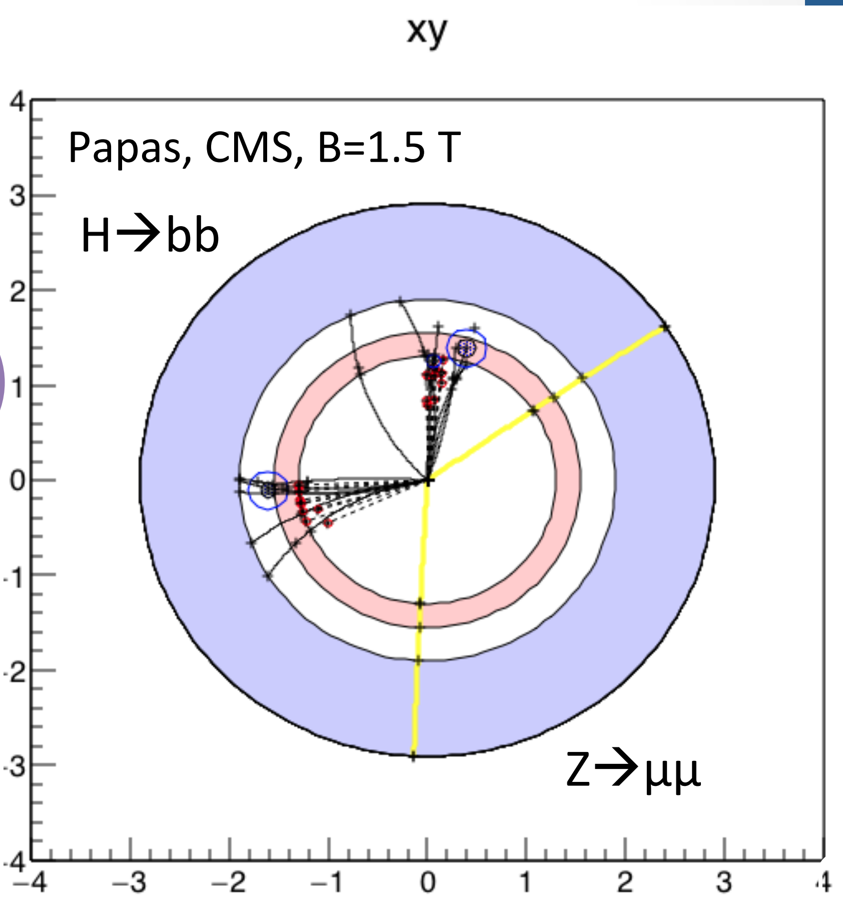
Papas
particle simulation
& reconstruction

Analysis
high-level
processing

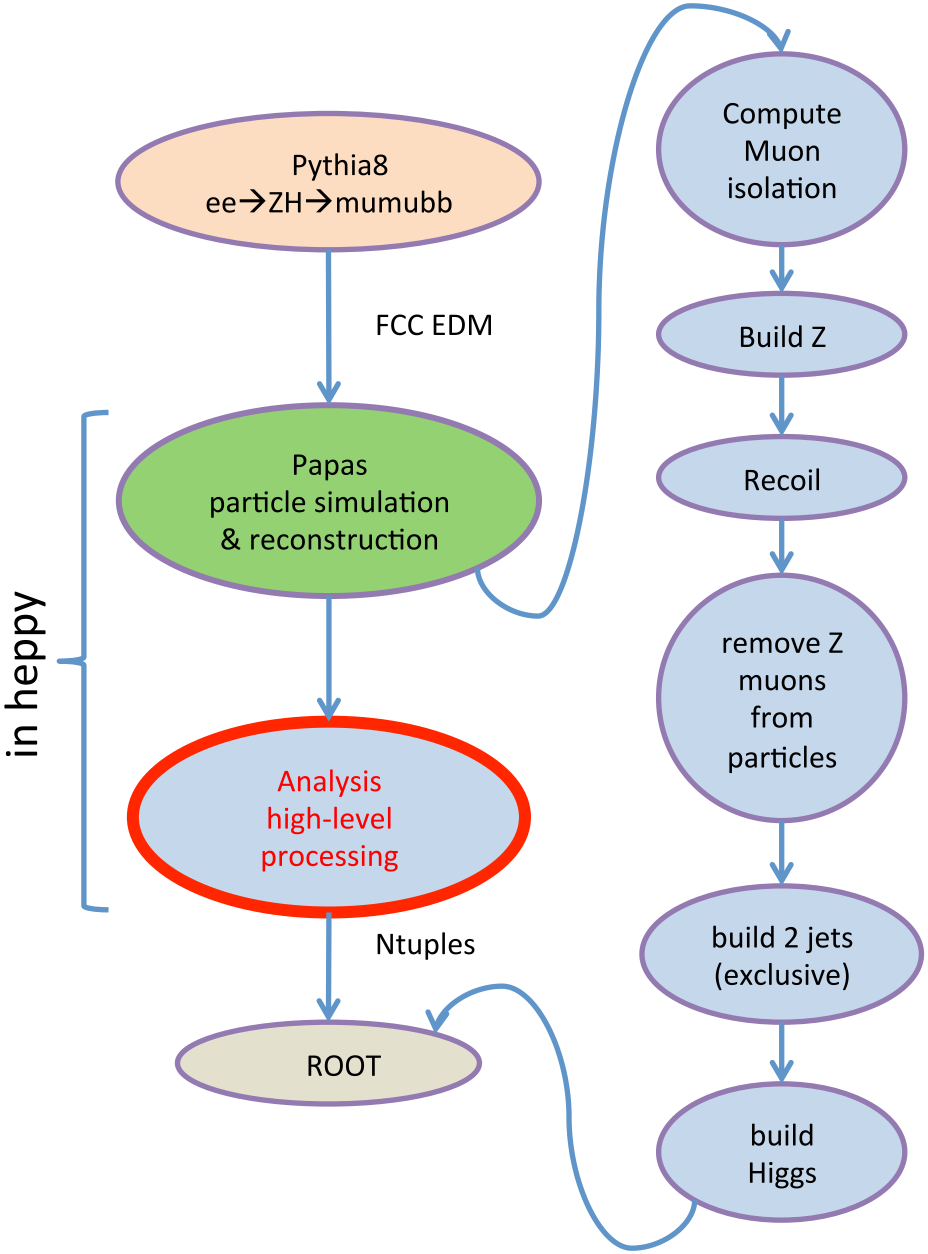
Ntuples

ROOT

in heppy



Example ZH analysis: processing flow



Pythia8
ee→ZH→mumubb

FCC EDM

Papas
particle simulation
& reconstruction

Analysis
high-level
processing

Ntuples

ROOT

in heppy

Compute
Muon
isolation

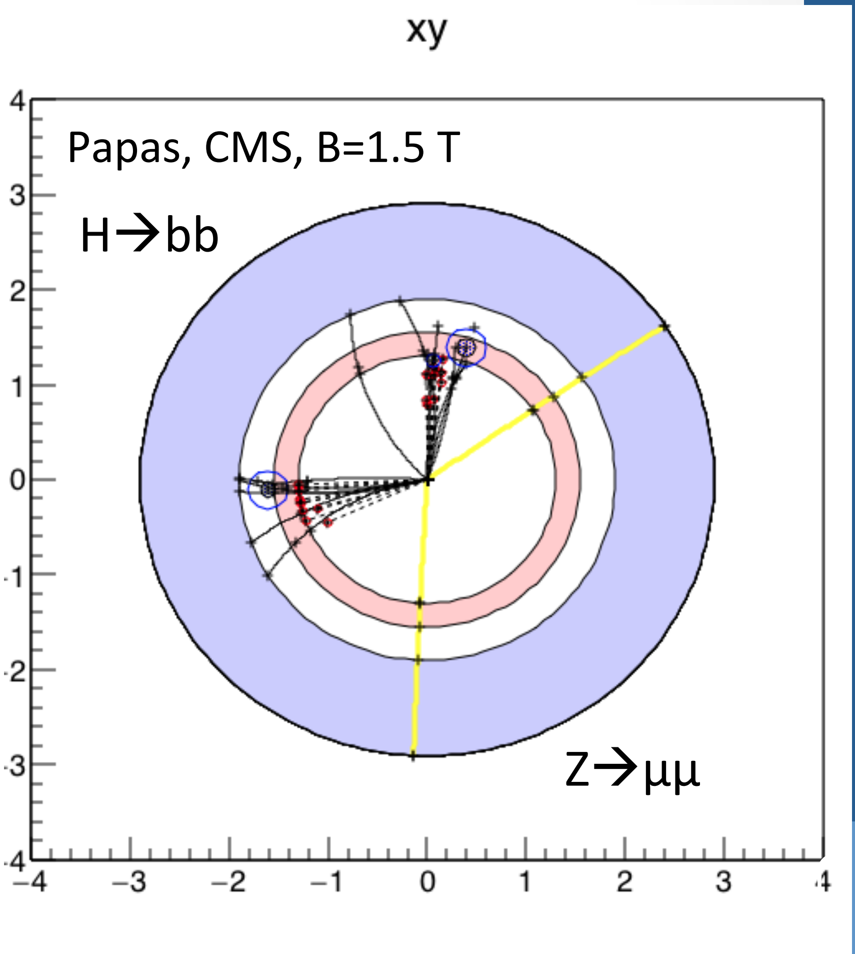
Build Z

Recoil

remove Z
muons
from
particles

build 2 jets
(exclusive)

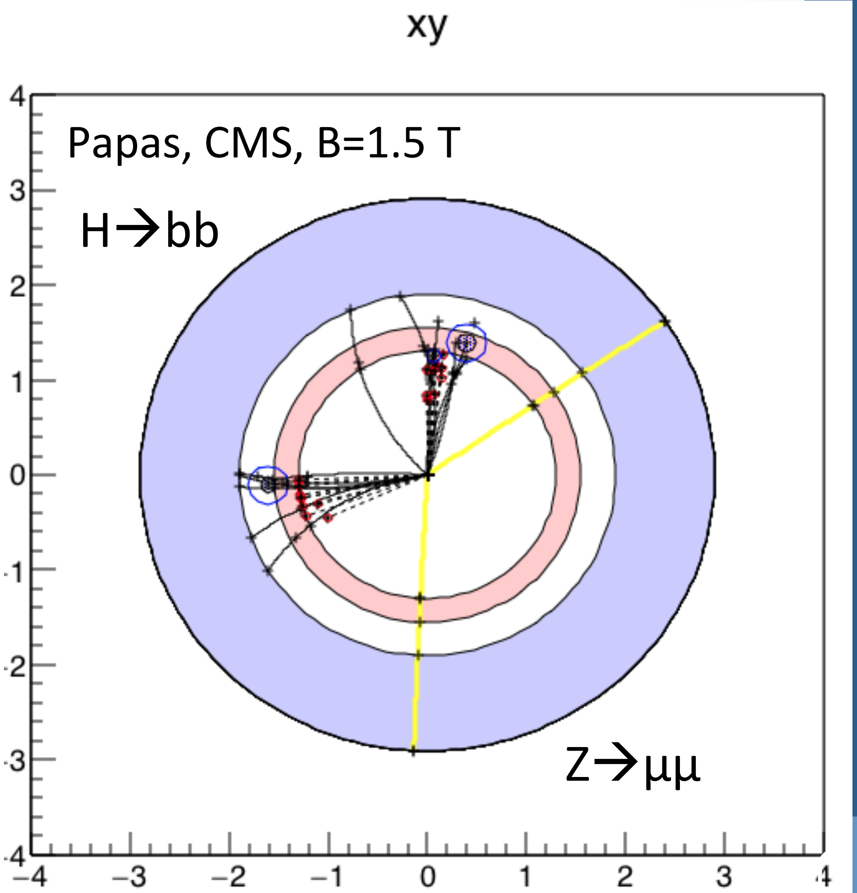
build
Higgs



Jet Reconstruction Performance

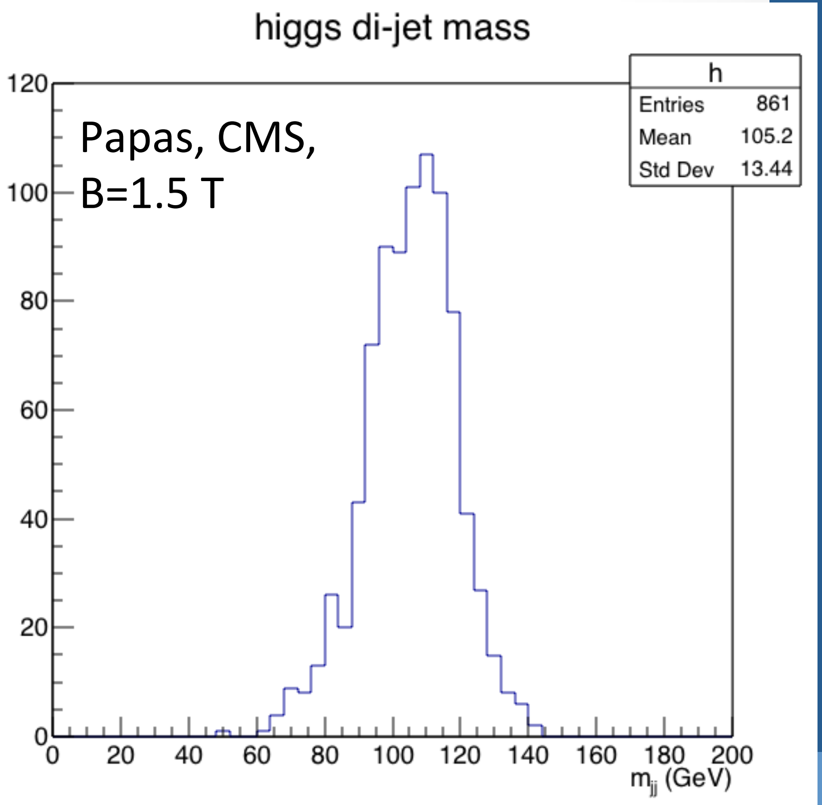
- Mass of the Higgs:
  - Use 2 jets (exclusive reconstruction)
  - $m_H = m(2\ jets)$

- Mass of Higgs candidate obtained from the recoil
  - $m_H = m(p_{ini} - p_Z)$
  - study particle flow reconstruction of jet particles



Jet Reconstruction Performance

- Mass of the Higgs:
  - Use 2 jets (exclusive reconstruction)
  - $m_H = m(2 \text{ jets})$
- Mass of Higgs candidate obtained from the recoil
  - $m_H = m(p_{ini} - p_Z)$
  - study particle flow reconstruction of jet particles

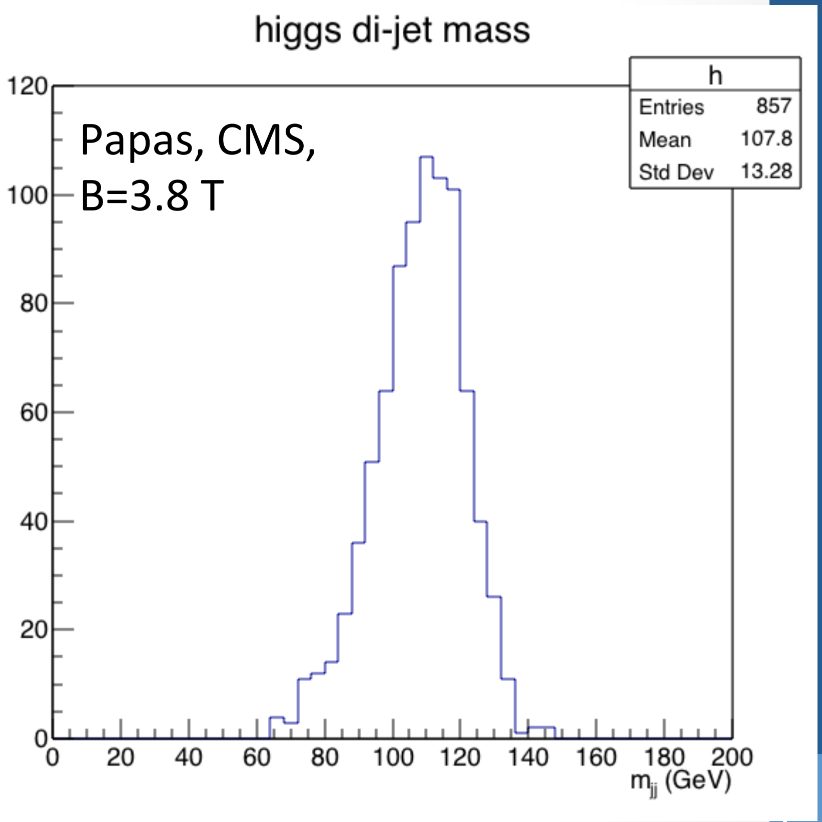


Jet Reconstruction Performance

- Mass of the Higgs:
  - Use 2 jets (exclusive reconstruction)
  - $m_H = m(2 \text{ jets})$

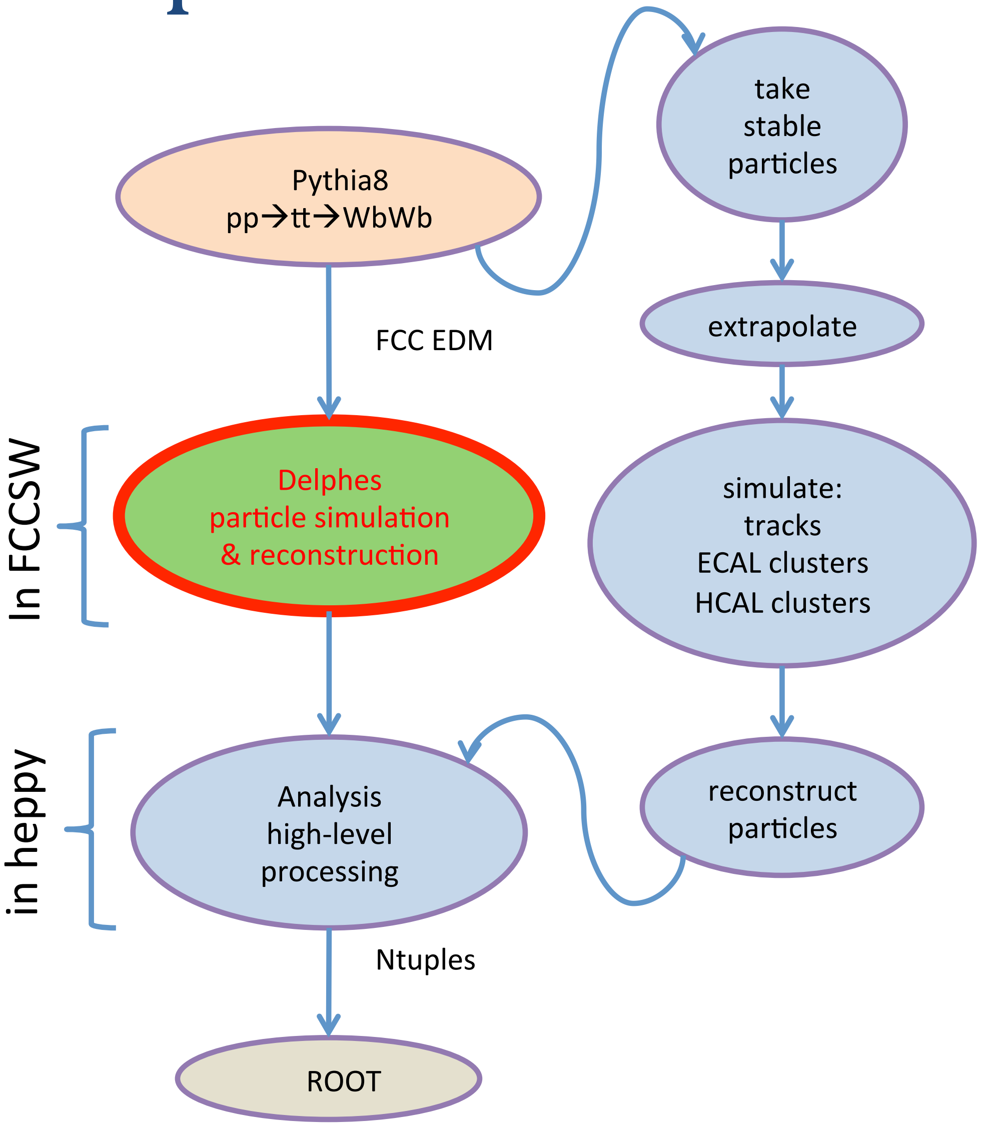
- Mass of Higgs candidate obtained from the recoil
  - $m_H = m(p_{ini} - p_Z)$
  - study particle flow reconstruction of jet particles

- Small influence of the B-Field



Delphes in FCCSW

Delphes used since many years for pheno studies



Pythia8
pp→tt→WbWb

FCC EDM

Delphes
particle simulation
& reconstruction

Analysis
high-level
processing

Ntuples

ROOT

take
stable
particles

extrapolate

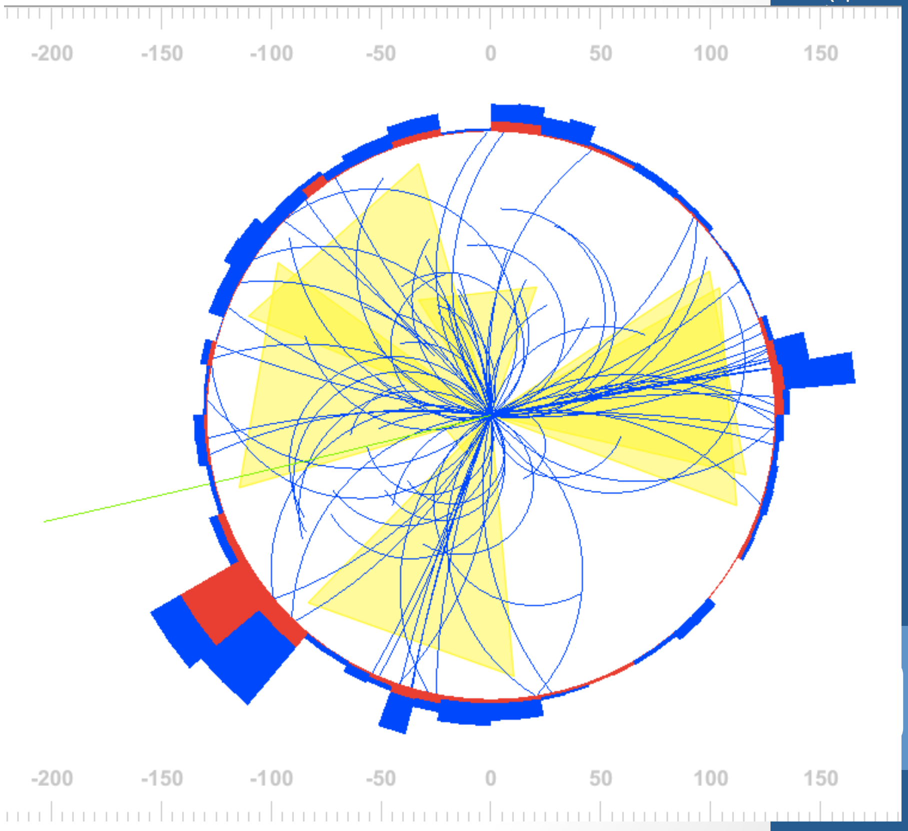
simulate:
tracks
ECAL clusters
HCAL clusters

reconstruct
particles

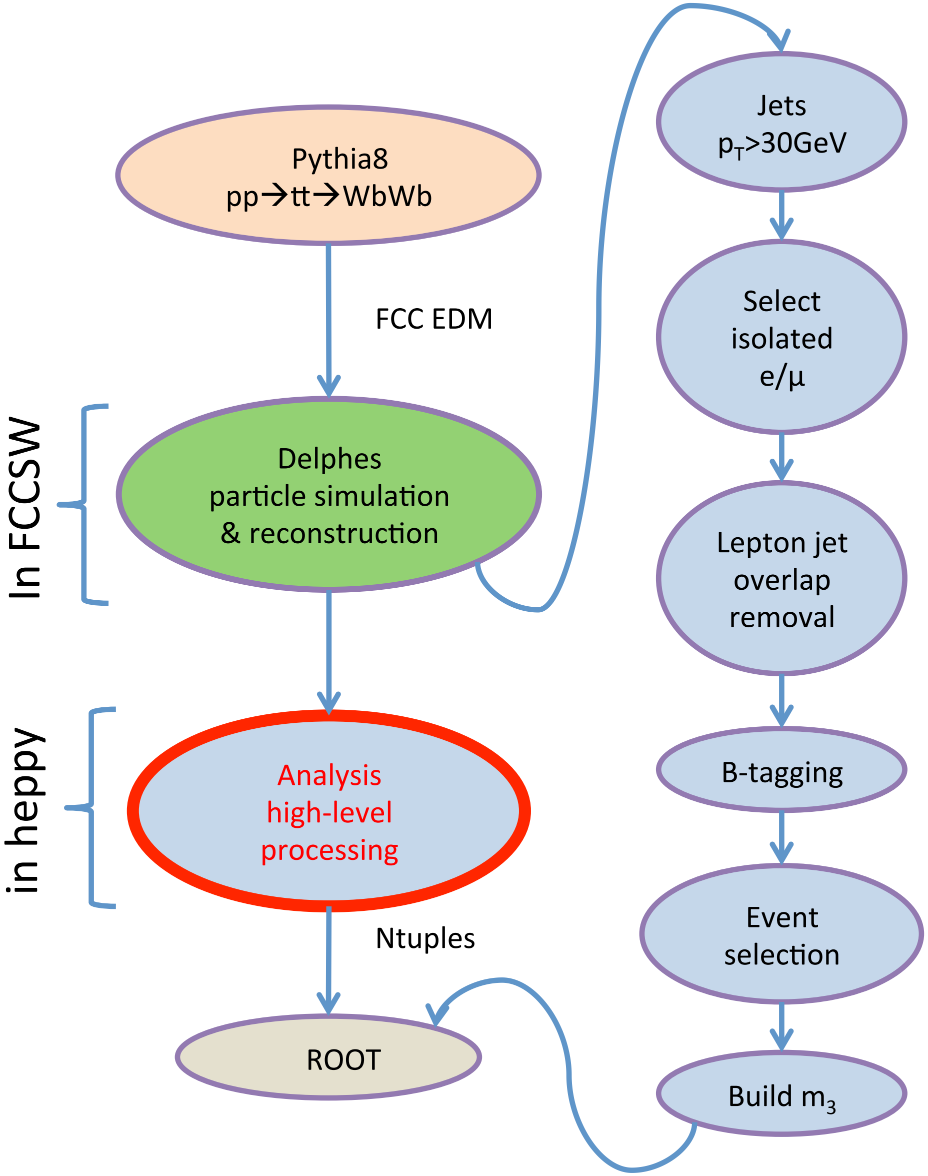
In FCCSW

in heppy

pp->tt->4jets + muon



Example ttbar analysis: processing flow


Pythia8
pp→tt→WbWb

FCC EDM

In FCCSW

Delphes
particle simulation
& reconstruction

in heppy

Analysis
high-level
processing

Ntuples

ROOT

Jets
p_T>30GeV

Select
isolated
e/μ

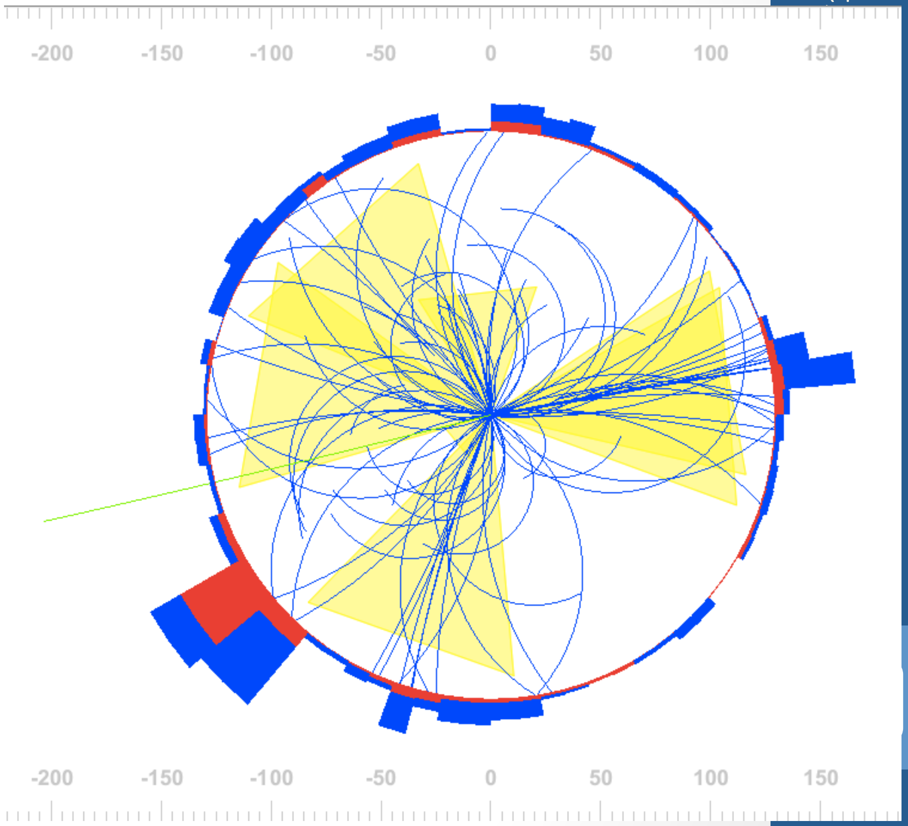
Lepton jet
overlap
removal

B-tagging

Event
selection

Build m_3

pp->tt->4jets + muon

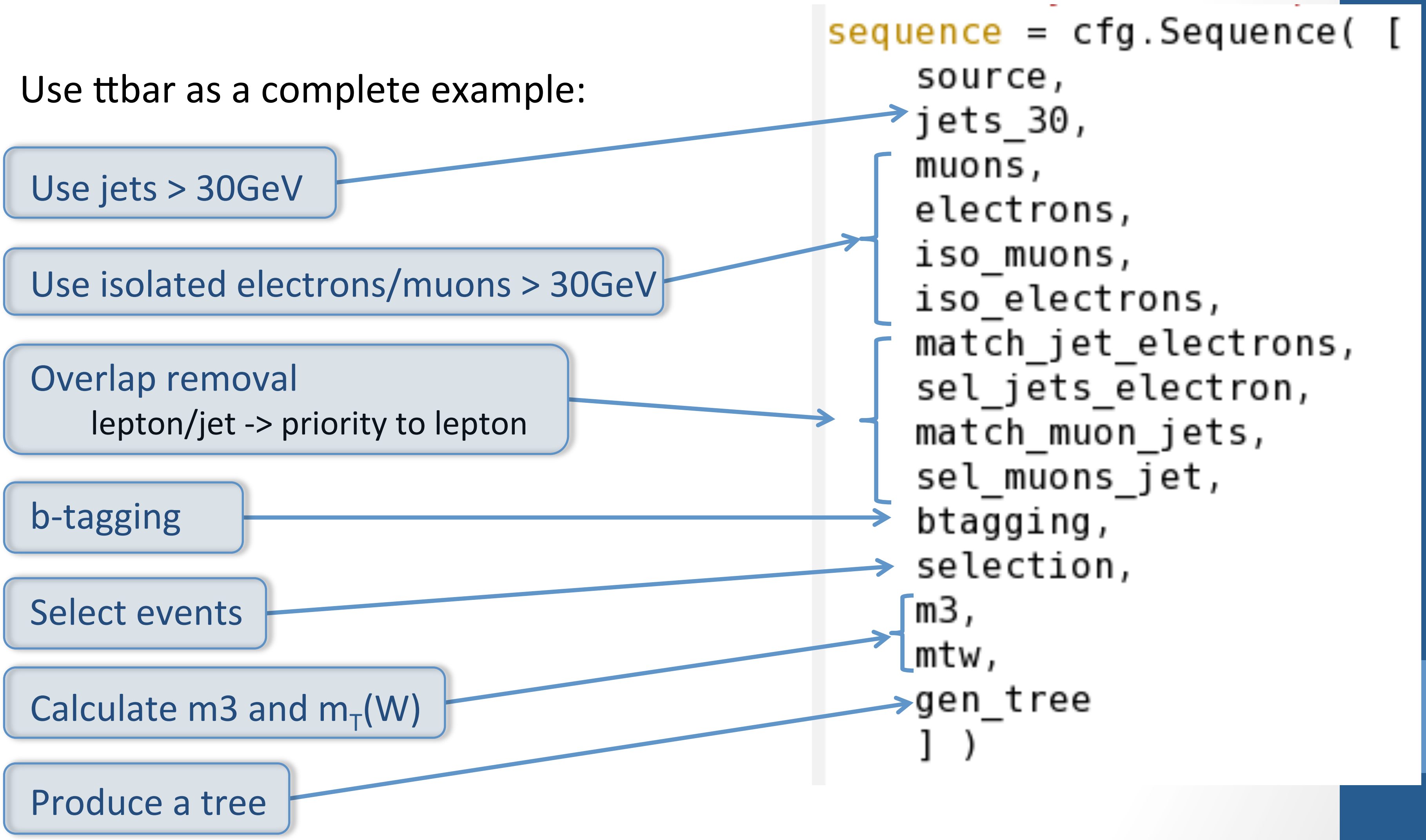


FCC-hh use case

- Aim: Provide an analysis skeleton

Definition of a sequence of analyzers
Analyzers will process each event in this order

- Use ttbar as a complete example:



Use jets > 30GeV

Use isolated electrons/muons > 30GeV

Overlap removal
lepton/jet -> priority to lepton

b-tagging

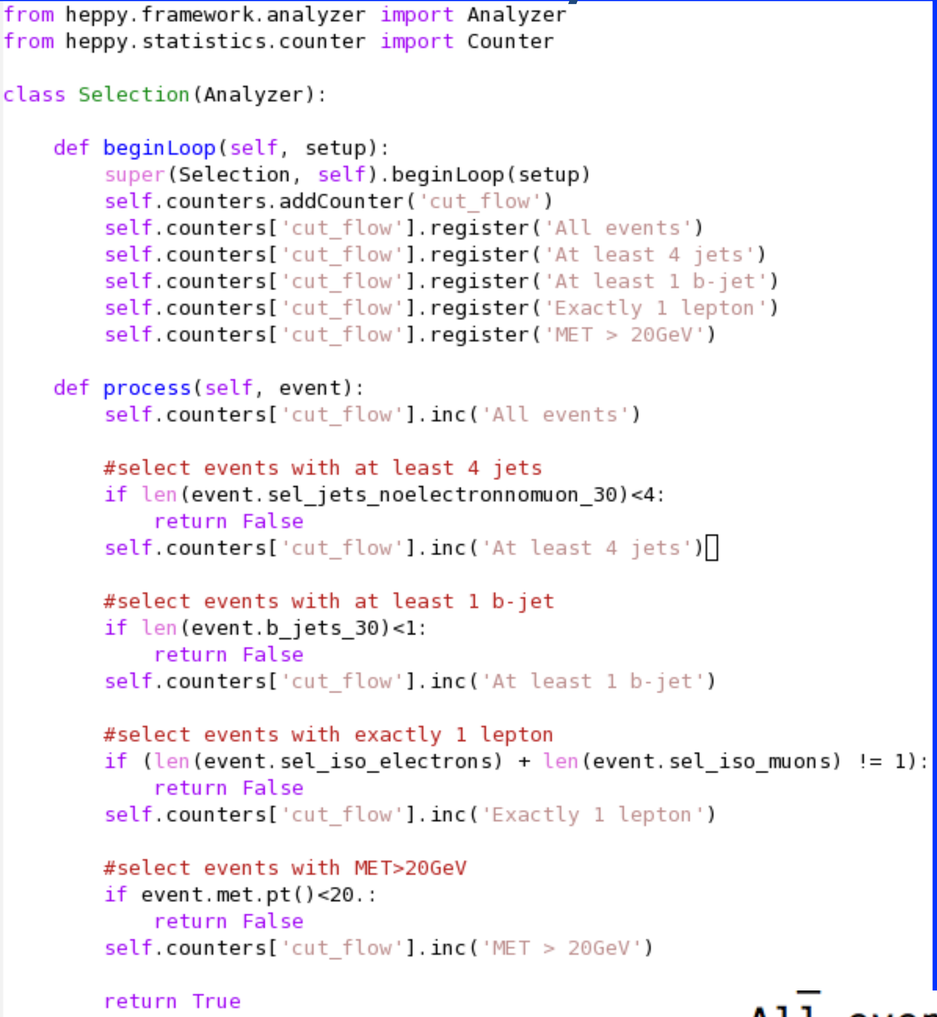
Select events

Calculate m3 and $m_T(W)$

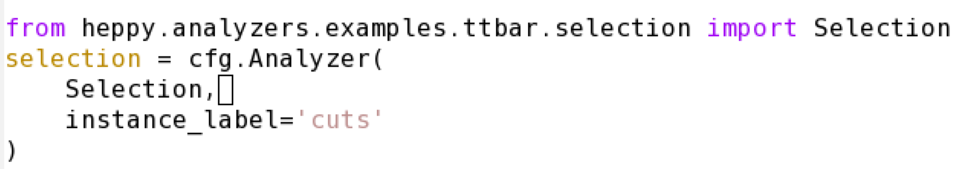
Produce a tree

```
sequence = cfg.Sequence( [
        source,
        jets_30,
        muons,
        electrons,
        iso_muons,
        iso_electrons,
        match_jet_electrons,
        sel_jets_electron,
        match_muon_jets,
        sel_muons_jet,
        btagging,
        selection,
        m3,
        mtw,
        gen_tree
        ] )
```

Selection cut/flow



```
from heppy.framework.analyzer import Analyzer
from heppy.statistics.counter import Counter

class Selection(Analyzer):

    def beginLoop(self, setup):
        super(Selection, self).beginLoop(setup)
        self.counters.addCounter('cut_flow')
        self.counters['cut_flow'].register('All events')
        self.counters['cut_flow'].register('At least 4 jets')
        self.counters['cut_flow'].register('At least 1 b-jet')
        self.counters['cut_flow'].register('Exactly 1 lepton')
        self.counters['cut_flow'].register('MET > 20GeV')

    def process(self, event):
        self.counters['cut_flow'].inc('All events')

        #select events with at least 4 jets
        if len(event.sel_jets_noelectronnomuon_30)<4:
            return False
        self.counters['cut_flow'].inc('At least 4 jets')□

        #select events with at least 1 b-jet
        if len(event.b_jets_30)<1:
            return False
        self.counters['cut_flow'].inc('At least 1 b-jet')

        #select events with exactly 1 lepton
        if (len(event.sel_iso_electrons) + len(event.sel_iso_muons) != 1):
            return False
        self.counters['cut_flow'].inc('Exactly 1 lepton')

        #select events with MET>20GeV
        if event.met.pt()<20.:
            return False
        self.counters['cut_flow'].inc('MET > 20GeV')

        return True
```

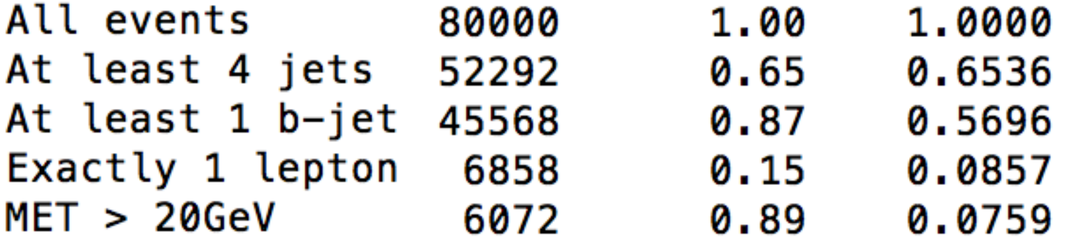


```
from heppy.analyzers.examples.ttbar.selection import Selection
selection = cfg.Analyzer(
    Selection,□
    instance_label='cuts'
)
```

From the collections defined before

1. Select events with ≥ 4 jets
2. Select events with == 1 lepton
3. Select events with ≥ 1 b-tag
4. Select events with MET > 20GeV

N evt Eff(cut-1) Eff

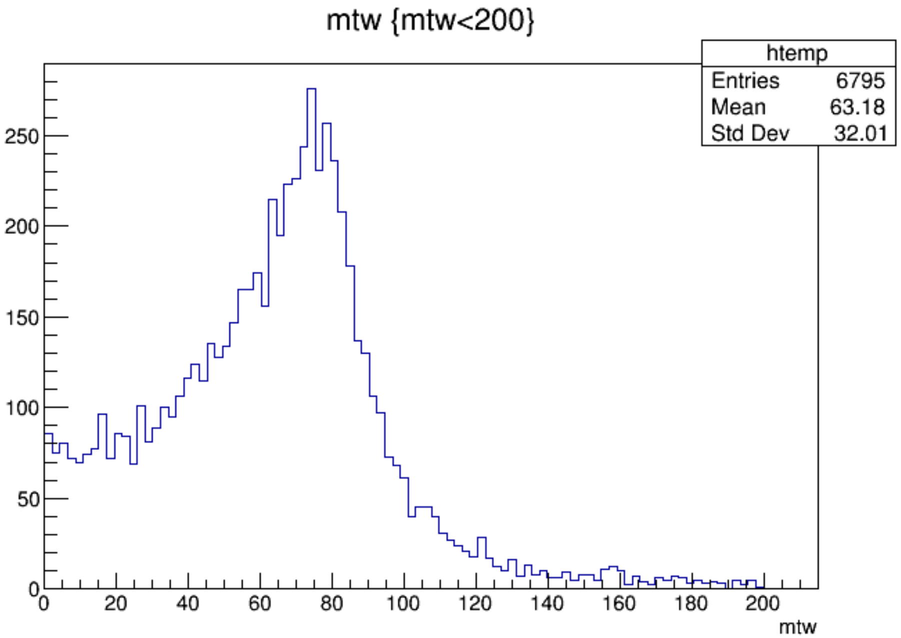


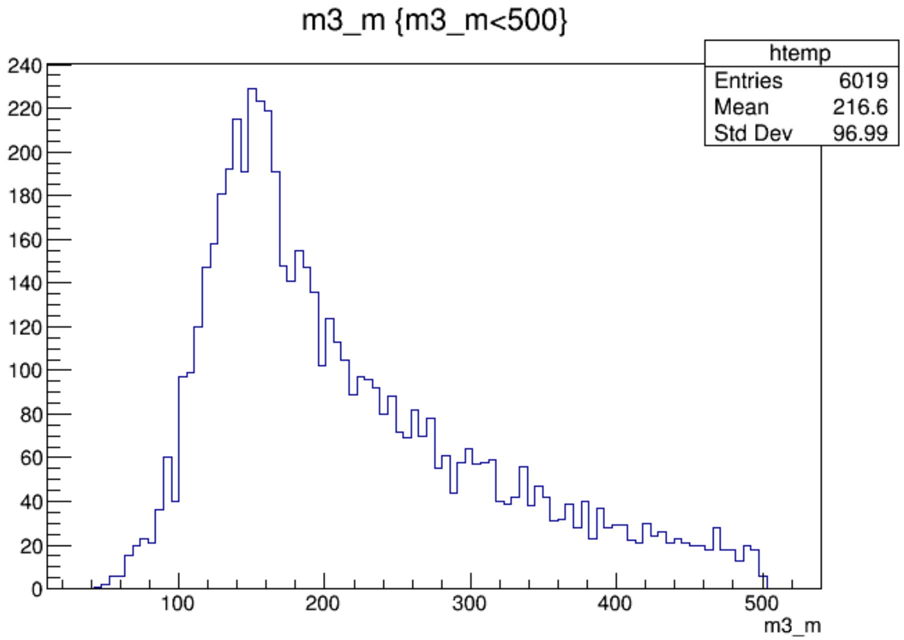
```
All events          80000      1.00    1.0000
At least 4 jets     52292      0.65    0.6536
At least 1 b-jet    45568      0.87    0.5696
Exactly 1 lepton     6858      0.15    0.0857
MET > 20GeV          6072      0.89    0.0759
```

Cut flow automatically written in text file

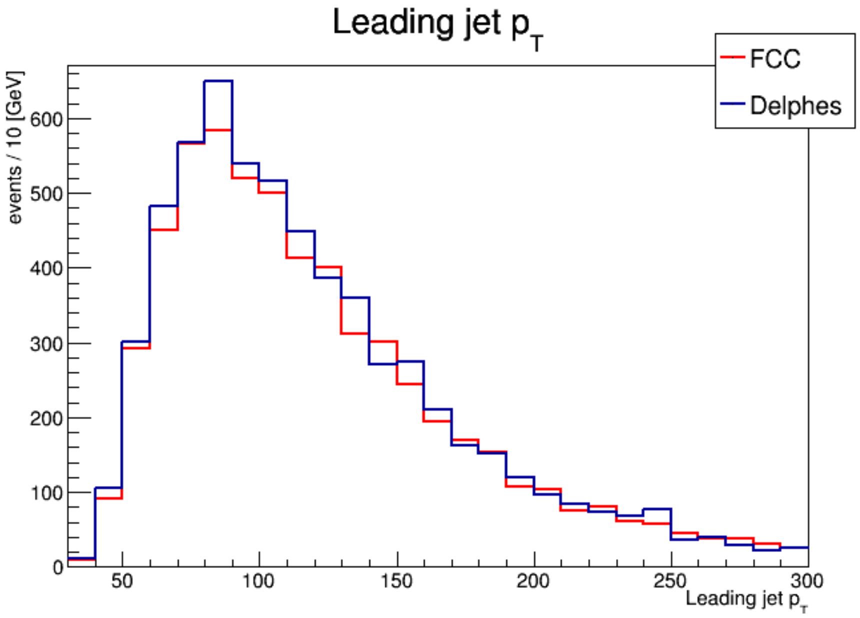
Some plots

- m_T(W) -> transverse mass of leptonic W semi leptonic event selection
  - $\surd(2*p_T^{l}*E_T^{miss} *(1-\cos\Delta(l, MET))$
- M_3-> represents the top hadronic mass



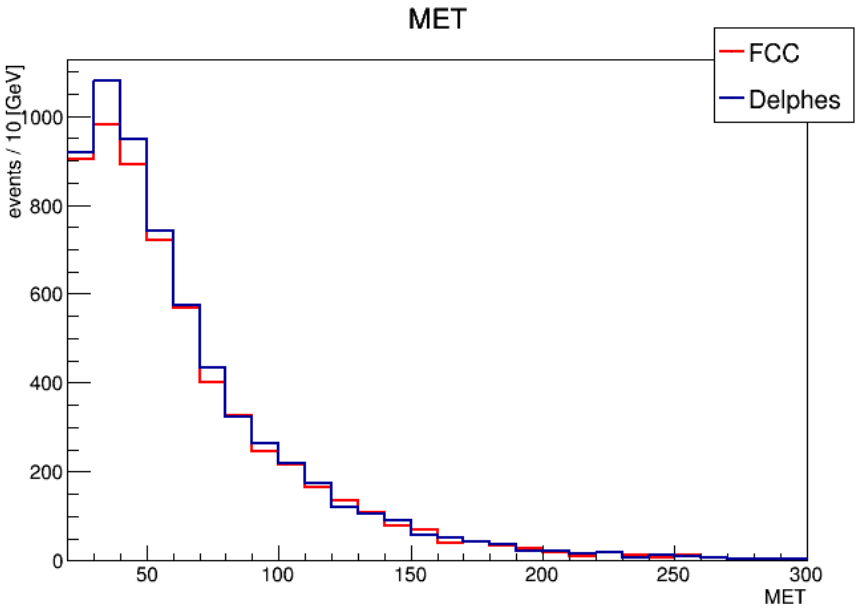


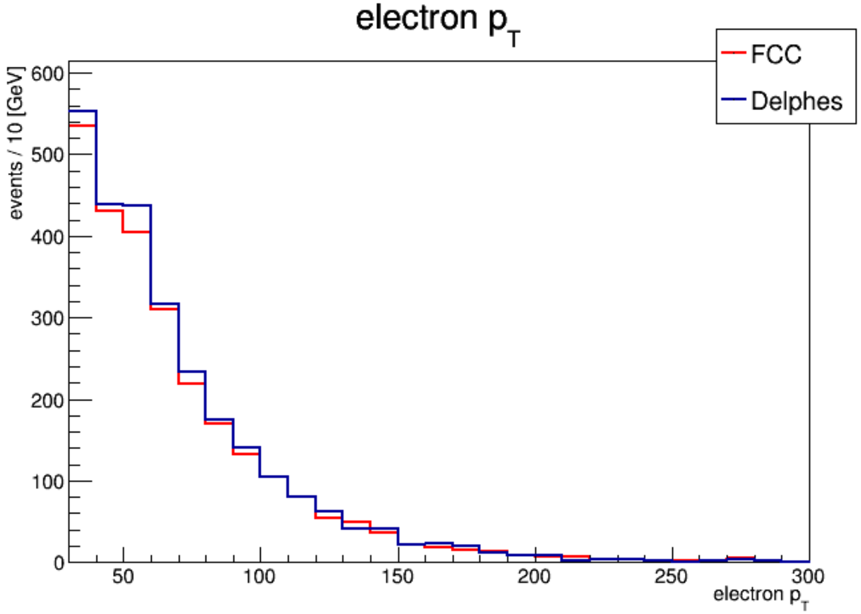
Delphes validation



- Comparing:
  - Delphes Standalone
  - Delphes within FCCSW

- Good agreement
  - Some differences due to lepton-jet overlap removal





Summary and next steps

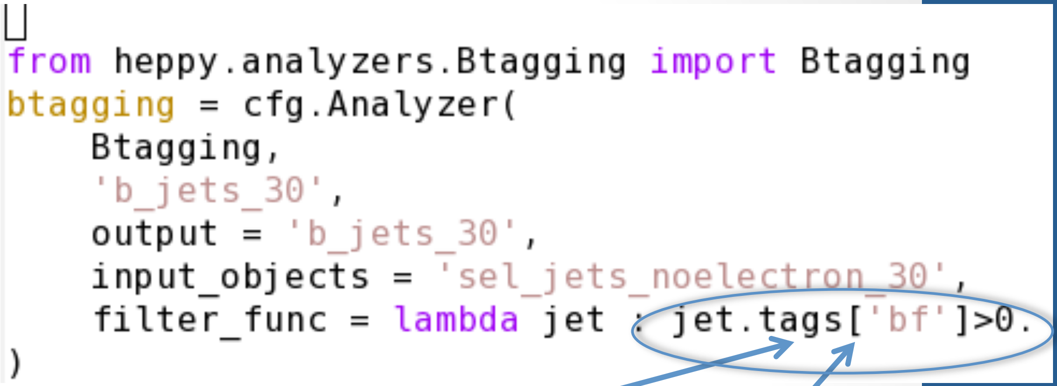
- Getting started instructions
    - https://twiki.cern.ch/twiki/bin/view/FCC/FccSoftwareGettingStarted
- Software tutorials
    - https://twiki.cern.ch/twiki/bin/viewauth/FCC/FccSoftware#Tutorials

- Analysis tools ready to be used!

- Coming soon:
- heppy batch tools from CMS
    - massive processing of eos files on lsf made easy
- Profiling for performance
    - now 30 Hz on a macbook
- Papas
    - Full FCC-ee example ZH analysis with signal, background, final fit, etc.
    - Full FCC-ee example WW→H analysis (study particle flow)
- Delphes
    - Full FCC-hh example ttbar analysis
    - Finalizing a nice and simple tutorial with ttbar as signal

Backup

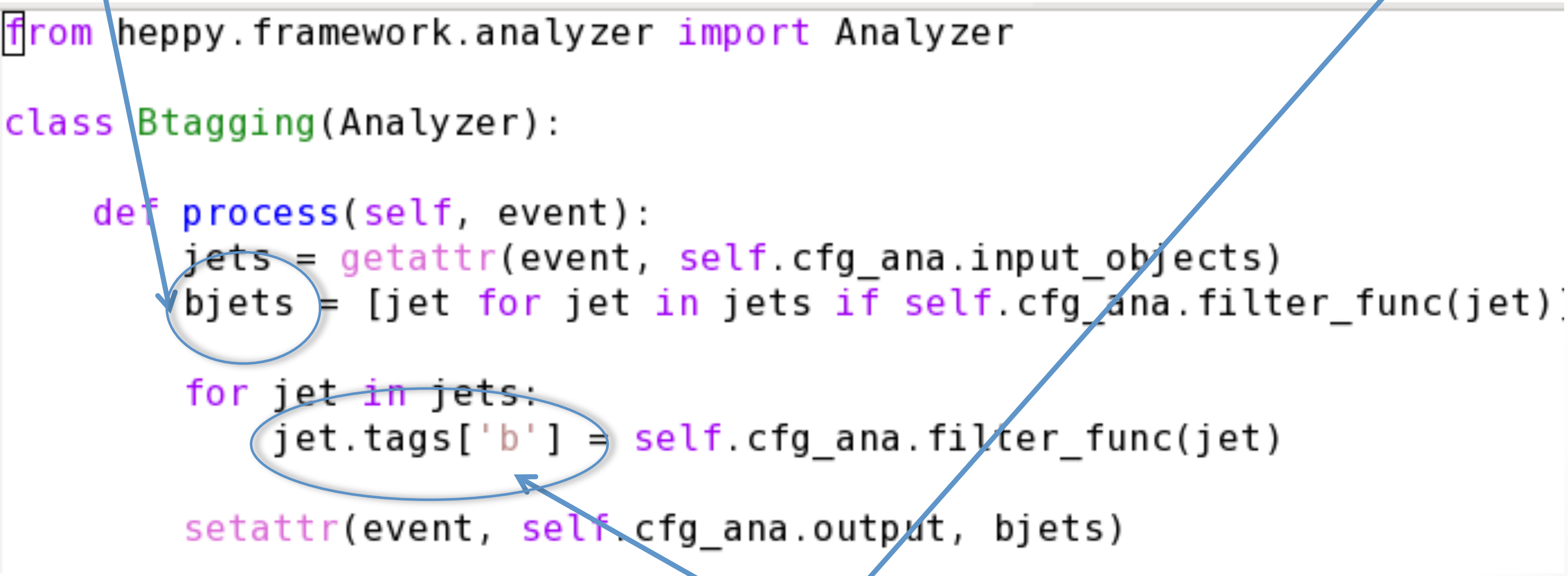
B-tagging



```
from heppy.analyzers.Btagging import Btagging
btagging = cfg.Analyzer(
    Btagging,
    'b_jets_30',
    output = 'b_jets_30',
    input_objects = 'sel_jets_noelectron_30',
    filter_func = lambda jet : jet.tags['bf']>0.
)
```

Define a collection of tag jets

B-tagging value added as a jet.tags when reading Delphes, but could also create our own algorithm

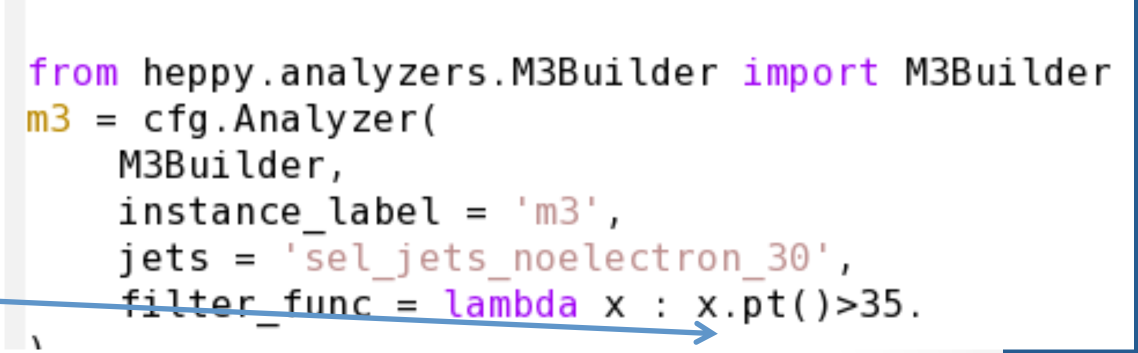


```
from heppy.framework.analyzer import Analyzer

class Btagging(Analyzer):

    def process(self, event):
        jets = getattr(event, self.cfg_ana.input_objects)
        bjets = [jet for jet in jets if self.cfg_ana.filter_func(jet)

        for jet in jets:
            jet.tags['b'] = self.cfg_ana.filter_func(jet)

        setattr(event, self.cfg_ana.output, bjets)
```

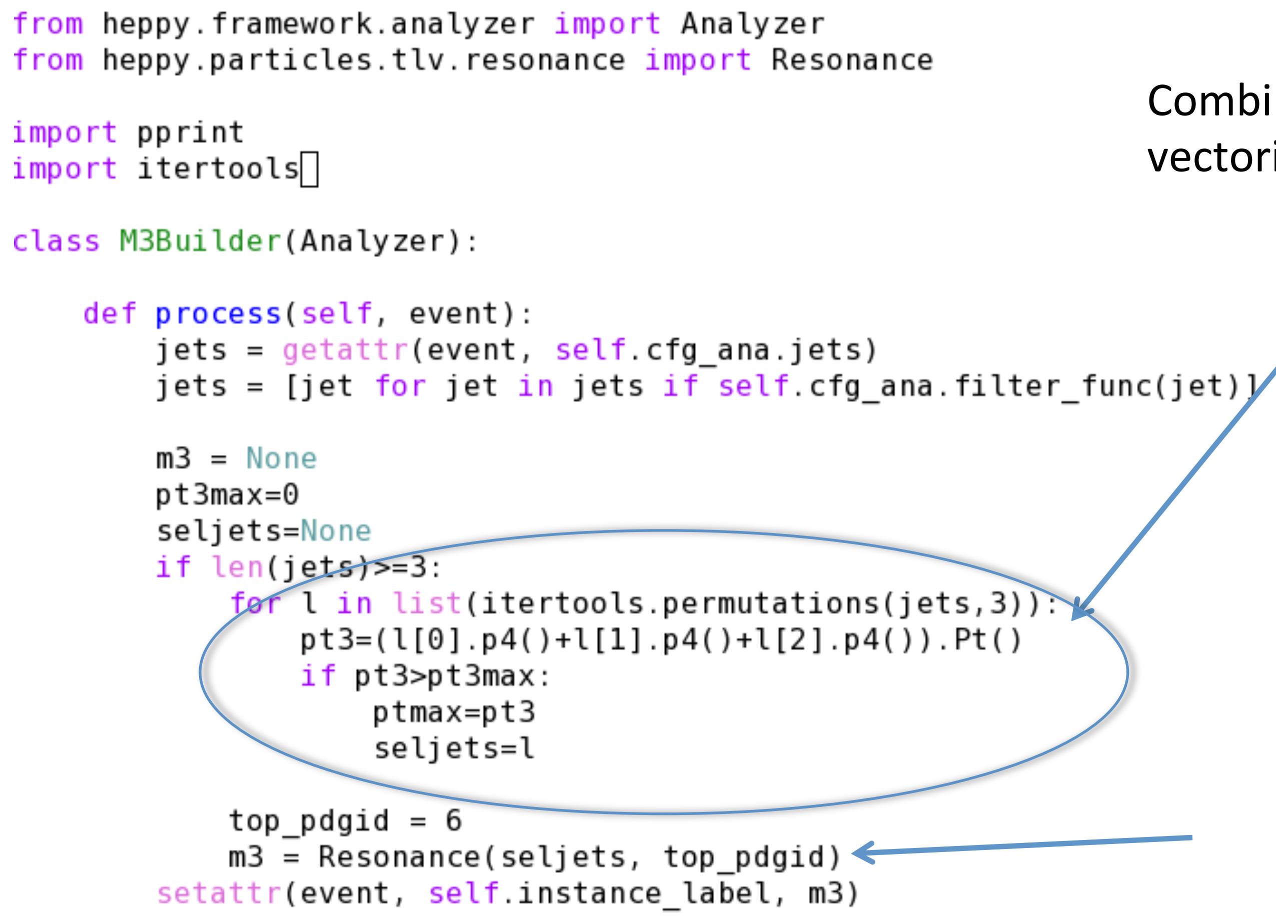
Add a new tag that is now a bool (result of the tag function

Top had mass

Build m3 with jets > 35GeV



```
from heppy.analyzers.M3Builder import M3Builder
m3 = cfg.Analyzer(
    M3Builder,
    instance_label = 'm3',
    jets = 'sel_jets_noelectron_30',
    filter_func = lambda x : x.pt()>35.
)
```



```
from heppy.framework.analyzer import Analyzer
from heppy.particles.tlv.resonance import Resonance

import pprint
import itertools□

class M3Builder(Analyzer):

    def process(self, event):
        jets = getattr(event, self.cfg_ana.jets)
        jets = [jet for jet in jets if self.cfg_ana.filter_func(jet)]

        m3 = None
        pt3max=0
        seljets=None
        if len(jets)>=3:
            for l in list(itertools.permutations(jets,3)):
                pt3=(l[0].p4()+l[1].p4()+l[2].p4()).Pt()
                if pt3>pt3max:
                    ptmax=pt3
                    seljets=l

            top_pdgid = 6
            m3 = Resonance(seljets, top_pdgid)
        setattr(event, self.instance_label, m3)
```

Combination with the vectorial sum of highest p_T

Could also veto combinations without exactly 1 b-tagged jet

Build a resonance out of the 3 selected jets