What's new in FastJet 3

http://fastjet.fr

Jets are self-aware

Jets from a cluster sequence know about properties deriving from the clustering itself

```
ClusterSequence cs(input_particles, jet_def);
vector<PseudoJet> jets = cs.inclusive_jets();
// extract the constituents of a jet
vector<PseudoJet> constituents = jets[0].constituents();


ClusterSequenceArea csa(input_particles, jet_def, area_def);
vector<PseudoJet> jets = csa.inclusive_jets();
// get the scalar area of a jet
double area = jets[0].area();
```

Composite jets

Jets can be joined

```
// subjets of a top candidate
PseudoJet W1;
PseudoJet W2;
PseudoJet b;

// build the top
PseudoJet W = join(W1,W2); // result is a ‘sensible’ PseudoJet,
PseudoJet top = join(W,b); // with additions (see pieces() below)

// top constituents: all the initial particles in the jet
vector<PseudoJet> constituents = top.constituents();

// top pieces: the b and the W
vector<PseudoJet> pieces = top.pieces();
```

UserInfoBase

Define and use more complex user info

```
// user defined class
class MyInfo : public PseudoJet::UserInfoBase {
 public:
  MyInfo(int id, int vertex) : pdg_id(id),
vertex_number(vertex){}
  int pdg_id, vertex_number; };


particle.set_user_info(new MyInfo(13,0)); // set the info


int id = particle.user_info<MyInfo>().pdg_id;// access it
```

Selectors

Cuts can be defined and applied.

New user-defined selectors are easily implemented

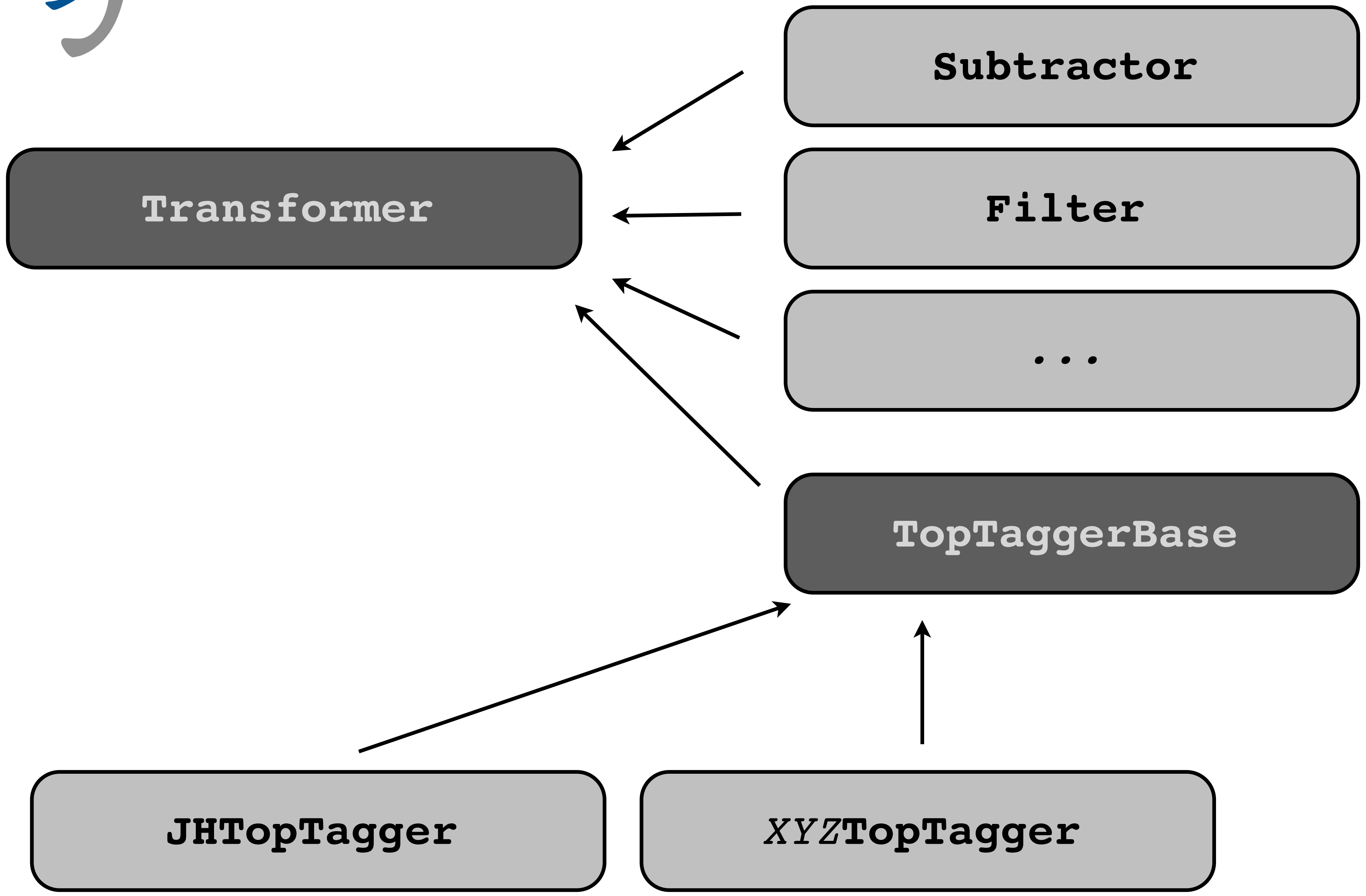
```
#include “fastjet/Selector.hh”

Selector rap_sel = SelectorAbsRapMax(1.0);     // y cut
Selector nhardest_sel = SelectorNHardest(2);  // select 2 hardest

// combine selectors using a non-commuting “and”
Selector full_sel = nhardest_sel * rap_sel;
vector<PseudoJet> two_hardest_within_cuts = full_sel(jets);

// most useful when passing (combined) cuts to functions, e.g.
Selector acceptance =
        SelectorRapRange(-3.0,-1.0) || SelectorRapRange(1.0,2.0);
GhostedAreaSpec gas(acceptance,.....); // place ghosts only where
                                       // needed
```

BackgroundEstimator

New classes to handle background estimation

```
#include “fastjet/tools/JetMedianBackgroundEstimator.hh”
#include “fastjet/tools/GridMedianBackgroundEstimator.hh”

double rapmax = 4.0;
Selector rap_sel = SelectorAbsRapMax(rapmax);  // y range

// define bkgd. est. using jets (as in FJ2)
JetMedianBackgroundEstimator bge(rap_sel,jet_def,area_def);
// alternatively, use new (and faster) GridMedian estimator
GridMedianBackgroundEstimator bge(rapmax, 0.55);

// set particles to be used in bge
bge.set_particles(input_particles);

double rho   = bge.rho();            // extract value of rho
double sigma = bge.sigma();          // extract value of sigma
```

Transformers



Subtractor

Noise removal I: background subtraction

```
#include “fastjet/tools/Subtractor.hh”

JetMedianBackgroundEstimator bge(....); // or GridMedian
bge.set_particles(input_particles);

Subtractor subtractor(&bge);        // define subtractor

ClusterSequenceArea csa(input_particles,jet_def, area_def);
vector<PseudoJet> full_jets = csa.inclusive_jets();

vector<PseudoJet> subtr_jets = subtractor(full_jets);
```

Filter

Noise removal 2: filtering/trimming

```
#include “fastjet/tools/Filter.hh”
double Rfilt = 0.3;

// filtering
Selector sel_3hardest = SelectorNHardest(3);
Filter filter(Rfilt,sel_3hardest);
PseudoJet filtered_jet = filter(jet);

// trimming
Selector sel_aboveptfrac = SelectorPtFractionMin(0.05);
Filter trimmer(Rfilt,sel_aboveptfrac);
PseudoJet trimmed_jet = trimmer(jet);

// structure of result
vector<PseudoJet> kept_subjets = filtered_jet.pieces();
vector<PseudoJet> rejected_subjets =
                  filtered_jet.structure_of<Filter>().rejected();
```

Pruner

Noise removal 3: pruning

```
#include “fastjet/tools/Pruner.hh”

double zcut = 0.1;
double rcut_factor = 0.5;

Pruner pruner(cambridge_algorithm, zcut, rcut_factor);
PseudoJet pruned_jet = pruner(jet);

// pruned-away subjets
vector<PseudoJet> rejected_subjets =
                  pruned_jet.structure_of<Pruner>().rejected();
```

Taggers

Example of a boosted top tagger

```
#include “fastjet/tools/JHTopTagger.hh”

double delta_pt=0.1, delta_r=0.19;
JHTopTagger = top_tagger(delta_pt,delta_r);
top_tagger.set_top_selector(SelectorMassRange(150,200));
top_tagger.set_W_selector(SelectorMassRange(65,95));

PseudoJet tagged_jet = top_tagger(jet);    // top candidate

// extract structure
if (tagged_jet != 0) {
 PseudoJet W = tagged_jet.structure_of<JHTopTagger>().W();
 PseudoJet nonW = tagged_jet.structure_of<JHTopTagger>().non_W();
}
```