MC4BSM
Beijing - May 2016

FastJet

Matteo Cacciari
LPTHE Paris
Université Paris Diderot

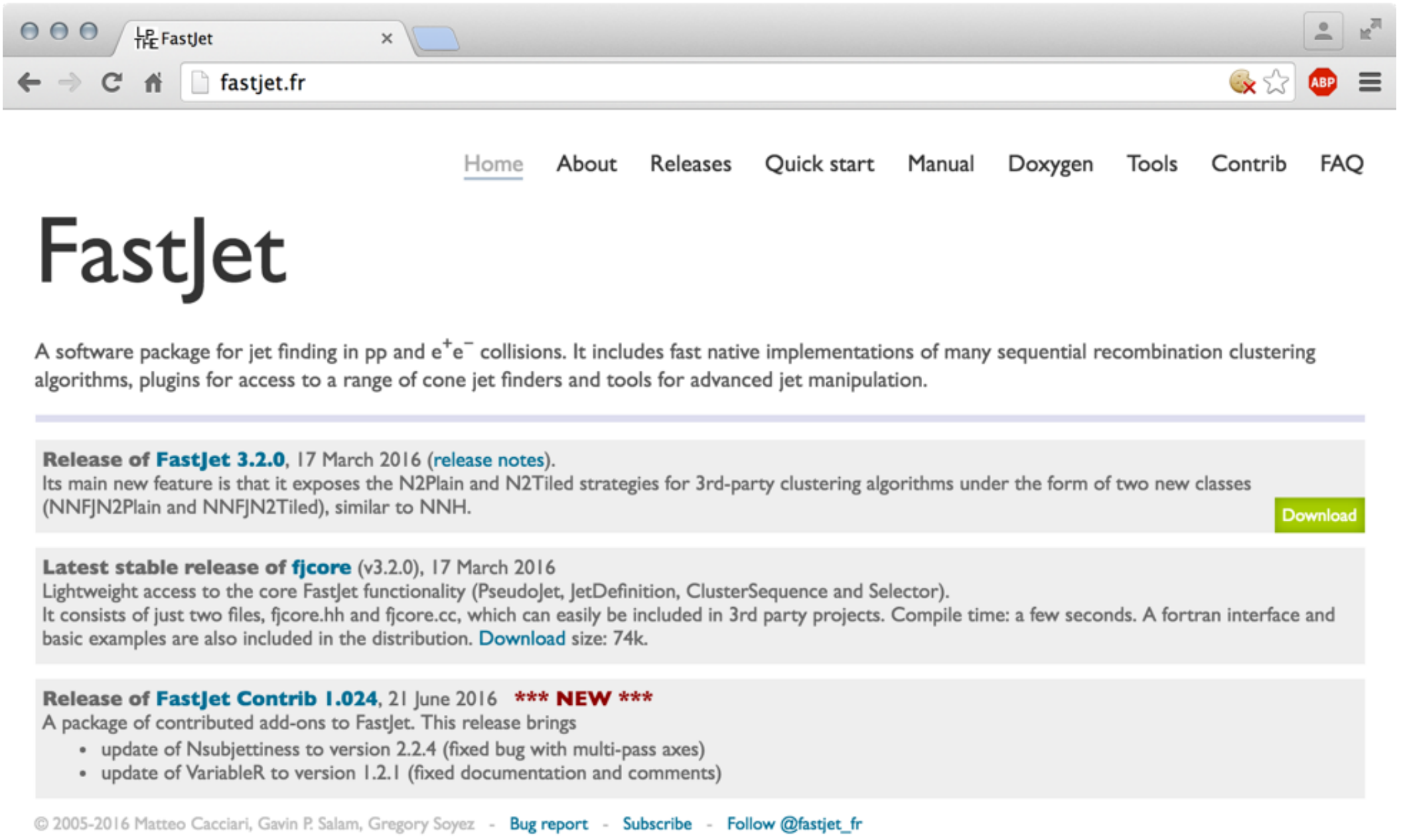
In collaboration with Gavin Salam and Gregory Soyez

Summary (from MC4BSM14 in Daejeon)

- **FastJet** is a C++ library for performing jet clustering and a number of other jet-related tasks
  - It includes tools for basic clustering, tagging, grooming, pileup subtraction, usually implemented with an eye to user-friendliness
  - Used by all LHC experiments and a number of phenomenological projects, it is available under GPLv2 license at http://fastjet.fr

- A sister project, **FastJet Contrib**, has been started in February 2013
  - It is hosted by HepForge at http://fastjet.hepforge.org/contrib
  - It aims at collecting, under a common build framework, jet tools that make use of FastJet
  - It is open to all third-party authors

http://fastjet.fr

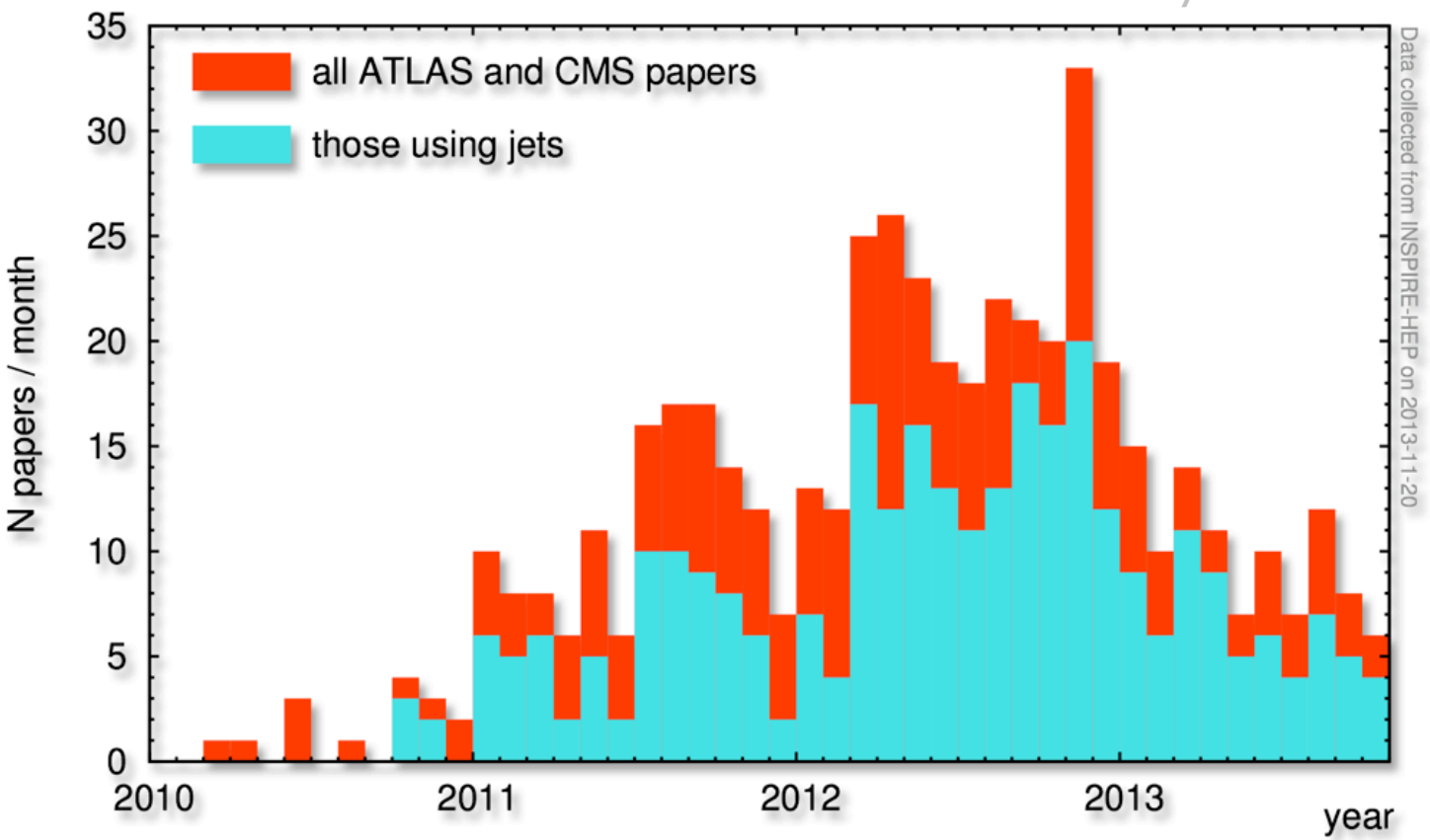


Authors: MC, Gavin Salam, Gregory Soyez

The pervasiveness of jets

- ATLAS and CMS have each published **300+** papers between 2010 and 2014
  - More than **a third** of these papers make use of **jets**
  - **60%** of the **searches** papers makes use of **jets**

Plot by G. Salam

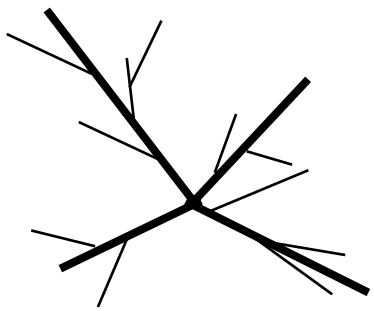


(Source: INSPIRE. Results may vary when employing different search keywords)

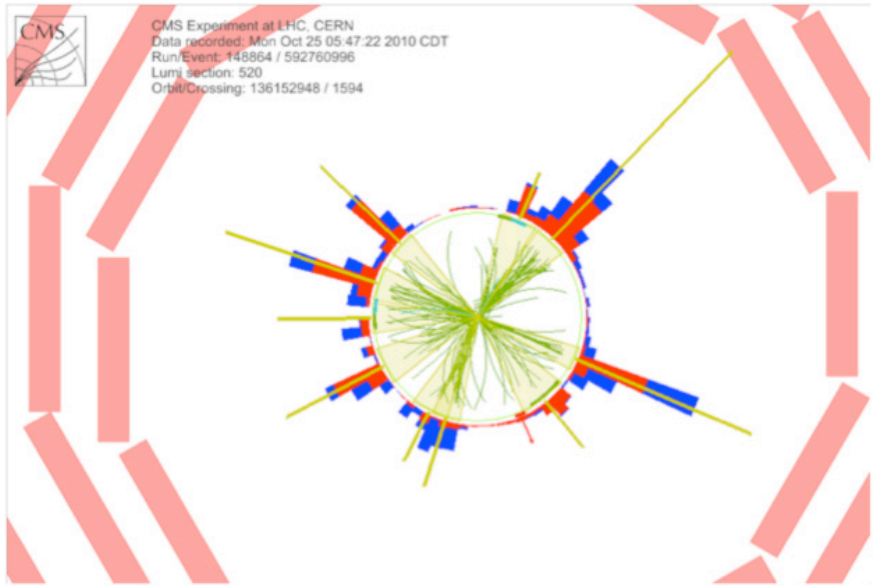
Why are jets so important?

Taming reality

Multileg + PS



??



QCD predictions

Real data

Jets

One purpose of a 'jet clustering' algorithm is to
reduce the complexity of the final state, simplifying many hadrons
to **simpler objects** that one can hope to **calculate**

FastJet's aims

FastJet tries to make jet clustering very **user-friendly**, **infrared and collinear safe**, and very **fast**

Suppose you are a BSM person....

What can FastJet do for you?

FastJet is a low-level library used as the main jet clustering code by all LHC experiments and by a number of packages (aMC@NLO, PYTHIA 8, Delphes, ...)

Chances are that, even if you didn't realize it, you are already using it

If you run one of these codes, and only need basic jet clustering, you are unlikely to need more

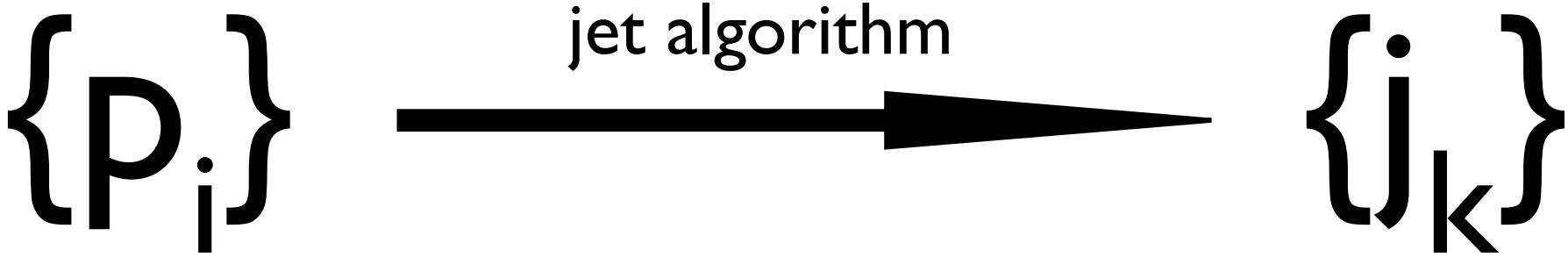
Jets can serve **two** purposes

- They can be ***observables***, that one can calculate and measure
- They can be ***tools***, that one can employ to extract specific properties of the final state

FastJet implements numerous tools
for both these tasks

Jet clustering algorithm

A **jet algorithm** maps the momenta of the final state particles into the momenta of a certain number of jets:



particles,
4-momenta,
calorimeter towers, ...

jets

Most algorithms contain a resolution parameter, **R**, which controls the extension of the jet

The k_t algorithm and its siblings

Sequential recombination algorithms,
based on iterative recombination according to the distance measures

$$d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p}) \frac{\Delta y^2 + \Delta\phi^2}{R^2} \qquad d_{iB} = p_{ti}^{2p}$$

p = 1 k_t algorithm

S. Catani, Y. Dokshitzer, M. Seymour and B. Webber, Nucl. Phys. B406 (1993) 187
S.D. Ellis and D.E. Soper, Phys. Rev. D48 (1993) 3160

p = 0 Cambridge/Aachen algorithm

Y. Dokshitzer, G. Leder, S.Moretti and B. Webber, JHEP 08 (1997) 001
M. Wobisch and T. Wengler, hep-ph/9907280

p = -1 **anti-k_t algorithm**

MC, G. Salam and G. Soyez, arXiv:0802.1189

NB: in anti-kt pairs with a **hard** particle will cluster first: if no other hard particles are close by, the algorithm will give **perfect cones**

Jet Definition

A jet algorithm
+
its parameters (e.g. R)
+
a recombination scheme
=
a **Jet Definition**

"*Jet [definitions] are legal contracts between theorists and experimentalists*"
-- MJ Tannenbaum

Jet clustering in FastJet

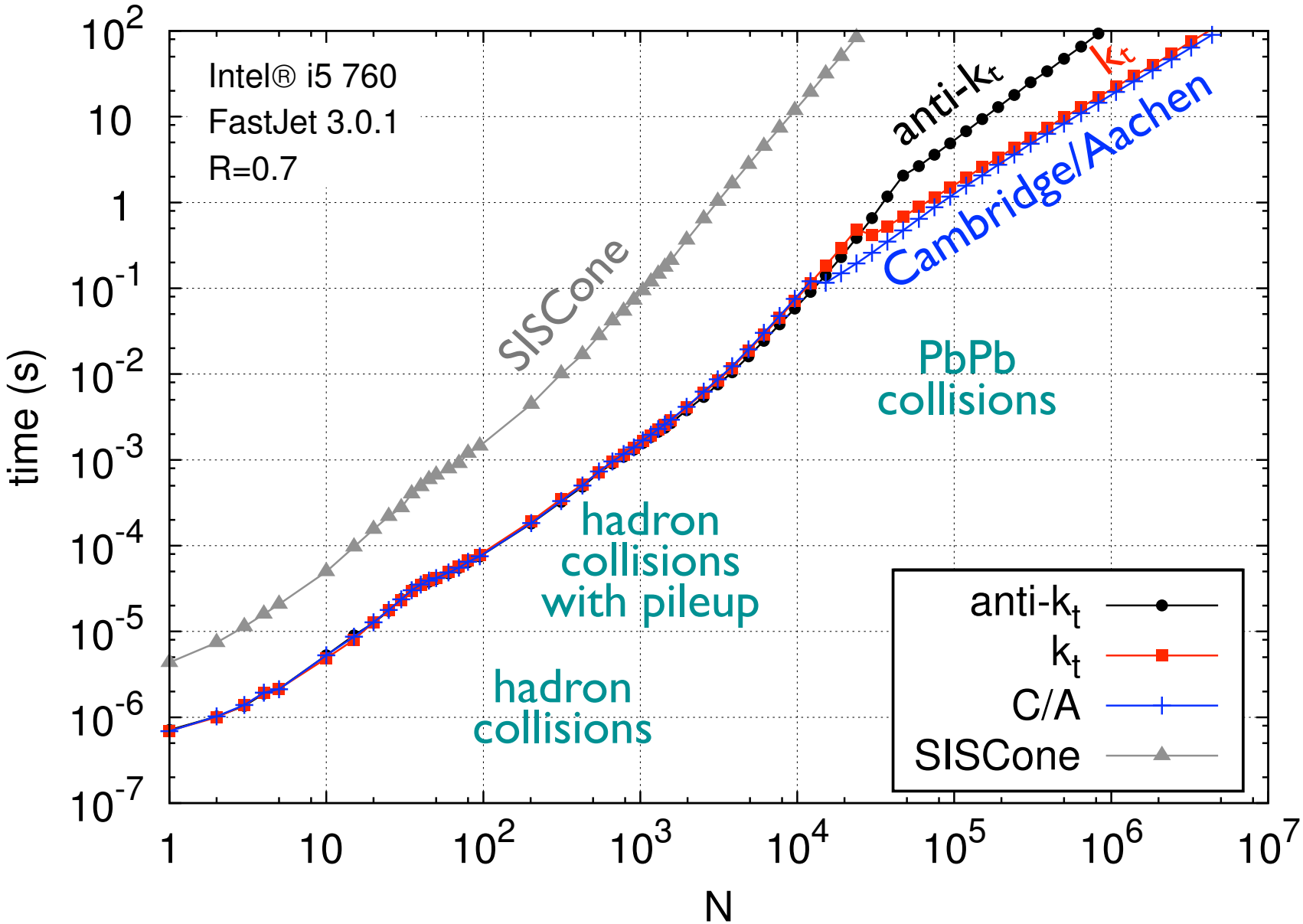
```
/// define a jet definition
JetDefinition jet_def(JetAlgorithm jet_algorithm,
                      double R,
                      RecombinationScheme rec_sch = E_scheme);
```

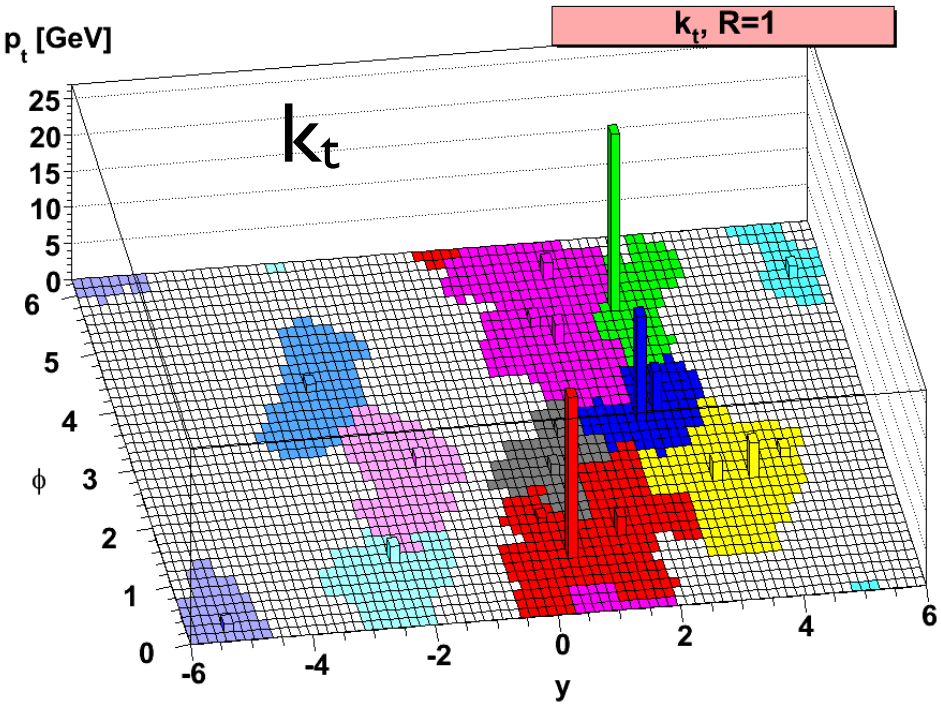
jet_algorithm can be any one of the four IRC safe algorithms, or also most of the old IRC-unsafe ones, for legacy purposes

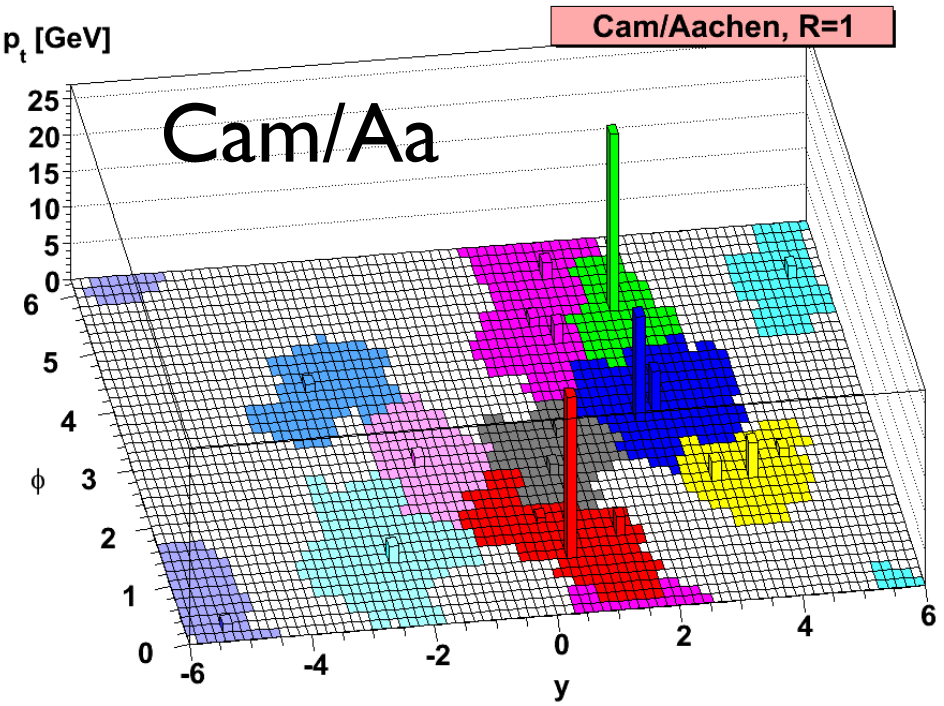
```
/// create a ClusterSequence, extract the jets
ClusterSequence cs(input_particles, jet_def);
vector<PseudoJet> jets = sorted_by_pt(cs.inclusive(jets));
...
// pt of hardest jet
double pt_hardest = jets[0].pt();
...
// constituents of hardest jet
vector<PseudoJet> constits = jets[0].constituents();
```

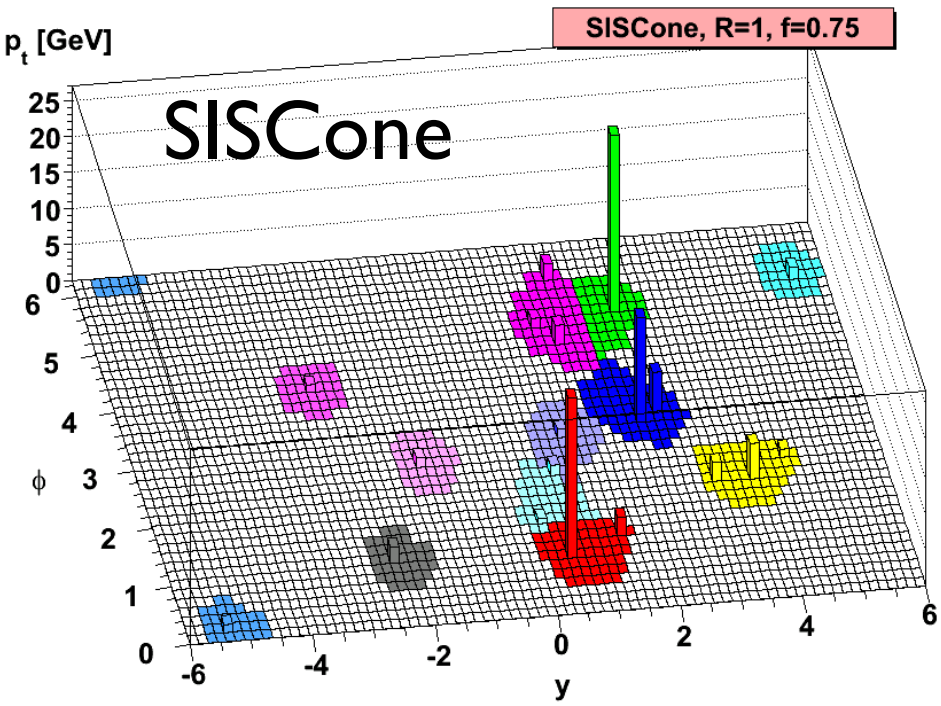
FastJet speed test

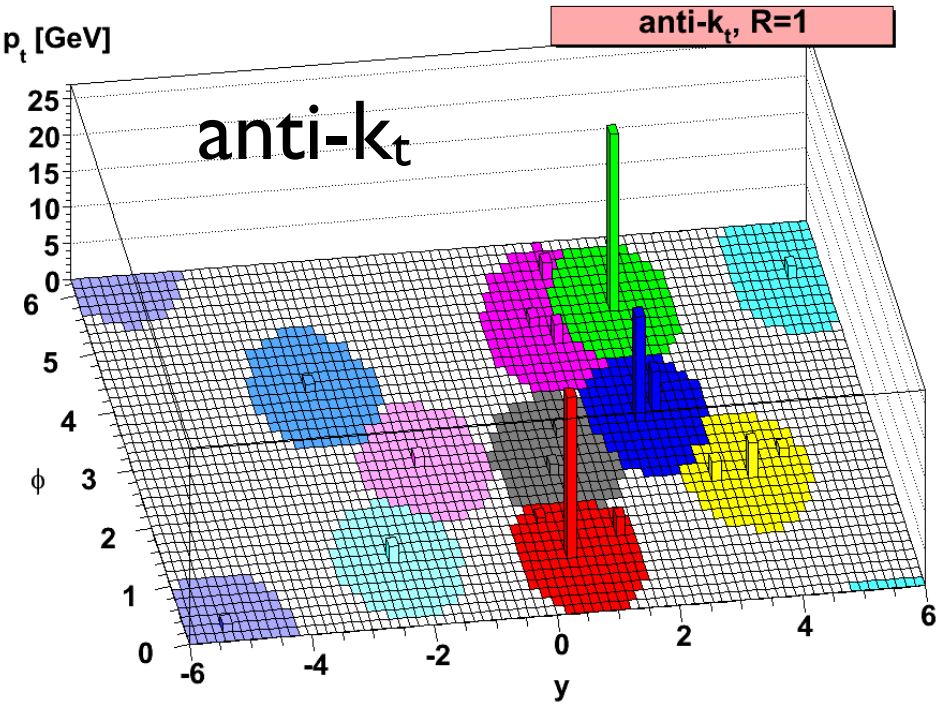
Time needed to cluster an event with N particles




k_t, R=1
p_t [GeV]
k_t
y
ϕ


Cam/Aachen, R=1
p_t [GeV]
Cam/Aa
y
ϕ


SISCone, R=1, f=0.75
p_t [GeV]
SISCone
y
ϕ


anti-k_t, R=1
p_t [GeV]
anti-k_t
y
ϕ

A word on how to include FastJet

FastJet is a C++ library

The basic, and recommended, usage mode is to install it and link to it

- All functionalities are available
- FJ updates are easily included

Other options:

- A standalone fjcore.[cc|hh] small file is provided for direct inclusion into projects
  - It contains most the main functionalities of FastJet
  - It is designed so as not to interfere with linking with a full FJ version
  - It is presently used by (at least) PYTHIA8 and aMC@NLO
- Copy the FJ files into your package **[deprecated!]**
  - FJ cannot be easily updated
  - Will likely interfere with a link to a different FJ version

PS. Whatever you do, if you are maintaining a public package that uses FastJet please print out the FJ banner where the users can see it

Beyond clustering

Beyond finding jets, FastJet is used for
at least two kinds of operations performed on jets

- Background subtraction

  Subtraction of contamination due to Underlying Event and/or pile-up

- Substructure Analysis

  Studying internal structure of jets in order to determine their origin

Background subtraction

Two main approaches to background subtraction

- Jet-based
  - Cluster the **full event**, determine the event-specific (ρ) and jet-specific (A) quantities, and subtract the relevant contamination from a given observable
    - Pros: largely unbiased subtraction
    - Cons: slow, potentially large(er) residual uncertainty
    - Examples: `jet area/median' in FastJet, GenericSubtractor for jet shapes, JetFFMoments for fragmentation functions, ...
- Particle-based
  - Produce a **reduced event**, by dropping some of the particles. Cluster this reduced event, and calculate from it the observables
    - Pros: fast, often small(er) residual uncertainty
    - Not natively unbiased, can depend on choice of parameters
    - Examples: ConstituentSubtractor, SoftKiller, PUPPI, ...

Background determination

Jet algorithms like k_t or Cambridge/Aachen allow one to determine
on an event-by-event basis
the **"typical" level of transverse momentum density**
of a **roughly uniform background noise**:

$$\rho \equiv \underset{\text{(over a single event)}}{\text{median}} \left[\left\{ \frac{p_t^{jet}}{\text{Area}_{jet}} \right\} \right]$$

MC, Salam, 2007

This ρ value can, in turn, be used to characterise the UE

Since this measurement is done with the jets, it is alternative/complementary to the usual analyses done using charged tracks (à la R. Field)

Background subtraction

Once ρ has been measured, it can be used to **correct** the transverse momentum of the hard jets:

$$p_T^{\text{hard jet, corrected}} = p_T^{\text{hard jet, raw}} - \rho \times \text{Area}_{\text{hard jet}}$$

ρ being measured on an **event-by-event** basis, and each jet subtracted **individually**, this procedure will remove many fluctuations and generally **improve the resolution** of, say, a mass peak

$$\Delta p_t = \rho A \pm (\sigma\sqrt{A} + \sigma_\rho A + \rho\sqrt{\langle A^2\rangle - \langle A\rangle^2}) + \Delta p_t^{BR}$$

NB. Also be(a)ware of ***backreaction***

Background estimation and subtraction

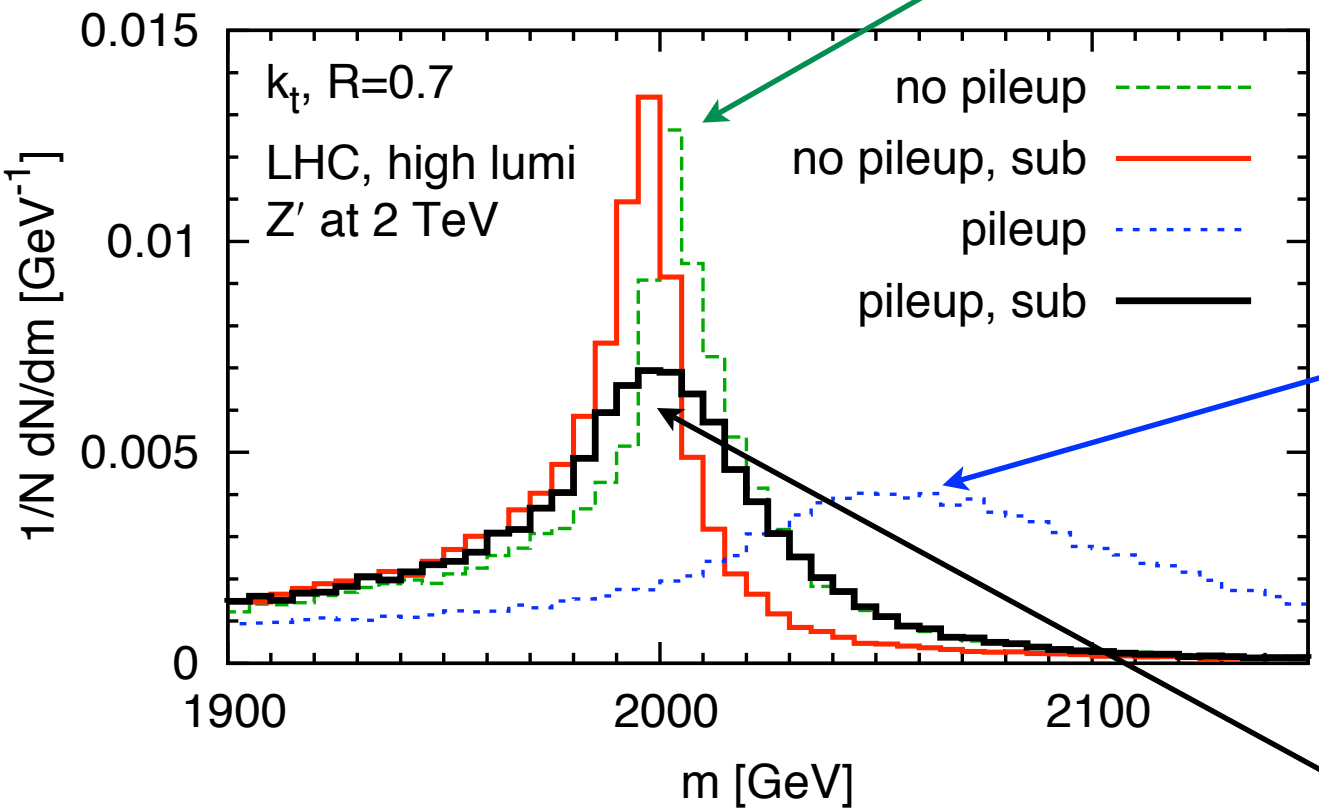
```
// constructor for a background estimator
JetMedianBackgroundEstimator bge(Selector sel,
                                 JetDefinition jet_def,
                                 AreaDefinition area_def);

// an alternative (faster) background estimator
//GridMedianBackgroundEstimator bge(Selector sel, grid_step);

bge.set_particles(input_particles);
....
double rho = bge.rho(jet);   // extract rho estimation
```

```
// define a subtractor
Subtractor sub(&bge);

// apply it to a jet (or a vector of jets)
PseudoJet subtracted_jet = sub(jet);
```

Example of pileup subtraction

Let's discover a leptophobic Z' and measure its mass:

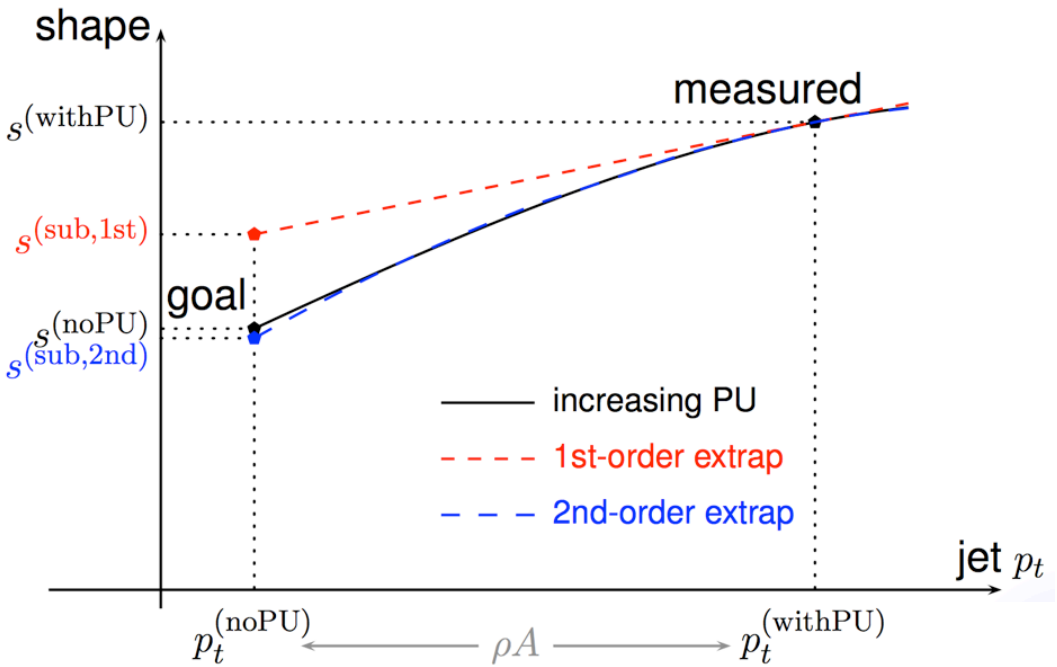
MC simulation:
m = 2000 GeV, width ~ 10 GeV



Naive measurement with PU:
m ~ 2050 GeV, width ~ 60 GeV

Measurement after subtraction:
m ~ 2000 GeV, width ~ 25 GeV

Numerical jet shape correction



Soyez et al. 1211.2811

A generic **jet shape** (a function of the momenta of all constituents of a jet) is modified by the addition of pileup

Correct it by calculating numerically the derivatives that enter its Taylor expansion and subtracting (this generalises the jet area/median subtraction for transverse mom.)

Pileup momentum density

Numerical derivative w.r.t. ghosts momenta

$$V_{\text{jet,sub}} = V_{\text{jet}} - \rho V_{\text{jet}}^{[1]} + \frac{1}{2}\rho^2 V_{\text{jet}}^{[2]} + \cdots$$

Shape subtraction

Pilup subtraction from jet shapes using GenericSubtractor from fjcontrib

```
#include "ExampleShapes.hh"
#include "GenericSubtractor.hh"

// define a specific jet shape
contrib::Angularity ang(1.0); // angularity with alpha=1.0

// define a generic subtractor
... construct a background estimator bge_rho....
contrib::GenericSubtractor gen_sub(&bge_rho);

// compute the subtracted shape
double subtracted_ang = gen_sub(ang, jet);
```

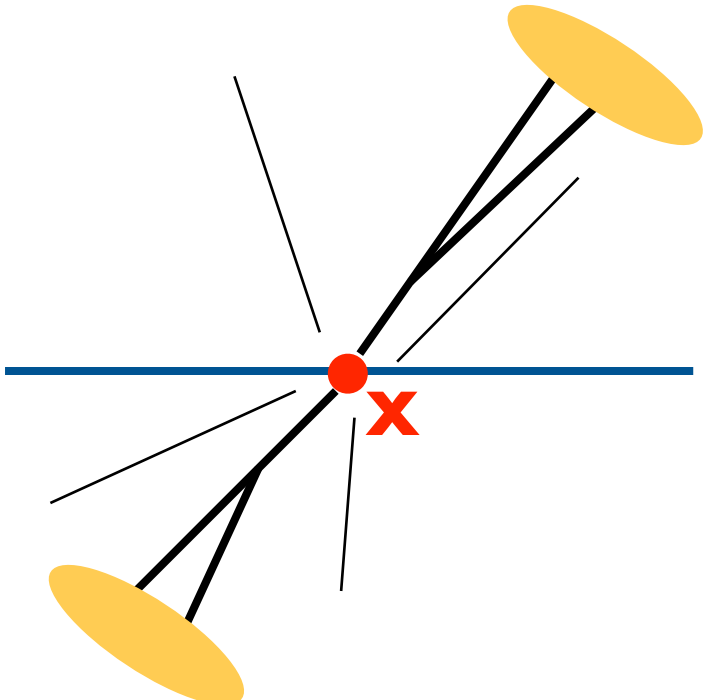
Particle-level pilup removal

Pilup removal using SoftKiller from fjcontrib

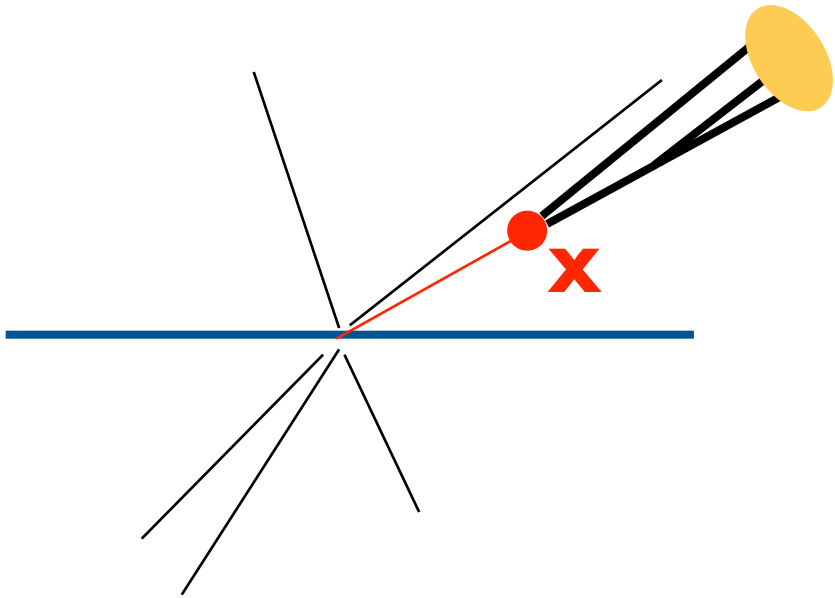
(SoftKiller progressively removes soft particles until ρ of event is zero)

```
#include "SoftKiller.hh"

// define SoftKiller
double grid_size = 0.4;
contrib::SoftKiller soft_killer(rapmax, grid_size);

// apply it to the full event
double pt_thresh //returns threshold for killed particles
vector<PseudoJet> soft_killed_event;
soft_killer.apply(full_event, soft_killed_event, pt_thresh);

// proceed with clustering and calculating shapes with
// the reduced soft_killed_event
ClusterSequence(soft_killed_event,.....)
```

Why boosted objects



Heavy particle X at **rest**



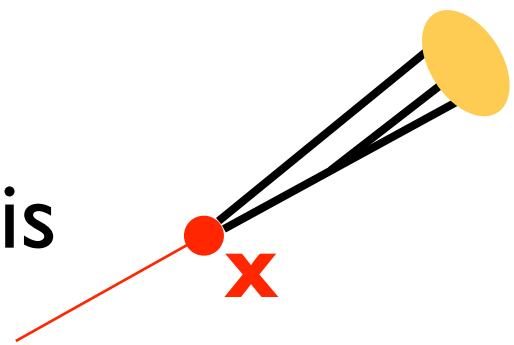
Boosted heavy particle X

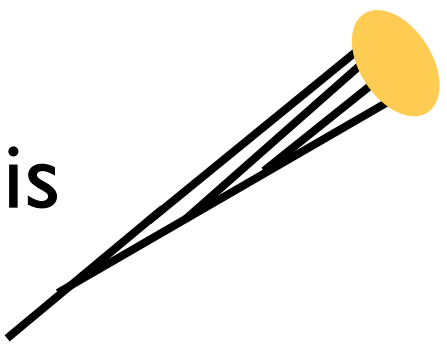
Easy to resolve jets and calculate invariant mass, but signal very likely swamped by background (eg H→bb v. tt →WbWb)

Cross section very much reduced, but acceptance better and some backgrounds smaller/ reducible

Boosted jets

When a jet is produced by a boosted heavy particle, all decay products are collimated into a small cone

How to tell this 

from this  ?

Decay of a heavy (boosted) object

Light parton fragmentation

Jet substructure

- The substructure of a jet (i.e. the ability to further resolve smaller components) can be exploited to
  - ***tag*** *a particular structure inside the jet, i.e. a massive particle*
    - *Examples: Higgs (2-prongs decay), top (3-prongs decay)*
  - *remove background contamination from the jet or its components, while keeping the bulk of the perturbative radiation*
    - *Examples: filtering, trimming, pruning*
    - *This operation if often generically denoted as* ***grooming***

(Boosted) jet studies at the LHC

Lily Asquith, summary talk at BOOST 2015

Boost is about:

1. Tagging high pT objects (SM and BSM)
2. Improving measurements (pileup, mass resolution etc)

ATLAS and CMS have taken different approaches to these things from day one.

ATLAS:
AKT4 CA12 split-fitered (BDRS)
AKT10 trimmed (R3/R2)
N-subjettiness WTA
JVT/ ρ
D2

CMS:
AKT5 CA8 pruned (p510)
CA15 HTT
N-subjettiness one-pass
Puppi
Soft drop

Essentially none of these tools existed as late as seven years ago

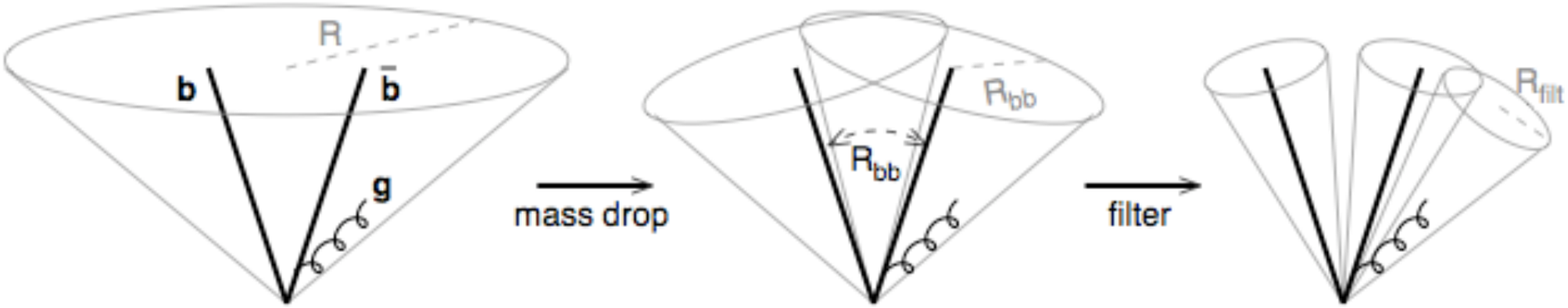
Glossary

What	i.e.	When	Ref.
AKT	Anti-kt algorithm	2008	0802.1189
CA	Cambridge/Aachen algorithm	1999	9907280
BDRS	mass-drop tagger, includes filtering	2008	0802.2470
trimmed	Trimming, tagger/groomer	2009	0912.1342
pruned	Pruning, tagger/groomer	2009	0903.5081
HTT	HepTopTagger	2009	0910.5472
N-subjettiness	jet shape function, used in tagging	2010	1011.2268
WTA	Winner-Take-All (recombination scheme)	2013	1310.7584
one-pass	choice of axis for N-subjettiness	2010	
JVT	Jet Vertex Tagger (used in pileup subtr.)	2014	
ρ	background density (used in pileup subtr.)	2007	0707.1378
D2	jet shape function, used in tagging	2014	1409.6298
PUPPI	particle-by-particle pileup subtr.	2014	1407.6013
Soft Drop	tagger/groomer	2014	1402.2657

$pp \to ZH \to \nu\bar{\nu}b\bar{b}$

The BDRS tagger

Butterworth, Davison, Rubin, Salam, 2008



It's a two-prongs tagger for boosted Higgs, which

- Uses the **Cambridge/Aachen** algorithm
- Employs a **Mass-Drop** condition (as well as an asymmetry cut) to find the relevant splitting (i.e. 'tag' the heavy particle)
- Includes a post-processing step, using '**filtering**' (introduced in the same paper) to clean as much as possible the resulting jets of UE contamination

BDRS in FastJet

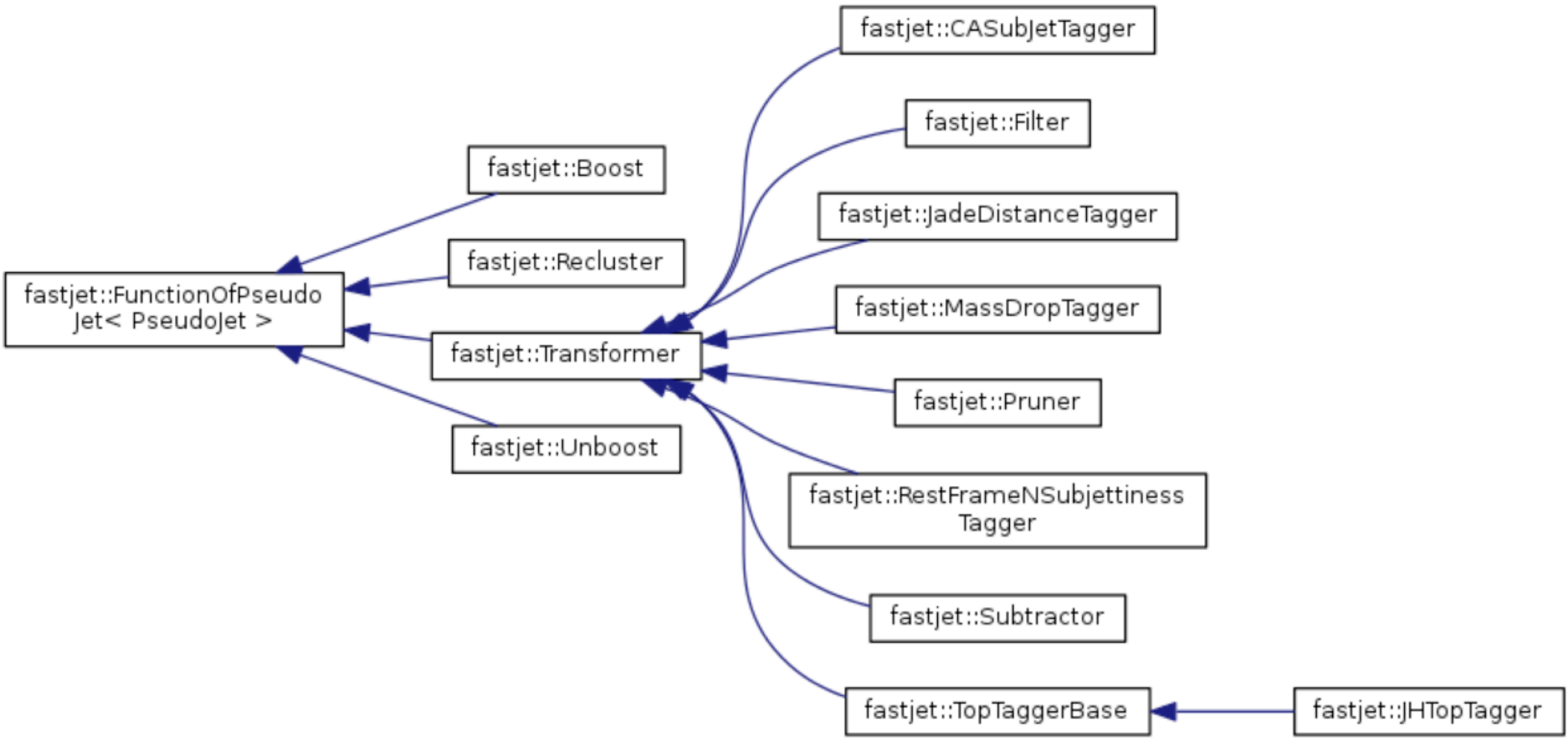
In FastJet

```
#include "fastjet/tools/MassDropTagger.hh"
#include "fastjet/tools/Filter.hh"

JetDefinition jet_def(cambridge_algorithm, 1.2);
ClusterSequence cs(input_particles, jet_def);

// define the tagger and use it
MassDropTagger md_tagger(0.667, 0.09);
PseudoJet tagged = md_tagger(jets[0]);

// define the filter and use it
Filter filter(0.3,SelectorNHardest(3));
Pseudojet higgs = filter(tagged);     // this is the Higgs!!
```

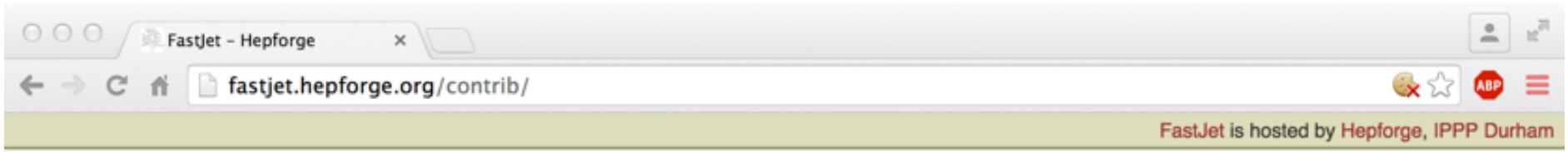
The real analysis is slightly more refined (b-tagging, dynamical filter radius, etc) but the main features are already present here

FastJet tools

Due to historical reasons, some of these tools (typically the early ones) are available directly in FastJet, others from the separate FastJet Contrib project



FastJet Contrib



- fastjet.fr
- fastjet-contrib
- contrib list
- contrib svn

FastJet Contrib

The fastjet-contrib space is intended to provide a common location for access to 3rd party extensions of FastJet.

Download the current version: fjcontrib-1.024 (released 21 June 2016), which contains these contributions. Changes relative to earlier versions are briefly described in the NEWS file.

After downloading and unpacking, enter the fjcontrib-1.024/ directory and then run

```
./configure [--fastjet-config=FILE] [--prefix=...] [...]
make
make check          # optional
make install
```

By default the package installs to the same directories as the FastJet installation.

A contribution named "SomeContrib" is usually accessed by including "fastjet/contrib/SomeContrib.hh" in your C++ file, and linking with -lSomeContrib.

Developers who wish to develop their own contribution or submit new ones should use svn to checkout the contrib framework,

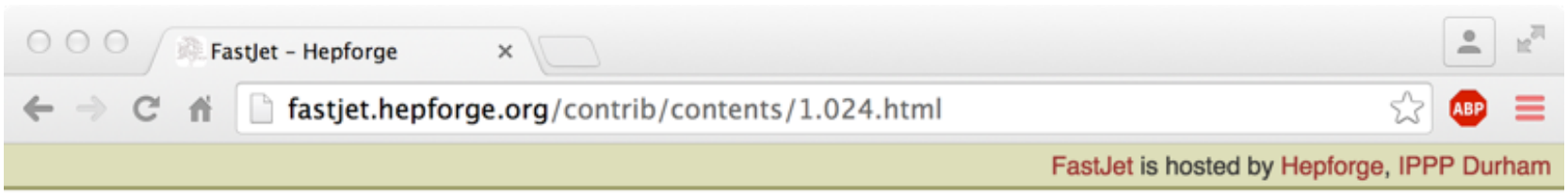
```
svn checkout http://fastjet.hepforge.org/svn/contrib/trunk fjcontrib
```

then run a local script to get the current set of individual contribs

```
cd fjcontrib/
scripts/update-contribs.sh
```

and follow the instructions in the README and DEVEL-GUIDELINES files.

FastJet Contrib



- fastjet.fr
- fastjet-contrib
- contrib list
- contrib svn

Version 1.024 of FastJet Contrib is distributed with the following packages

Package	Version	Release date	Information
ClusteringVetoPlugin	1.0.0	2015-05-04	README NEWS
ConstituentSubtractor	1.1.1	2016-02-09	README NEWS
EnergyCorrelator	1.1.0	2014-11-21	README NEWS
GenericSubtractor	1.3.1	2016-03-30	README NEWS
JetCleanser	1.0.1	2014-08-16	README NEWS
JetFFMoments	1.0.0	2013-02-07	README NEWS
JetsWithoutJets	1.0.0	2014-02-22	README NEWS
Nsubjettiness	2.2.4	2016-06-14	README NEWS
QCDAwarePlugin	1.0.0	2015-10-08	README NEWS
RecursiveTools	1.0.0	2014-07-09	README NEWS
ScJet	1.1.0	2013-06-03	README NEWS
SoftKiller	1.0.0	2014-08-17	README NEWS
SubjetCounting	1.0.1	2013-09-03	README NEWS
ValenciaPlugin	2.0.0	2015-02-24	README NEWS
VariableR	1.2.1	2016-06-01	README NEWS

15 external contributions

Alternative tagger: N-subjettiness

Thaler, van Tilburg, 2010

$$\tau_N^{(\beta)} = \sum_i p_{Ti} \min \left\{ R_{1,i}^{\beta}, R_{2,i}^{\beta}, \ldots, R_{N,i}^{\beta} \right\}$$

Sum over constituents of a jet

Distances to axes of N subjets

τ_N measures departure from N-parton energy flow:

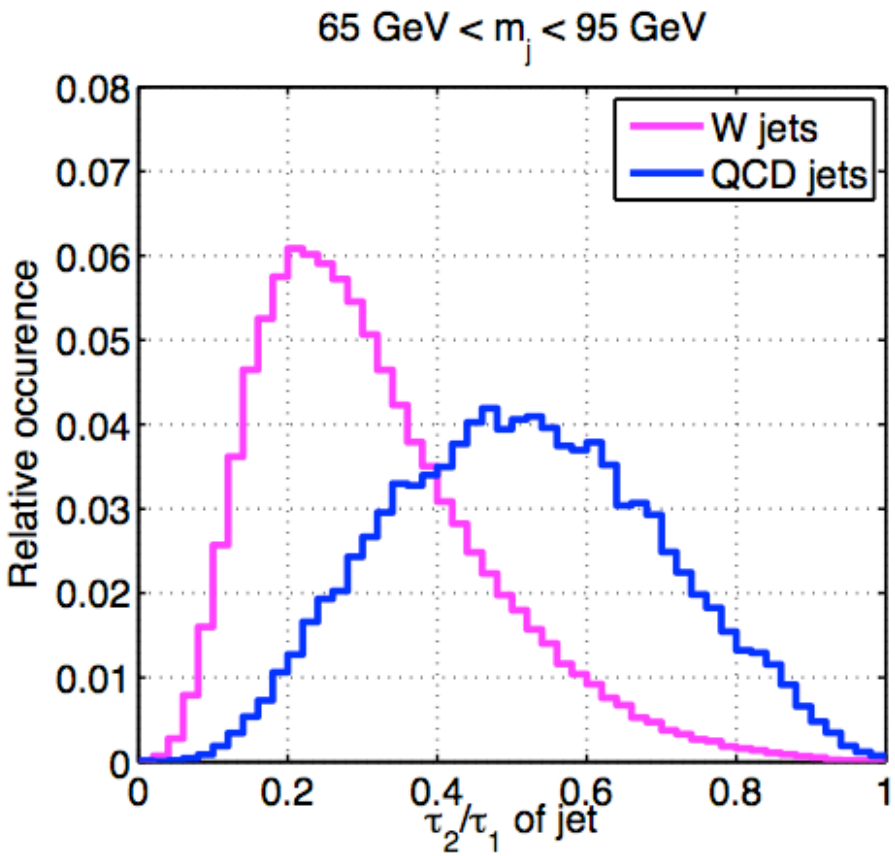
if a jet has N subjets, τ_{N-1} should be much larger than τ_N

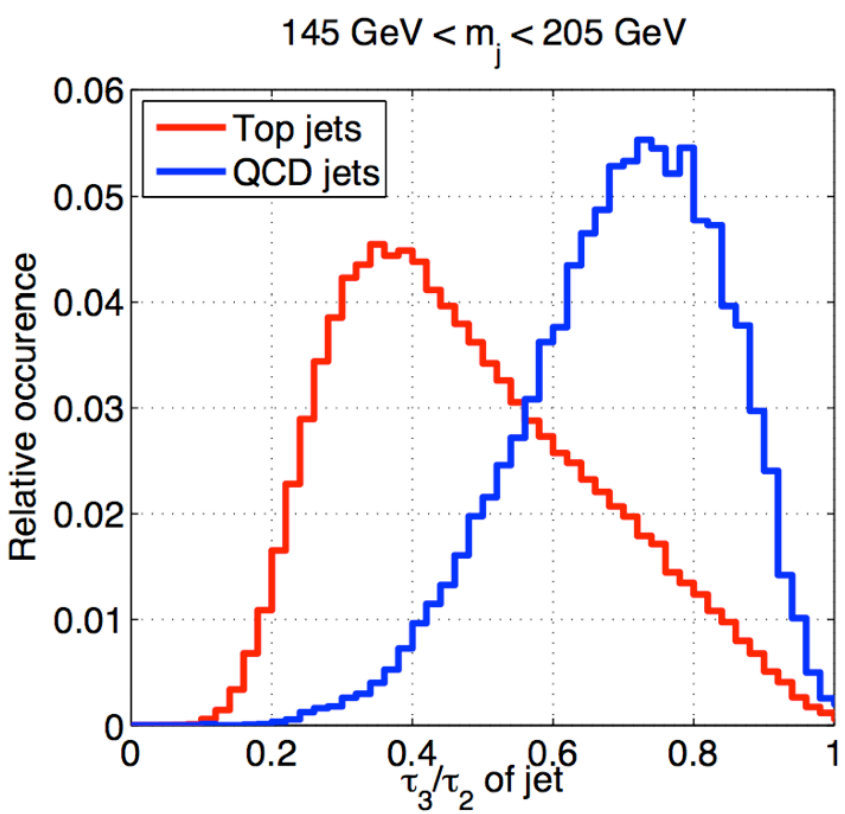
N-subjettiness

Thaler, van Tilburg, 2010

$$\tau_{N,N-1}^{(\beta)} \equiv \frac{\tau_N^{(\beta)}}{\tau_{N-1}^{(\beta)}}$$

A jet with a **small** $\tau_{N,N-1}$ is more likely to have N than N-1 subjets





(from 1011.2268, with β=1)

Nsubjettiness in FastJet

Important: first download and install **FastJet contribs**

Also, have in the Makefile

```
INCLUDE += `fastjet-config --cxxflags`/contrib
LIBRARIES += -lNsubjettiness
```

In FastJet

```
#include "fastjet/contrib/Nsubjettiness.hh"

double beta = 1.0;
double tau21_cut = 0.3;
NsubjettinessRatio nSub21_beta1(2,1,
                                OnePass_WTA_KT_Axes(),
                                UnnormalizedMeasure(beta));
double tau21_beta1 = nSub21_beta1(jets[0]);
if ( tau21_beta1 > tau21_cut ) jets[0]=PseudoJet();
```

Packaging

Subtractors, taggers, groomers, etc
could be easily packaged into

`FindBoostedHeavyParticleDecayingTo2ProngsInThisEvent`
`FindBoostedHeavyParticleDecayingTo3ProngsInThisEvent`

classes, and even into a

`FindVeryHeavyParticleDecayingTo2HeavyParticlesInThisEvent`

class

What has changed in FJ since 2014?

- Versions: 3.0.6 → 3.1.0 → 3.1.1 → 3.1.2 → 3.1.3 → 3.2.0
- Many small (and less small) changes and (generally small) bug fixes, e.g.
  - fortran wrapper for fjcore
  - direct interface to jets: `jets = jet_def(input_particles);`
  - helpers for C++11 and C++14 features
- More relevant progress:
  - Significant speed improvements (x1.5-10) for large N (few thousand –10^5)
  - Exposure of two 'fast' clustering strategies for 3rd party clustering algs
- Upgrades and new contributions in FastJet contribs

Tutorials

Two FastJet tutorials are available on the MC4BSM virtual machine:

- boosted HZ production
- heavy Z' resonance decaying to boosted WW

In both cases, the use of tagging and grooming to enhance the signal over the dijet background is illustrated