Jet clustering tools

Gavin P. Salam
LPTHE, UPMC Paris 6 & CNRS

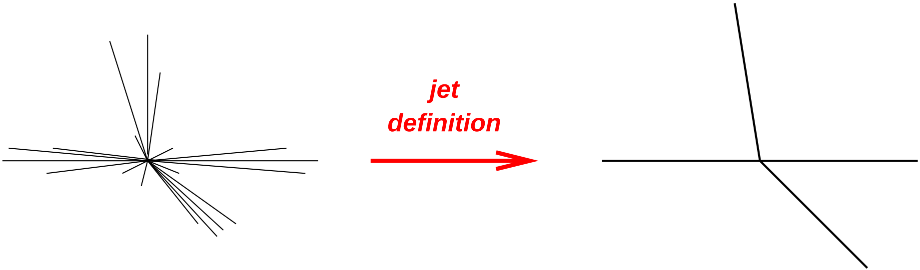
MC4LHC readiness workshop

31 March 2010
CERN

Concentrating mostly on FastJet, developped with
Matteo Cacciari and Gregory Soyez

Jet definition / algorithm

A jet definition is a systematic procedure that **projects away the multiparticle dynamics**, so as to leave a simple picture of what happened in an event:



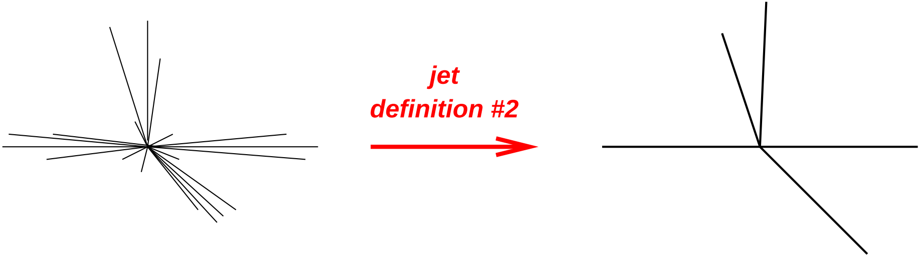
Jets are *as close as we can get to a physical single hard quark or gluon:* with good definitions their properties (multiplicity, energies, [flavour]) are

- finite at any order of perturbation theory
- insensitive to the parton → hadron transition

NB: finiteness ⟷ set of jets depends on jet def.

Jet definition / algorithm

A jet definition is a systematic procedure that **projects away the multiparticle dynamics**, so as to leave a simple picture of what happened in an event:

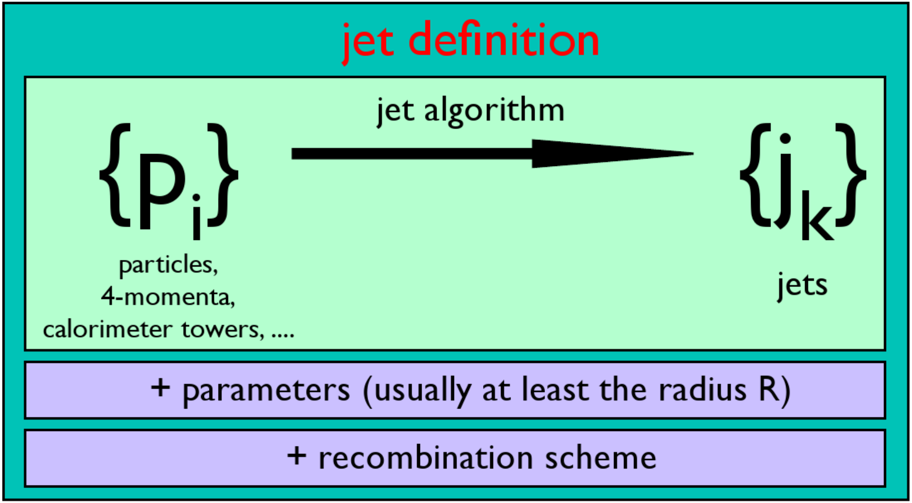


Jets are *as close as we can get to a physical single hard quark or gluon:* with good definitions their properties (multiplicity, energies, [flavour]) are

- finite at any order of perturbation theory
- insensitive to the parton → hadron transition

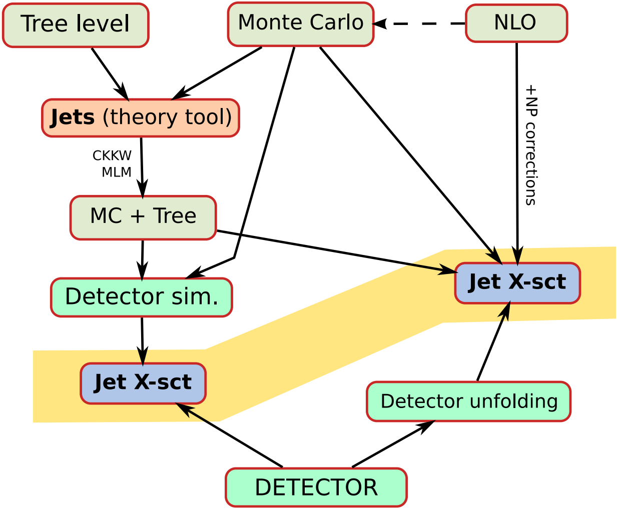
NB: finiteness ⟷ set of jets depends on jet def.

Les Houches 2007 proceedings, arXiv:0803.0678



Reminder: running a jet definition gives a well defined physical observable, which we can measure and, hopefully, calculate

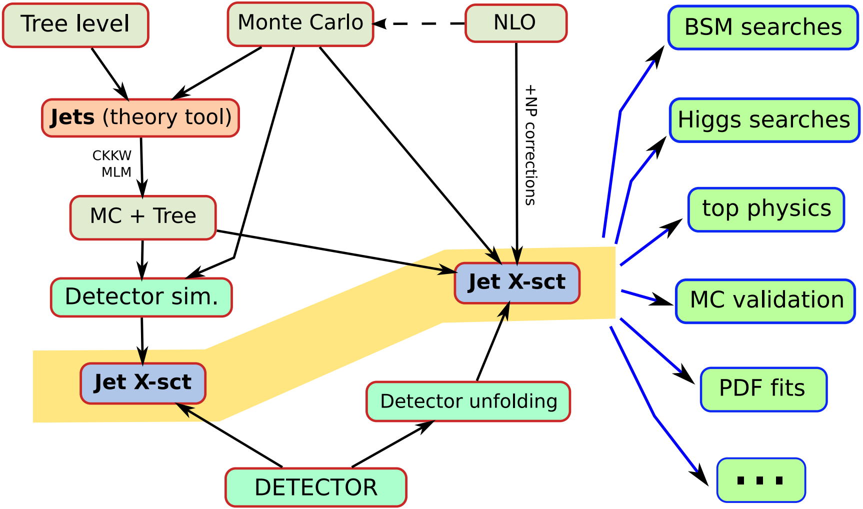
QCD jets flowchart



Jet (definitions) provide central link between expt., “theory” and theory

And jets are an input to almost all analyses

QCD jets flowchart



Jet (definitions) provide central link between expt., "theory" and theory

And jets are an input to almost all analyses

Public jet codes

In rough order of first public release:

- KTCLUS (Fortran)
- ARCLUS (Fortran)
- PxCone (Fortran)
- KTJet (C++)
- Optimal Jet Finder [OJF] (Fortran)
- FastJet + plugins (C++)
- CDF MidPoint and JetClu codes (C++)
- SpartyJet (C++)
- FFTJet (C++)

Also: jet finders in non-jet tools: simple cone jet finders in Pythia, Isajet, Alpgen, PGS, AcerDet, ...; k_t and seedless/midpoint cones in MCFM, NLOJet++, etc.

FastJet

`http://fastjet.fr/`

FastJet

Jan 2006 (FJ 1.0):

- Fast implementation of pp k_t algorithm — Cacciari & GPS

  N^2 and $N \ln N$ timings for clustering N particles v. N^3 with earlier codes

  $N \ln N$ strategy relies on external package CGAL

Oct 2006 (FJ 2.0):

- Implementation of Cambridge/Aachen algorithm

  including coding of Chan's Closest Pair algorithm
- Introduction of jet areas and background estimation/subtraction
- New interface — for long-term stability

Apr 2007 (FJ 2.1):

- Plugin mechanism giving common interface to external jet finders
- Inclusion of plugins that wrap CDF (JetClu, Midpoint) code and PxCone
- Inclusion of SISCone as a plugin

FastJet

Jan 2008 (FJ 2.3): Soyez joined development team

- Added the anti-k_t algorithm (fast, native implementation)
- Added "passive" and "Voronoi" areas
- Switched to autotools for compilation/installation
- Better access to information for subjet studies
- Basic Fortran wrapper

April 2009 (FJ 2.4):

- Added plugins for DØRunIICone, ATLAS cone, CMS cone, TrackJet

  DØ and Trackjet code contributed by Sonnenschein

  ATLAS code taken from SpartyJet
- Added gen-k_t + e^+e^- algorithms (k_t, Cambridge, Jade, e^+e^- anti-k_t)
- Framework for handling user-supplied clustering distances (NNH)

Jet algorithms list

Native implementations:

- longitudinally invariant kt
- (inclusive) Cambridge/Aachen
- anti-kt
- gen-kt
- e^+e^- kt and gen-kt

Plugins (distributed with FastJet)

- SISCone
- CDF MidPoint [IR_{3+1} unsafe]
- CDF JetClu [IR_{2+1} unsafe]
- D0 Run II Cone [IR_{3+1} unsafe]
- ATLAS Cone algorithm [IR2+1 unsafe]
- CMS Cone algorithm [Coll_{3+1} unsafe]
- TrackJet [Coll_{3+1} unsafe]
- PxCone (fortran 77) [IR_{3+1} unsafe]
- e^+e^- (spherical) SISCone
- e^+e^- JADE algorithm
- e^+e^- Cambridge algorithm

Jet algorithms list

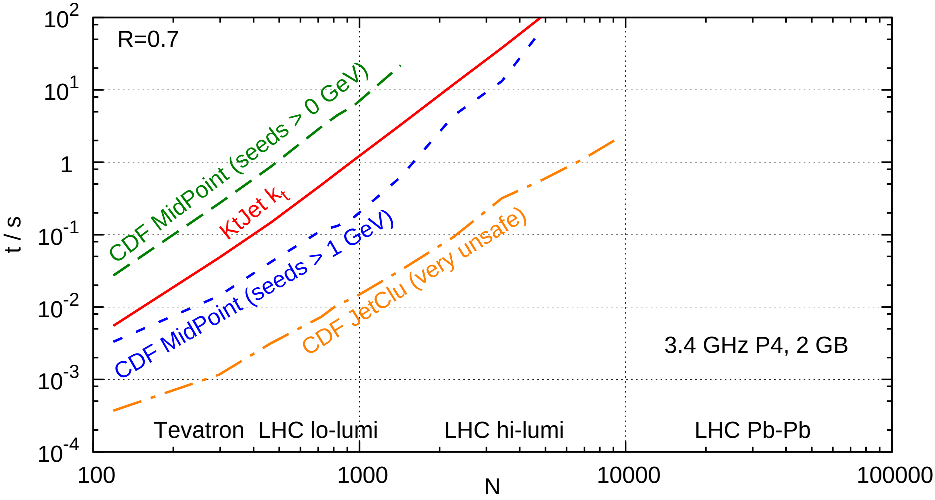
Native implementations:

- longitudinally invariant kt
- (inclusive) Cambridge/Aachen
- anti-kt
- gen-kt
- e^+e^- kt and gen-kt

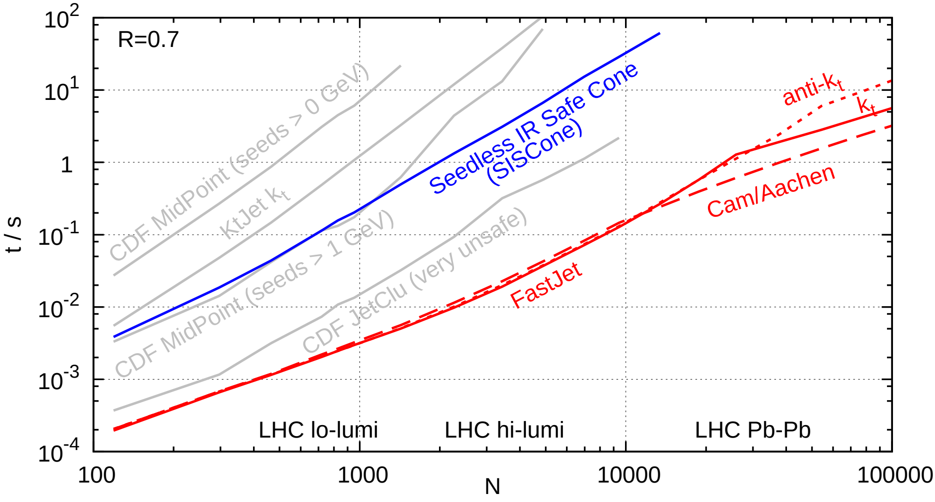
Plugins (distributed with FastJet)

- SISCone
- CDF MidPoint [IR_{3+1} unsafe]
- CDF JetClu [IR_{2+1} unsafe]
- D0 Run II Cone [IR_{3+1} unsafe]
- ATLAS Cone algorithm [IR2+1 unsafe]
- CMS Cone algorithm [$Coll_{3+1}$ unsafe]
- TrackJet [$Coll_{3+1}$ unsafe]
- PxCone (fortran 77) [IR_{3+1} unsafe]
- e^+e^- (spherical) SISCone
- e^+e^- JADE algorithm
- e^+e^- Cambridge algorithm

Speeds in 2005



Speeds in 2009



Adapting seq. rec. to give circular jets

Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08

Adapting seq. rec. to give circular jets

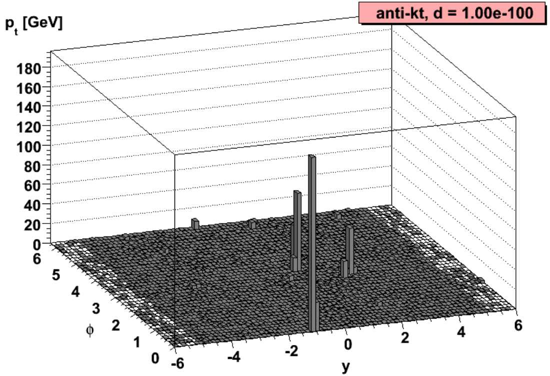
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-}\mathbf{k_t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

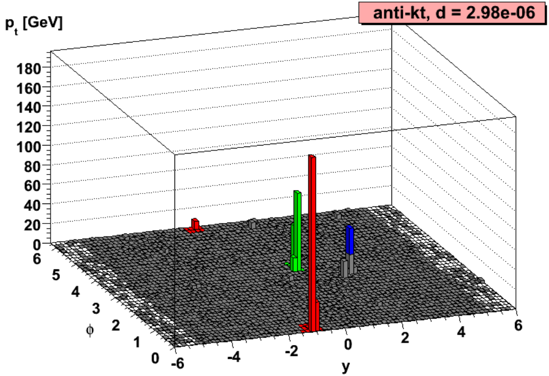
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

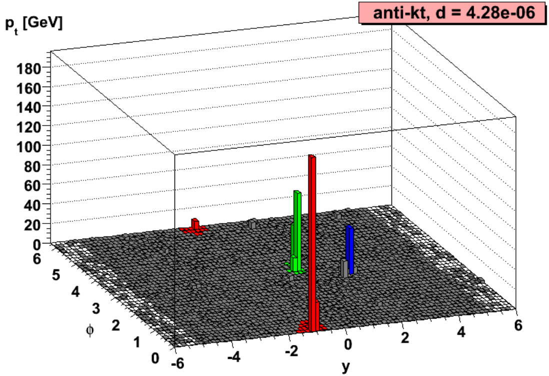
Soft stuff clusters with nearest neighbour

$$k_t\text{: } d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\text{: } d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

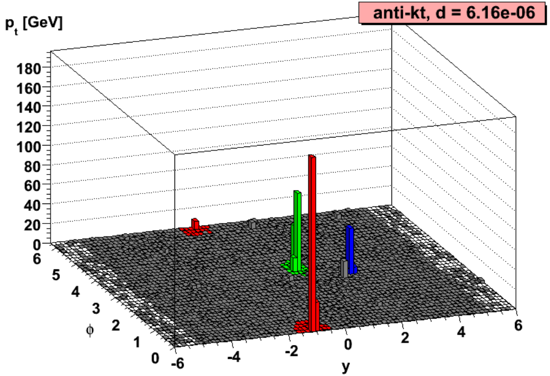
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

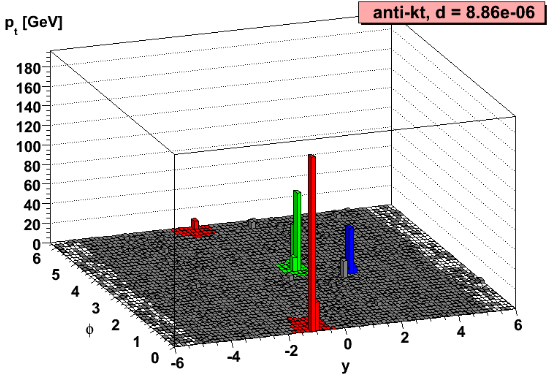
Soft stuff clusters with nearest neighbour

$$k_t\text{: } d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\text{: } d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

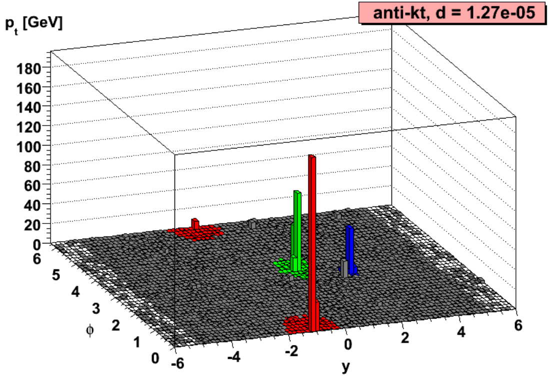
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

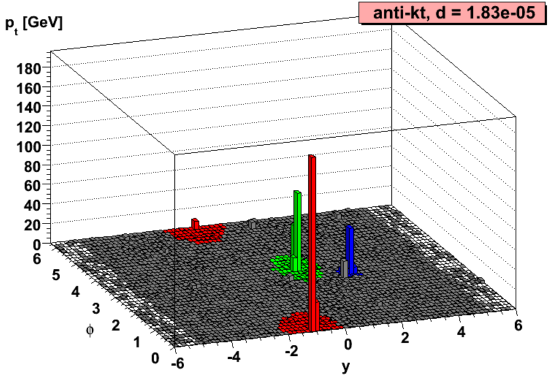
Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

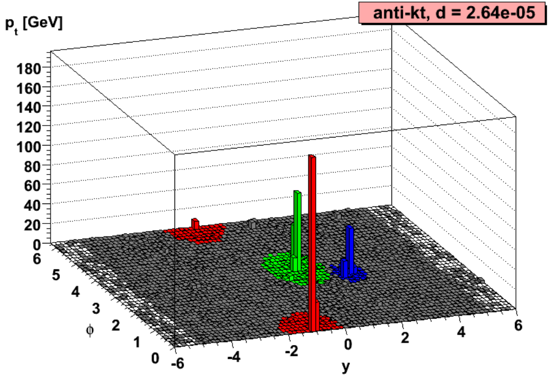
Soft stuff clusters with nearest neighbour

$$k_t\text{: } d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-}\mathbf{k_t}\text{: } d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

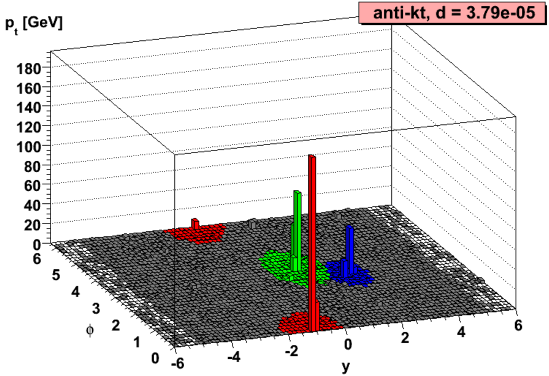
Soft stuff clusters with nearest neighbour

$$k_t:\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}:\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

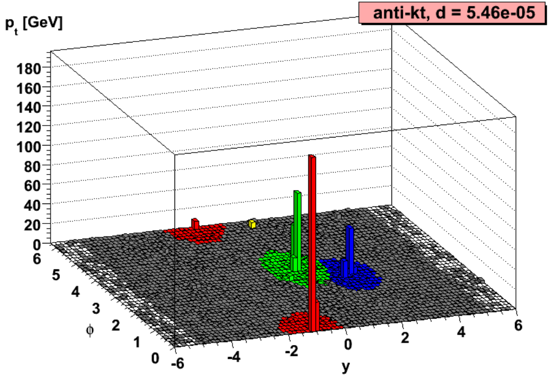
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

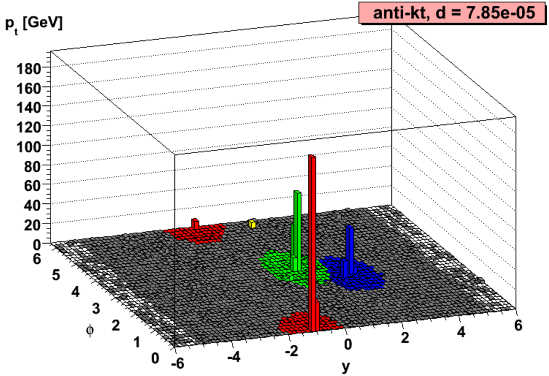
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



Adapting seq. rec. to give circular jets

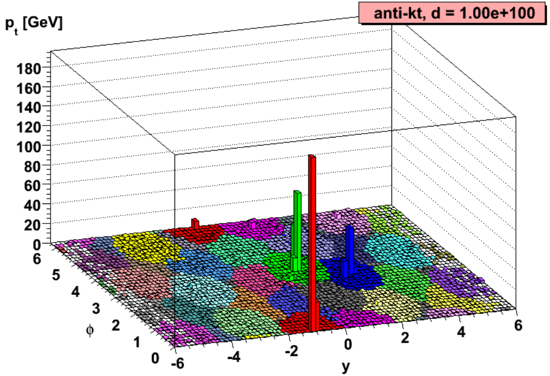
Soft stuff clusters with nearest neighbour

$$k_t\colon\ d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2 \longrightarrow \textbf{anti-k}_\mathbf{t}\colon\ d_{ij} = \frac{\Delta R_{ij}^2}{\max(k_{ti}^2, k_{tj}^2)}$$

Hard stuff clusters with nearest neighbour

Privilege collinear divergence over soft divergence

Cacciari, GPS & Soyez '08



anti-k_t gives cone-like jets without using stable cones

Example code

```
#include "fastjet/ClusterSequence.hh"
using namespace fastjet;
using namespace std;

int main () {
  // choose a jet definition
  double R = 0.7;
  JetDefinition jet_def(kt_algorithm, R);

  vector<PseudoJet> particles;
  // build event with 2 particles:   px py pz    E
  particles.push_back( PseudoJet( 100.0, 0, 0, 100.0) );
  particles.push_back( PseudoJet(-100.0, 0, 0, 100.0) );

  // run the clustering, extract the jets
  ClusterSequence cs(particles, jet_def);
  vector<PseudoJet> jets = cs.inclusive_jets();
}
```

Example code

```
#include "fastjet/ClusterSequence.hh"
using namespace fastjet;
using namespace std;

int main () {
  // choose a jet definition
  double R = 0.7, f = 0.75;
  JetDefinition jet_def = new SISConePlugin(R, f);

  vector<PseudoJet> particles;
  // build event with 2 particles:   px py pz    E
  particles.push_back( PseudoJet( 100.0, 0, 0, 100.0) );
  particles.push_back( PseudoJet(-100.0, 0, 0, 100.0) );

  // run the clustering, extract the jets
  ClusterSequence cs(particles, jet_def);
  vector<PseudoJet> jets = cs.inclusive_jets();
}
```

Jets “ecosystem”

Jets ecosystem: users

Individuals

- Anyone needing simple jet finding — need stable, simple interface
- People playing with new jet ideas — need flexible interface
- Theorists who still like Fortran

Community-wide projects

- Rivet — One of the drivers for inclusion of "legacy" jet algorithms
- Delphes detector simulation

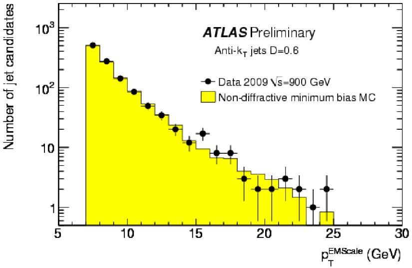
Experiments

- The four main LHC experiments all use FastJet
- ATLAS and CMS have chosen anti-k_t as the first jet alg. to calibrate
- ATLAS uses FastJet in the high-level trigger — It had better not crash!

Anti-k_t at LHC ($\sqrt{s} = 900$ GeV)

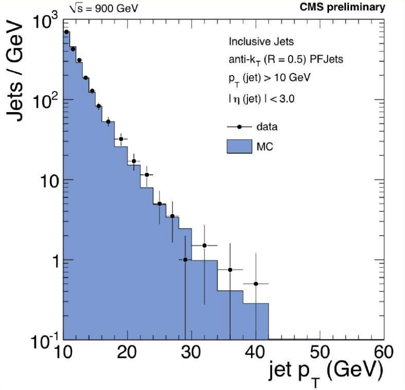
ATLAS

Jets from AntiK$_T$ D=0.6 algorithm
|η|< 2.6 and p$_T$> 7 GeV



CMS

Particle Flow



Jets ecosystem: plugins (or not)

External plugins for FastJet: (not included in release)

- Variable R plugin — Krohn, Thaler & Wang '09
- Pruning plugin — Ellis, Vermillion & Walsh '09
- Trimming plugin — Krohn, Thaler & Wang '10

Algorithms not naturally "pluggable" into FastJet:

FastJet designed for algorithms for which each particle ends up in at most 1 jet. Not all algorithms fit this picture:

- ARCLUS ($3 \to 2$ clustering)
- OJF (a particle has weighted assignment to multiple jets)
- FFTJet, in its "fuzzy" mode (weighted assignment)

Jets ecosystem: extensions, tools

SpartyJet

Delsart, Geerlings, Huston & Martin '06-

- Provides `root` interface to FastJet, including PyRoot access
- Provides visualisation tools
- Also has a number of native implementations of jet algs

FastJet Tools page

- A range of boosted-particle finders (Higgs, top, etc.)
  Our own, links to other people's, and our implementations of other people's
- Background (UE/pileup) estimation and subtraction tools
  Already in FJ, more flexible versions in the works
- Filtering
  cleanup of UE/pileup noise to improve resolution
  [Butterworth, Davison, Rubin & GPS '08]
  ["trimming" is closely related]

Physics Roadmap:

Questions include

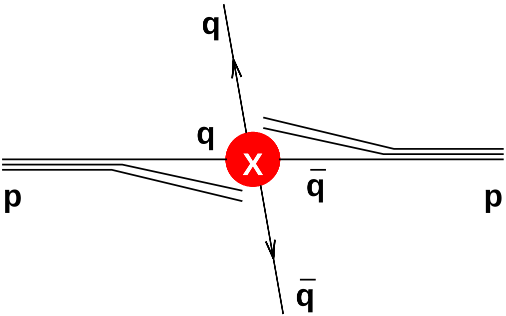
a) Developing (analytical) understanding of different uses of jets

b) Designing better analyses as a result

What follows is an illustration

What R is best for an isolated jet?

E.g. to reconstruct $m_X \sim (p_{tq} + p_{t\bar{q}})$



PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadronisation:

$$q: \quad \langle \Delta p_t \rangle \simeq -\frac{C_F}{R} \cdot 0.4 \text{ GeV}$$

Underlying event:

$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5\text{–}15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

in small-R limit (?!)
cf. Dasgupta, Magnea & GPS '07

What R is best for an isolated jet?

PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadronisation:

$$q: \quad \langle \Delta p_t \rangle \simeq -\frac{C_F}{R} \cdot 0.4 \text{ GeV}$$

Underlying event:

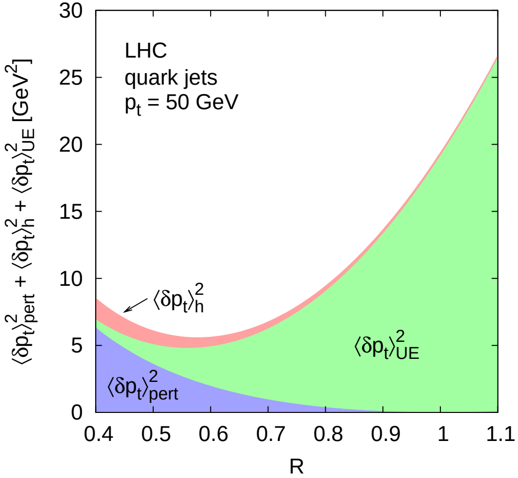
$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5\text{--}15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

50 GeV quark jet



in small-R limit (?!)

cf. Dasgupta, Magnea & GPS '07

What R is best for an isolated jet?

PT radiation:

$$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{\pi} p_t \ln R$$

Hadronisation:

$$q: \quad \langle \Delta p_t \rangle \simeq -\frac{C_F}{R} \cdot 0.4 \text{ GeV}$$

Underlying event:

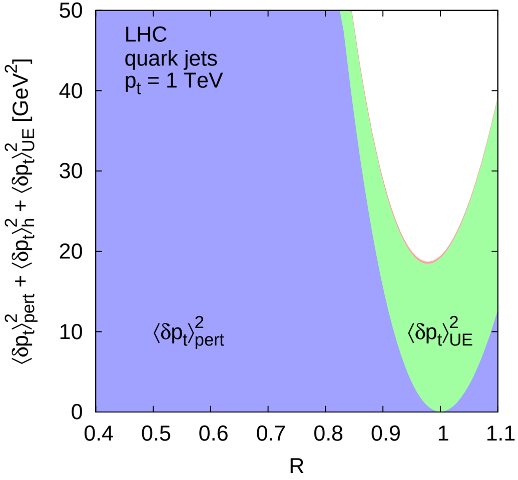
$$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5\text{–}15 \text{ GeV}$$

Minimise fluctuations in p_t

Use crude approximation:

$$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$$

1 TeV quark jet



in small-R limit (?!)

cf. Dasgupta, Magnea & GPS '07

What R is best for an isolated jet?

PT radiation:

$q: \quad \langle \Delta p_t \rangle \simeq \frac{\alpha_s C_F}{} p_t \ln R$

At low p_t, small R limits relative impact of UE

At high p_t, perturbative effects dominate over non-perturbative $\to R_{best} \sim 1$.

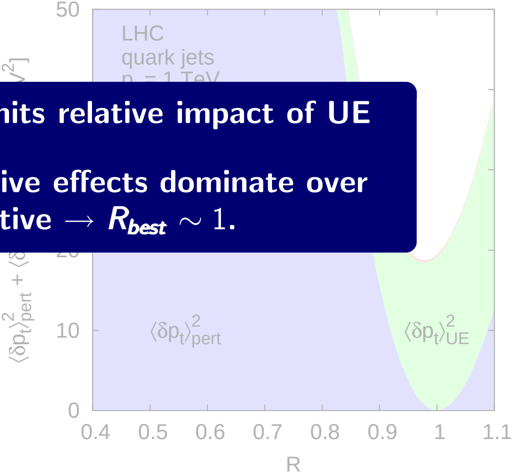
Hadr

$q:$

Underlying event:

$q, g: \quad \langle \Delta p_t \rangle \simeq \frac{R^2}{2} \cdot 2.5 - 15 \text{ GeV}$

1 TeV quark jet



in small-R limit (?!)

cf. Dasgupta, Magnea & GPS '07

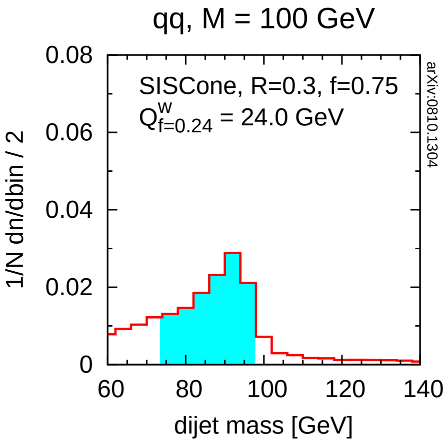
Minimise fluctuations in p_t

Use crude approximation:

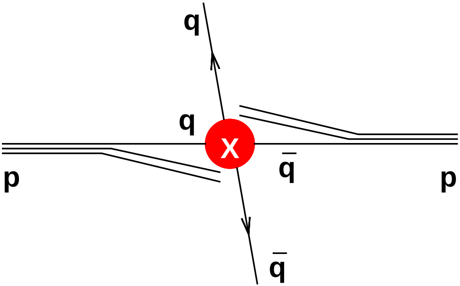
$\langle \Delta p_t^2 \rangle \simeq \langle \Delta p_t \rangle^2$

Dijet mass: scan over *R* [Pythia 6.4]

R = 0.3

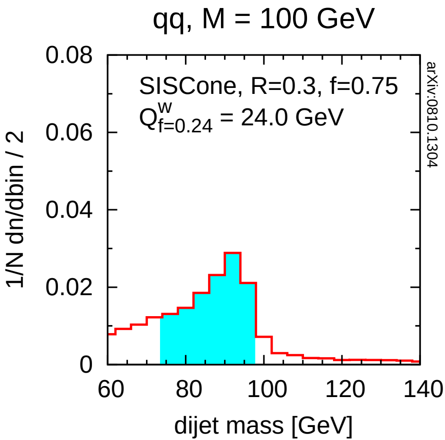


Resonance X → dijets

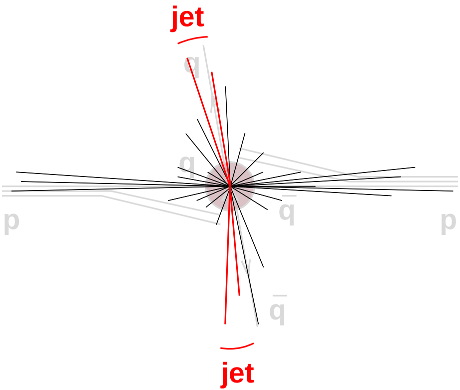


Dijet mass: scan over R [Pythia 6.4]

R = 0.3

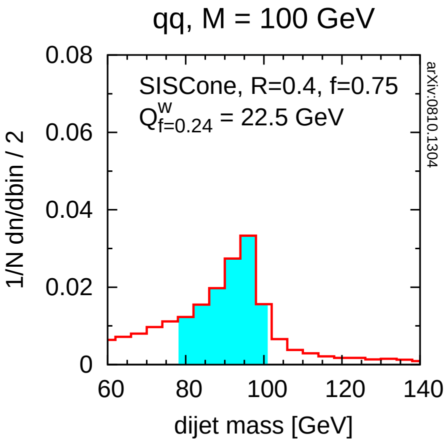


Resonance X → dijets

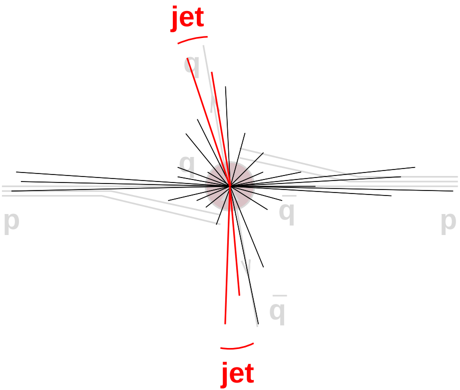


Dijet mass: scan over *R* [Pythia 6.4]

R = 0.4

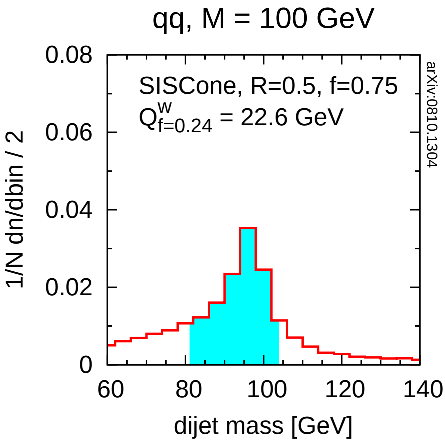


Resonance X → dijets

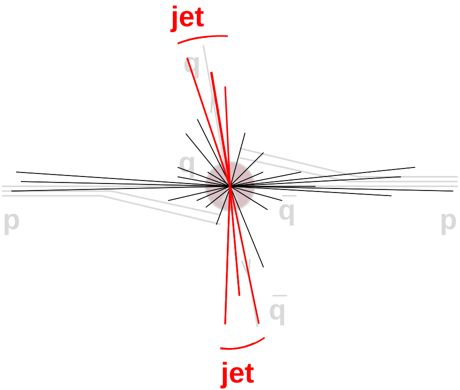


Dijet mass: scan over *R* [Pythia 6.4]

R = 0.5

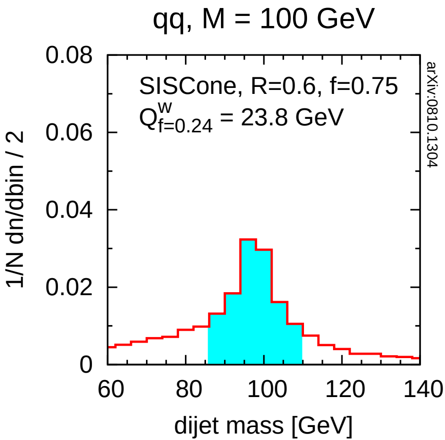


Resonance X → dijets

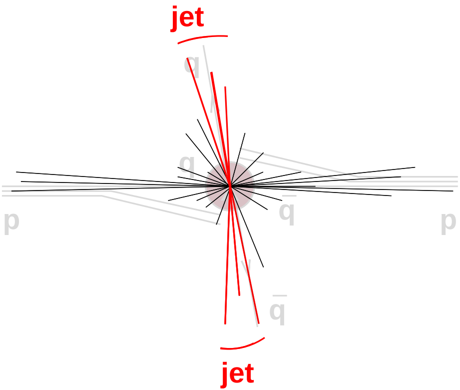


Dijet mass: scan over *R* [Pythia 6.4]

R = 0.6

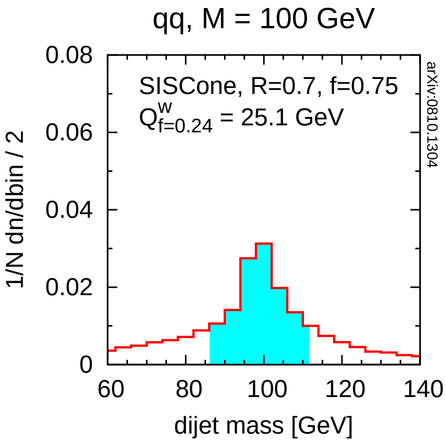


Resonance X → dijets

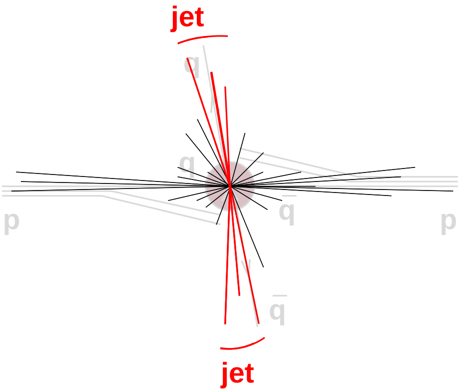


Dijet mass: scan over *R* [Pythia 6.4]

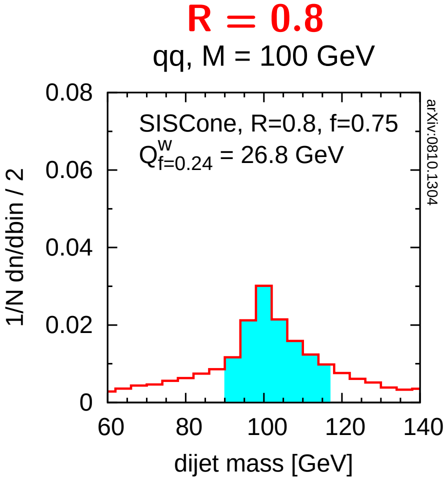
R = 0.7



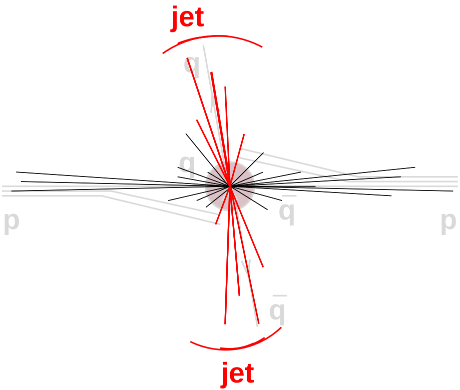
Resonance X → dijets



Dijet mass: scan over *R* [Pythia 6.4]



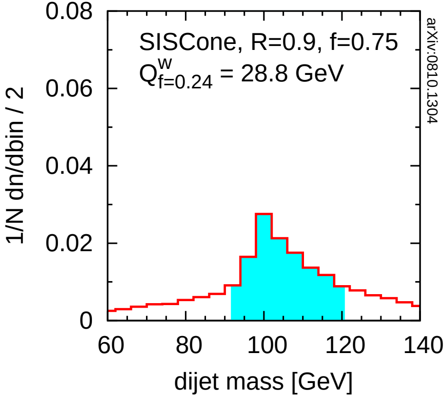
Resonance X → dijets



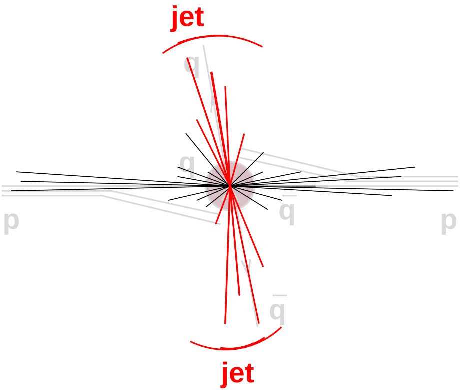
Dijet mass: scan over *R* [Pythia 6.4]

R = 0.9

qq, M = 100 GeV



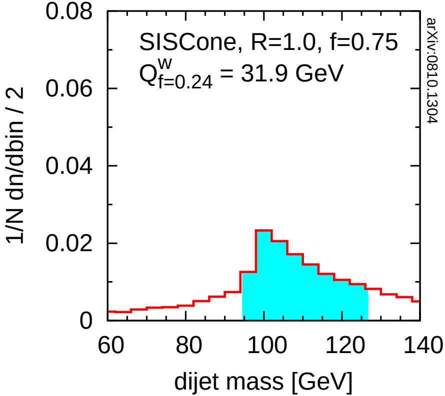
Resonance X → dijets



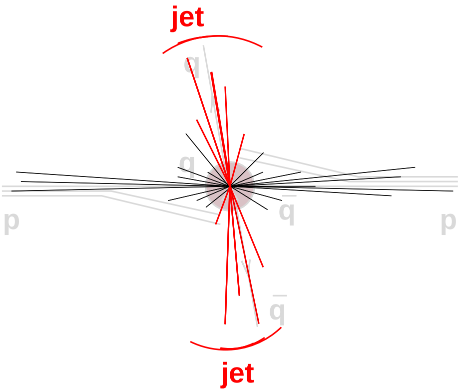
Dijet mass: scan over R [Pythia 6.4]

R = 1.0

qq, M = 100 GeV

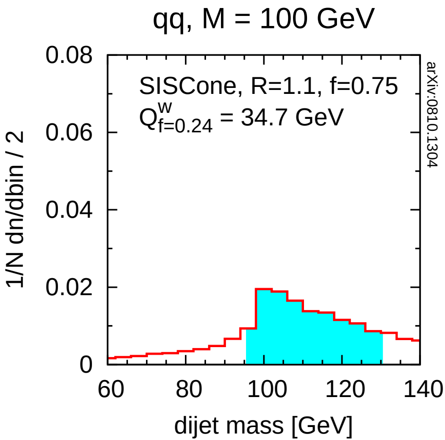


Resonance X → dijets

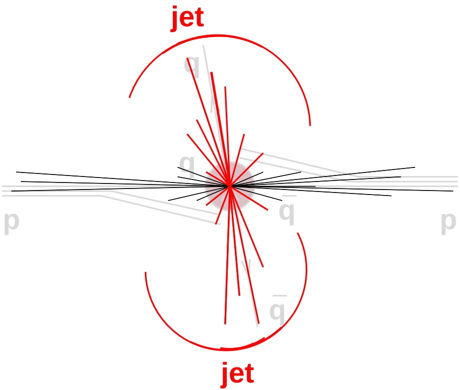


Dijet mass: scan over R [Pythia 6.4]

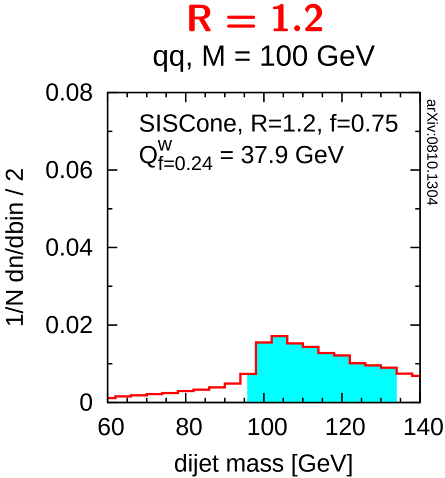
R = 1.1



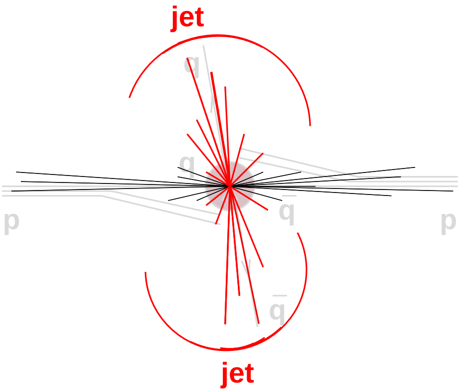
Resonance X → dijets



Dijet mass: scan over *R* [Pythia 6.4]

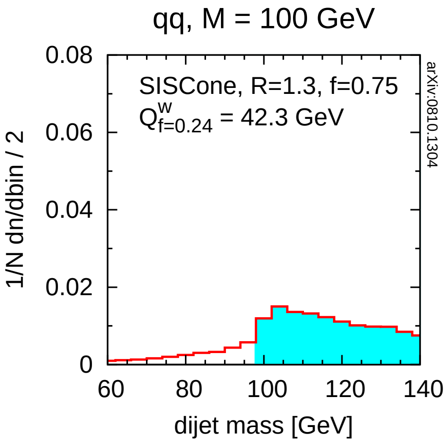


Resonance X → dijets

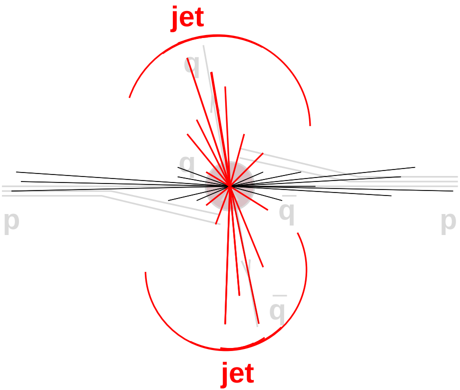


Dijet mass: scan over *R* [Pythia 6.4]

R = 1.3

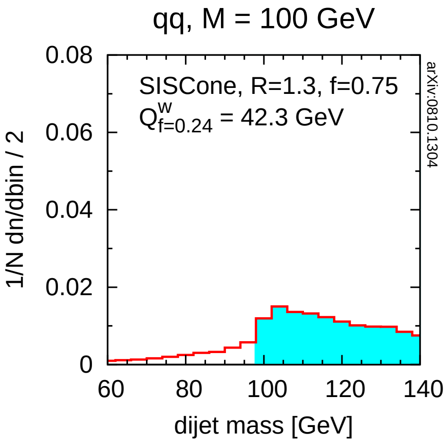


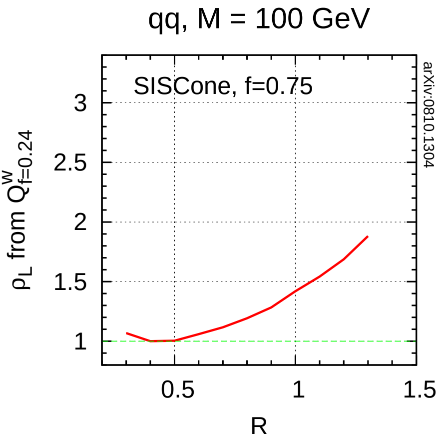
Resonance X → dijets



Dijet mass: scan over R [Pythia 6.4]

R = 1.3



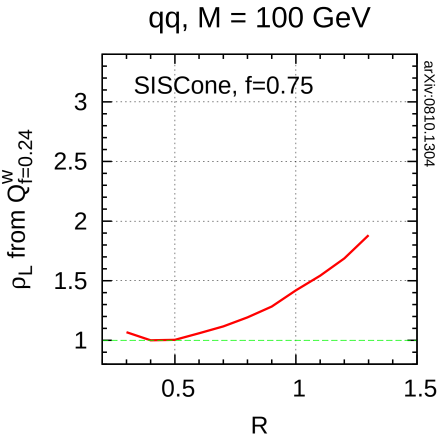


After scanning, summarise "quality" v. *R*. Minimum ≡ BEST

picture not so different from crude analytical estimate

Scan through $q\bar{q}$ mass values

$m_{qq} = 100$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

  NB: current analytics too crude

BUT: so far, LHC's plans involve running with fixed smallish R values

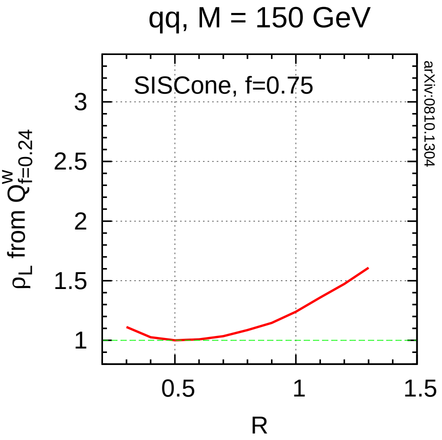
e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$m_{qq} = 150$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

BUT: so far, LHC's plans involve running with fixed smallish R values

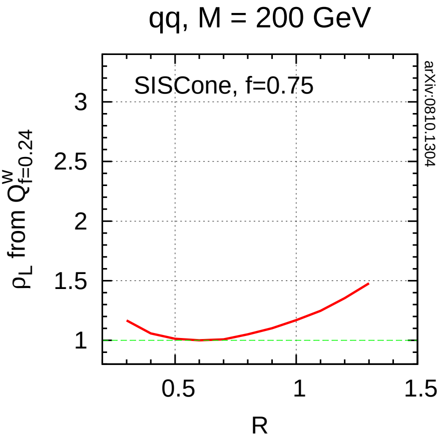
e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 200\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

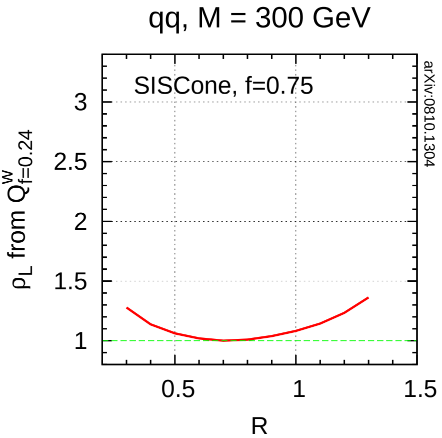
BUT: so far, LHC's plans involve running with fixed smallish R values

e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 300\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

  NB: current analytics too crude

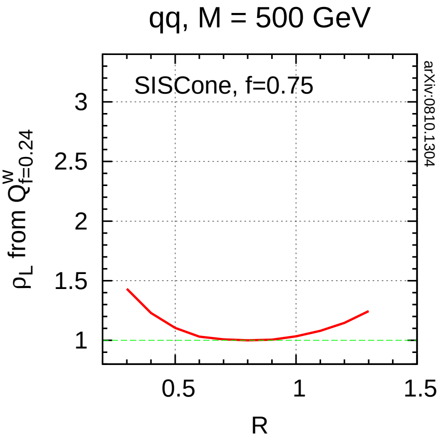
BUT: so far, LHC's plans involve running with fixed smallish R values

e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

m_{qq} = 500 GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

  NB: current analytics too crude

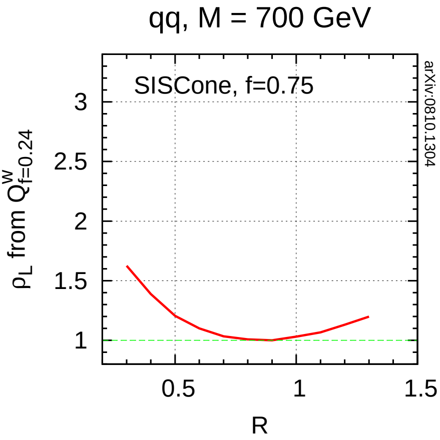
BUT: so far, LHC's plans involve running with fixed smallish R values

e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$m_{qq} = 700$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

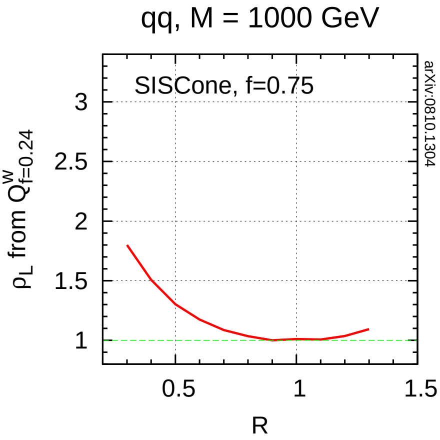
BUT: so far, LHC's plans involve running with fixed smallish R values

e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 1000\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

BUT: so far, LHC's plans involve running with fixed smallish R values

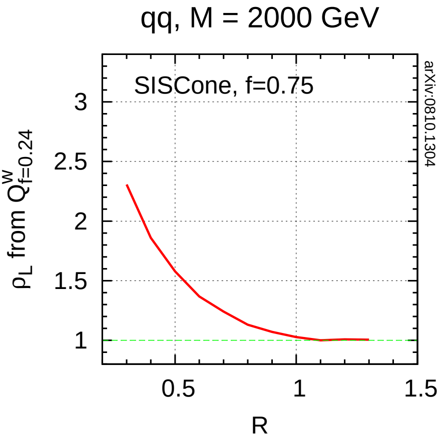
e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 2000\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

BUT: so far, LHC's plans involve running with fixed smallish R values

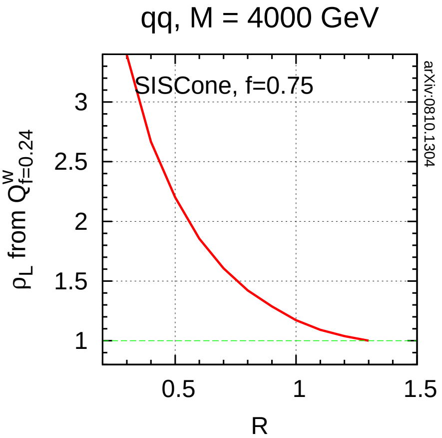
e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$m_{qq} = 4000$ GeV



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

BUT: so far, LHC's plans involve running with fixed smallish R values

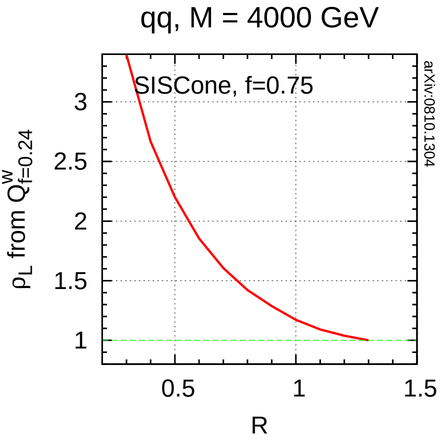
e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 4000\ GeV}$



Best R is at minimum of curve

- Best R depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

BUT: so far, LHC's plans involve running with fixed smallish *R* values

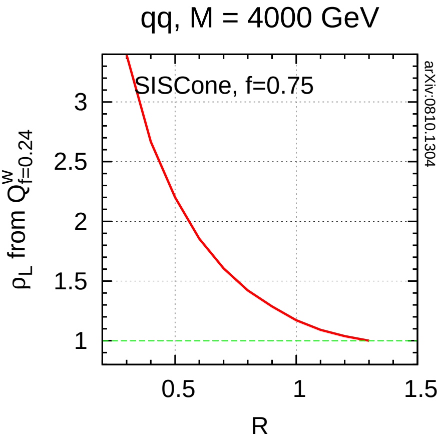
e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow qq and gg resonances from `http://quality.fastjet.fr`

Cacciari, Rojo, GPS & Soyez '08

Scan through $q\bar{q}$ mass values

$\mathbf{m_{qq} = 4000\ GeV}$



Best *R* is at minimum of curve

- Best *R* depends strongly on mass of system
- Increases with mass, just like crude analytical prediction

NB: current analytics too crude

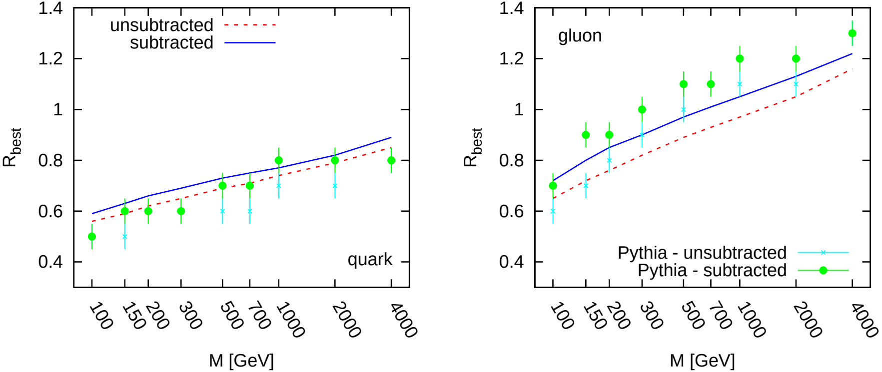
BUT: so far, LHC's plans involve running with fixed smallish *R* values

e.g. CMS arXiv:0807.4961

NB: 100,000 plots for various jet algorithms, narrow *qq* and *gg* resonances from `http://quality.fastjet.fr` Cacciari, Rojo, GPS & Soyez '08

Analytical R_{best}

Medium-term aim: have ability for FastJet to suggest near-optimal parameter choices for different classes of analysis. Can be based on MC study, or analytical calculations:



Soyez, preliminary (LHC 14 TeV)

Other active "Physics Roadmap" topics

- Search strategies with jets in complex events
  - Boosted objects [Several groups working]
  - Non-boosted objects
- Further work on noise reduction (UE/pileup removal)
- Further understanding of UE/pileup characterisation

  e.g. Cacciari, GPS & Sapeta '10
- Jets in heavy-ion collisions

Software Roadmap

Long-term maintainability
Facilitation of “advanced” usage

Long-term maintainability

The most frequently used core code (`N2Tiled` strategy) was written in the space of a couple of days in 2005.

Not quite spaghetti code (C-style macaroni code?)

It could do with a cleanup

An important part of maintanability is validation:

- We currently check compilation and clustering results for 1000 MC events for all algorithms every night on several systems.

  We would like to switch to 10k events

- Other aspects of FastJet (e.g. jet areas) not yet subject to automatic validation, but that should probably change.

Planned interface evolution

We aim to maintain backwards compatibility for extended periods of time (allow 2-3 years from "deprecation" to "removal" of any feature).

Apparently "small" user-interface additions. E.g. from

```
vector<PseudoJet> constituents = cluster_sequence->constituents(jet)
```

to

```
vector<PseudoJet> constituents = jet.constituents();
```

Has memory management implications. But can help significantly with advanced usage.

One step on the way to a simple "tools" interface.

Goal: easy and centralized access to helpful utilities

Conclusions

Conclusions

Main public jet-clustering package currently is FastJet

- Code is quite stable
- Provides access to a lot more than just native FastJet algorithms

Future evolution

- Physics-driven: how can we make better use of jets?
- Code should provide facilities to make this easy in pracice

User feedback is welcome!

- It has driven inclusion of "legacy" plugins
- It can help shape future evolution of code
- E.g. should there be built-in access from Python, PyRoot?