DIS 2106
Tsukuba, 21 April 2006

Dispelling the N^3 Myth for the k_t Jet-Finder

Matteo Cacciari
LPTHE - Paris 6

with Gavin Salam, hep-ph/0512210

Outline

- Why Jets — 'cuz it's better
- Why k_t clustering
- Why `FastJet` — 'cuz it's way faster

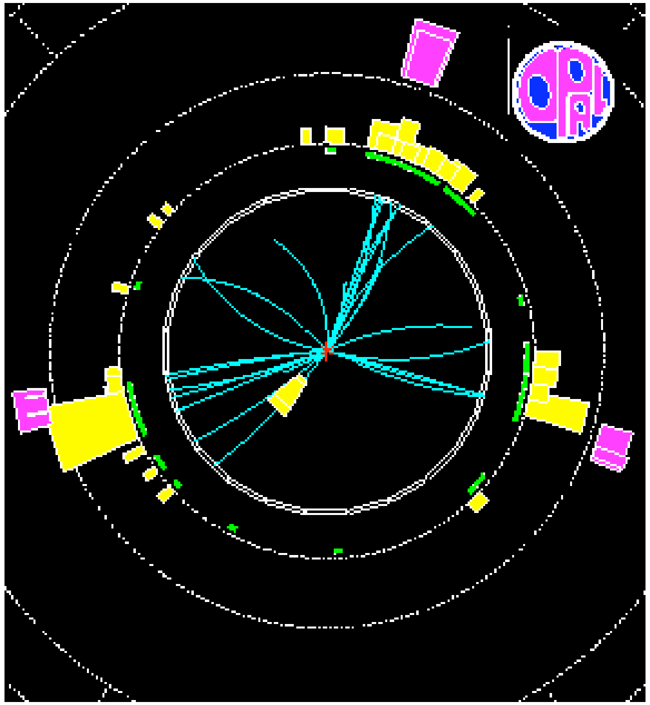
Short version of this talk

Why Jets

Well, because.... jets happen!

A high energy event will in general show **collimated bunches of hadrons**

Starting from this observation and this very loose definition we must work to make jets **good proxies** of the underlying partons, **quarks and gluons**



NB. This is not a review talk in jet physics or even in jet-clustering algorithms

Rather, just a shameless **sales pitch** for our **`FastJet`** code

Why Jets

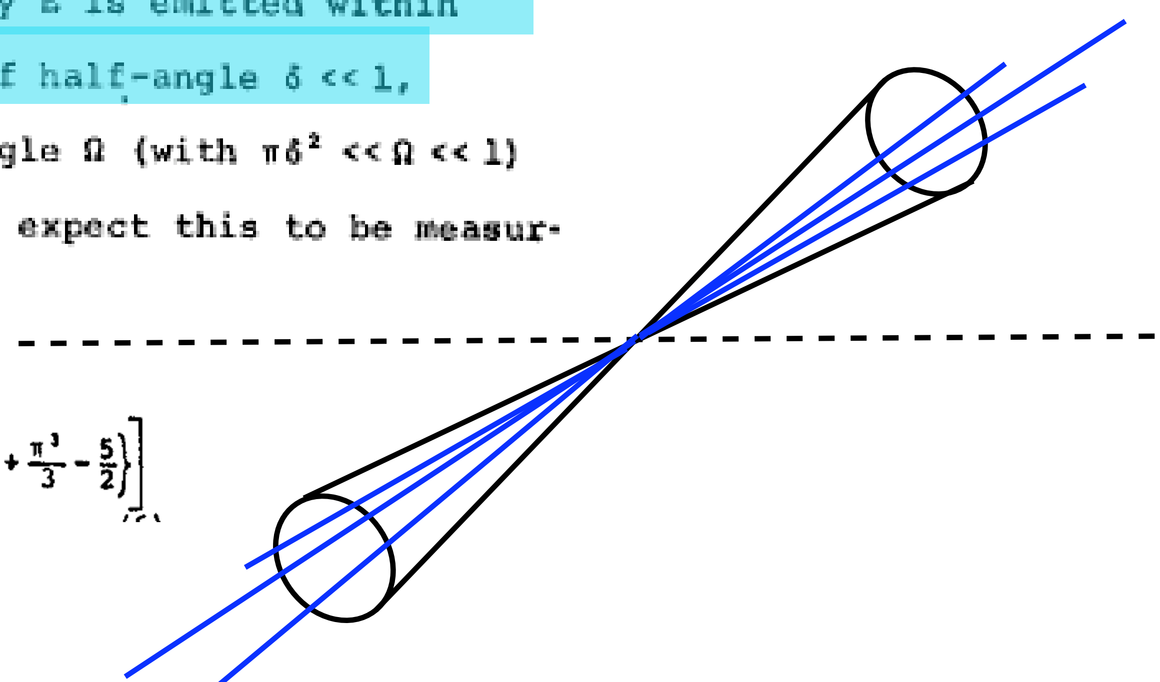
Jets are as old as the parton model (yes, even older than QCD...):

S.D. Drell, D.J. Levy and T.M. Yan, Phys. Rev. **187**, 2159 (1969) and **D1**, 1617 (1970)
N. Cabibbo, G. Parisi and M. Testa, Lett. Nuovo Cimento **4**, 35 (1970)
J.D. Bjorken and S. D. Brodsky, Phys. Rev. **D1**, 1416 (1970)
R.P. Feynman, Photon Hadron Interactions, p. 166 (1972)

The first rigorous definition of an **infrared and collinear safe** jet in QCD is due to Sterman and Weinberg, Phys. Rev. Lett. **39**, 1436 (1977):

To study jets, we consider the partial cross section $\sigma(E,\theta,\Omega,\epsilon,\delta)$ for e^+e^- hadron production events, in which all but a fraction $\epsilon \ll 1$ of the total e^+e^- energy E is emitted within some pair of oppositely directed cones of half-angle $\delta \ll 1$, lying within two fixed cones of solid angle Ω (with $\pi\delta^2 \ll \Omega \ll 1$) at an angle θ to the e^+e^- beam line. We expect this to be measur-

$$\sigma(E,\theta,\Omega,\epsilon,\delta) = (d\sigma/d\Omega)_0\,\Omega\left[1 - (g_E^2/3\pi^2)\left\{3\ln\delta + 4\ln\delta\,\ln 2\epsilon + \frac{\pi^2}{3} - \frac{5}{2}\right\}\right]$$



Why Jets

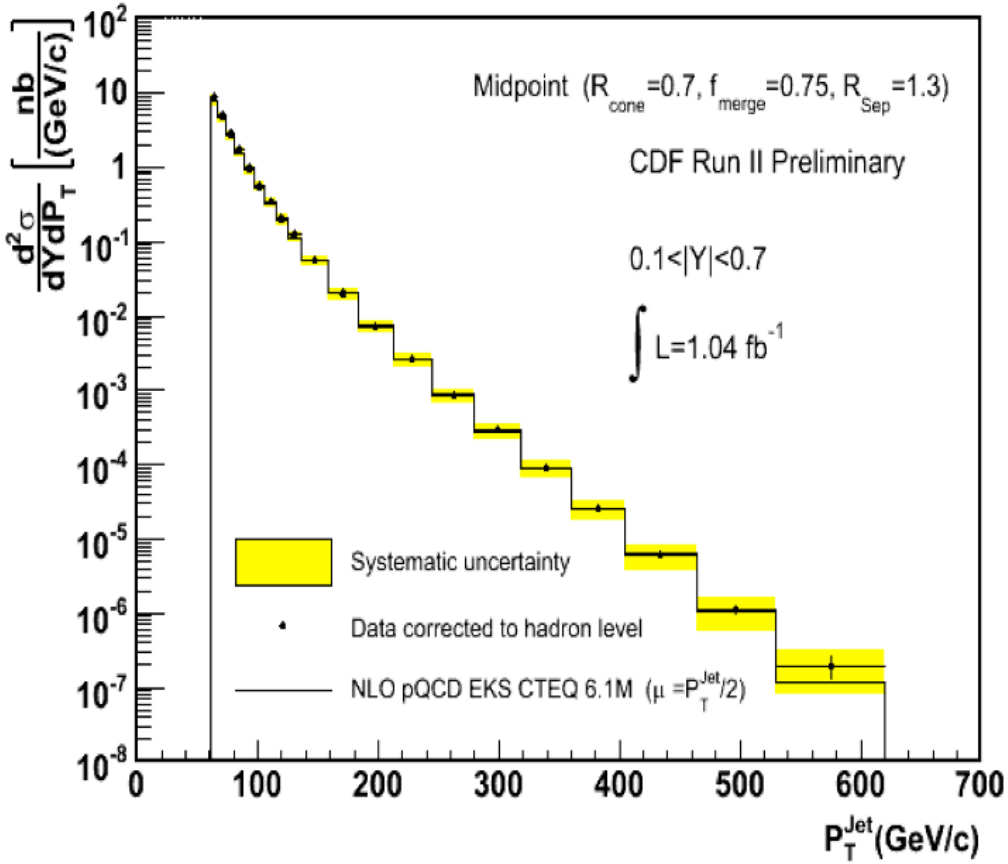
Two main jet-finder classes: **cone algorithms** and **sequential clustering algorithms**

- **Cone-type** algorithms are mainly used at the Tevatron. Extensions of original Sterman-Weinberg idea, i.e. **identify energy flow into cones**. Detailed definition can be messy. Infrared/collinear safety must be carefully studied.

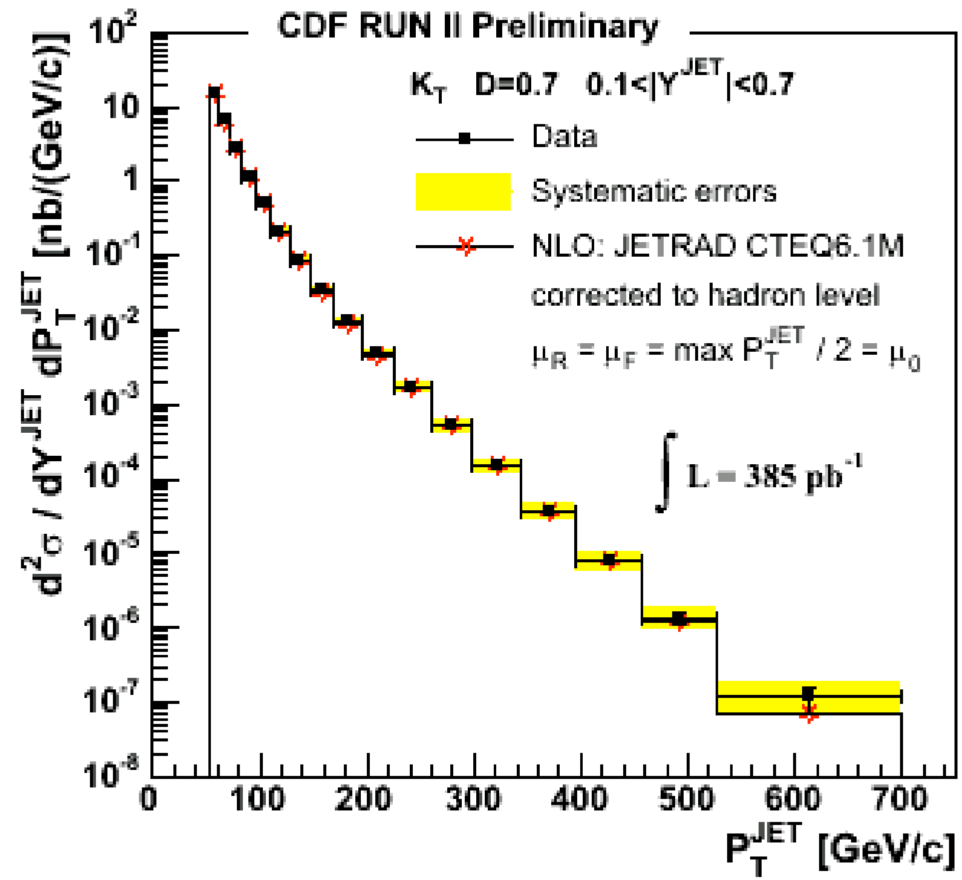
- **Sequential clustering** algorithms are based on **pair-wise successive recombinations**. Widely used at LEP and HERA. Simple definition, safely infrared and collinear safe.

Why k_t

Cone



k_t



[CDF Coll., presented at La Thuile and Moriond]

At face value, both cone and k_t allow for good data/theory comparisons.

However, there are a number of reasons why k_t should be preferred

Why k_t

The definition of a cone algorithm can be extremely complicated.

For instance, take **MidPoint**:

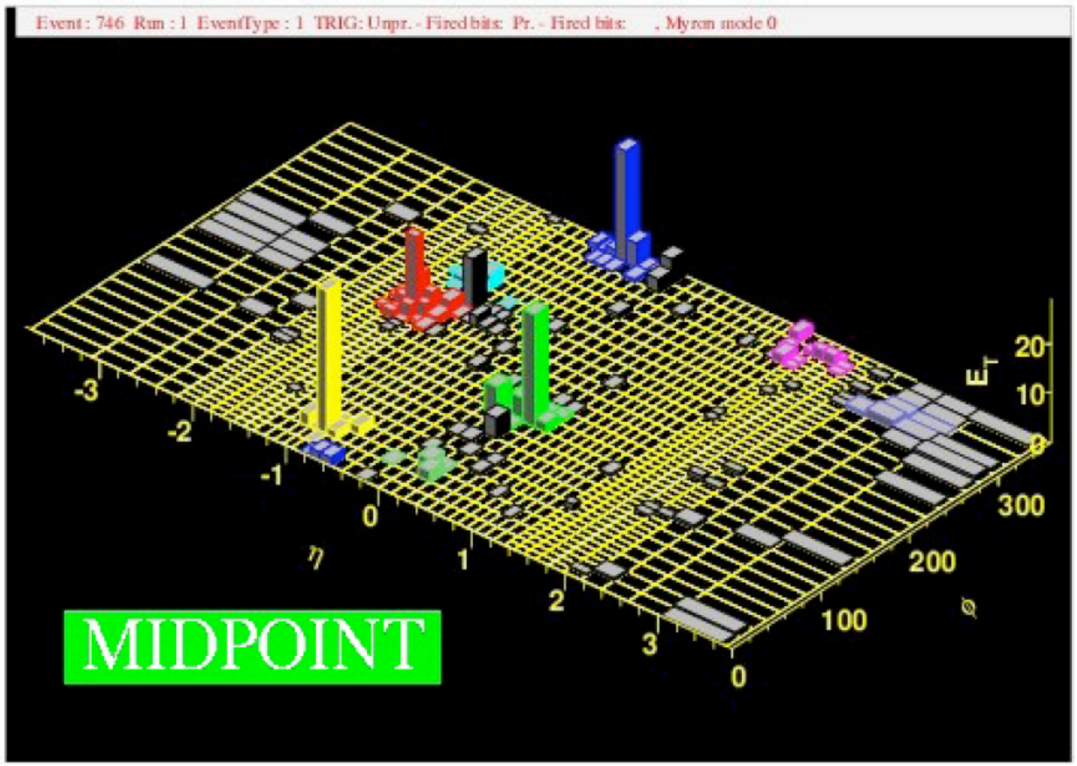
- Begin with 1 GeV seed towers
- Cluster towers with p_T > 100 MeV into jet if $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} < 0.7$
- Start new search cones at midpoint of stable cones
- Merge jets if overlapping energy is > 0.75 times the energy of the smaller jet
- Calculate jet quantities from stable cones

At least <u>four</u> more or less arbitrary parameters

Why k_t

More troubles:

Dark towers



Some of the energy is not collected in any jet

Solution (?!)

- Implement an initial search cone step with the search cone size = $R_{cone}/2$
- Less sensitive to effects of *great attractors* far away
- After stable cones are formed, expand jet cones to full size and decide whether to split/merge overlapping jets according to the standard criteria

(Cure worse than disease?)

[NB. Fifth parameter...]

[A sixth parameter is also introduced (by CDF only!) to tweak the NLO calculation when running the algorithm on theoretical results]

Yet more troubles: at the end of the game, the modified Midpoint algorithm (the 'search cone') might not even be infrared safe

Why k_t

The definition of a sequential clustering algorithm, on the other hand, is extremely simple.

For instance, take **the longitudinally invariant k_t:**

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

- Calculate the distances between the particles: $d_{ij} = \min(k_{ti}^2, k_{tj}^2)(\Delta\eta^2 + \Delta\phi^2)$
- Calculate the beam distances: $d_{iB} = k_{ti}^2$
- Combine particles with smallest distance or, if d_{iB} is smallest, call it a jet
- Find again smallest distance and repeat procedure until no particles are left

This definition is infrared/collinear safe, has no artificial parameters, does not lead to dark towers or overlapping jets, can be applied equally well to data and theory

Why FastJet

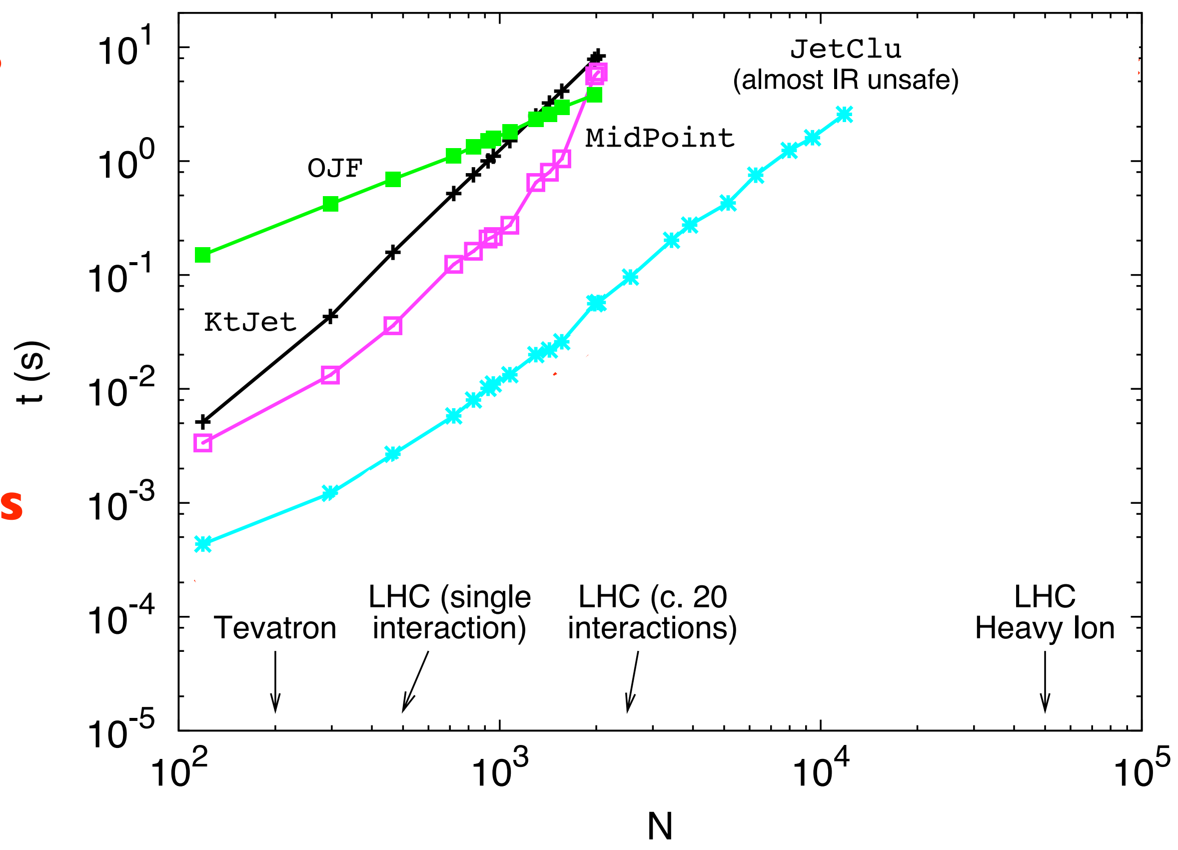
The k_t jet-finder has, however, an apparent drawback: finding all the distances is an N^2 operation, to be repeated N times

⇒ naively, the k_t algorithm scales like N^3

Time taken to cluster N particles:

10 s

1 ms



Clustering quickly gets very slow: processing millions of events at LHC is simply not feasible with standard clustering algorithms

e.g. clustering a single heavy ion event at LHC would take 1 day of CPU!

Why FastJet

To improve the speed of the algorithm we must find more efficiently which particle is "close" to another and therefore gets combined with it

Observation (MC, G.P. Salam, hep-ph/0512210):

If i and j form the smallest d_{ij} and $k_{ti} < k_{tj}$ $\Rightarrow$ $R_{ij} < R_{jk}$ $\forall\ k \neq j$

(Approximate) translation from mathematics:

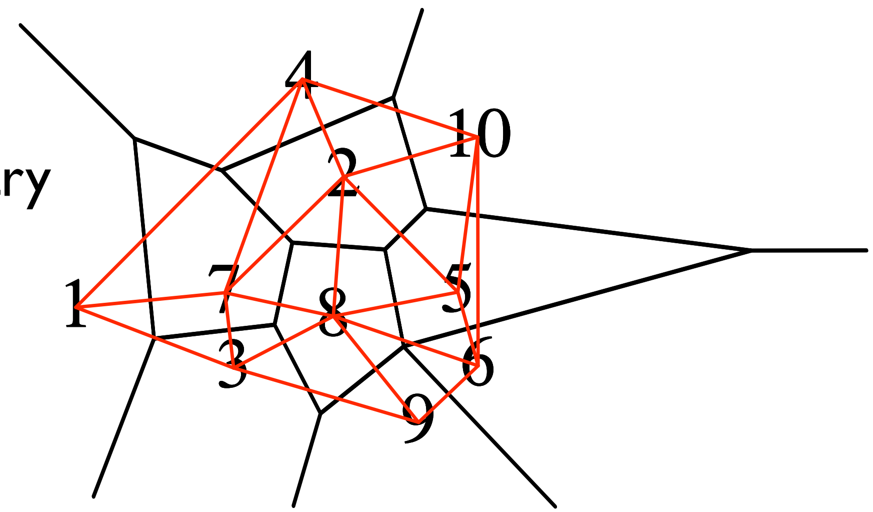
When a particle gets combined with another, its partner will be its **geometrical nearest neighbour** on the cylinder spanned by η and ϕ

This means that we need to look for partners only among the O(N) nearest neighbours of each particle

Why `FastJet`

Our problem has now become a **geometrical** one:
how to find the (nearest) neighbour(s) of a point

Widely studied problem in computational geometry
Tool: **Voronoi diagram**



Definition: each cell contains the locations which have the given point as nearest neighbour

The **dual** of a Voronoi diagram is a **Delaunay triangulation**

Once the Voronoi diagram is constructed, the nearest neighbour of a point will be in one of the O(1) cells sharing an edge with its own cell

Example : the G(eometrical) N(earest) N(eighbour) of point 7 will be found among 1,4,2,8 and 3 (it turns out to be 3)

Why `FastJet`

<u>The `FastJet` algorithm:</u>

MC and G.P. Salam, hep-ph/0512210

Construct the Voronoi diagram of the N particles using the CGAL library — **O(N lnN)**

Find the GNN of each of the N particles. Construct the d_{ij} distances, store the results in a map — **O(N lnN)**

Merge/eliminate particles appropriately

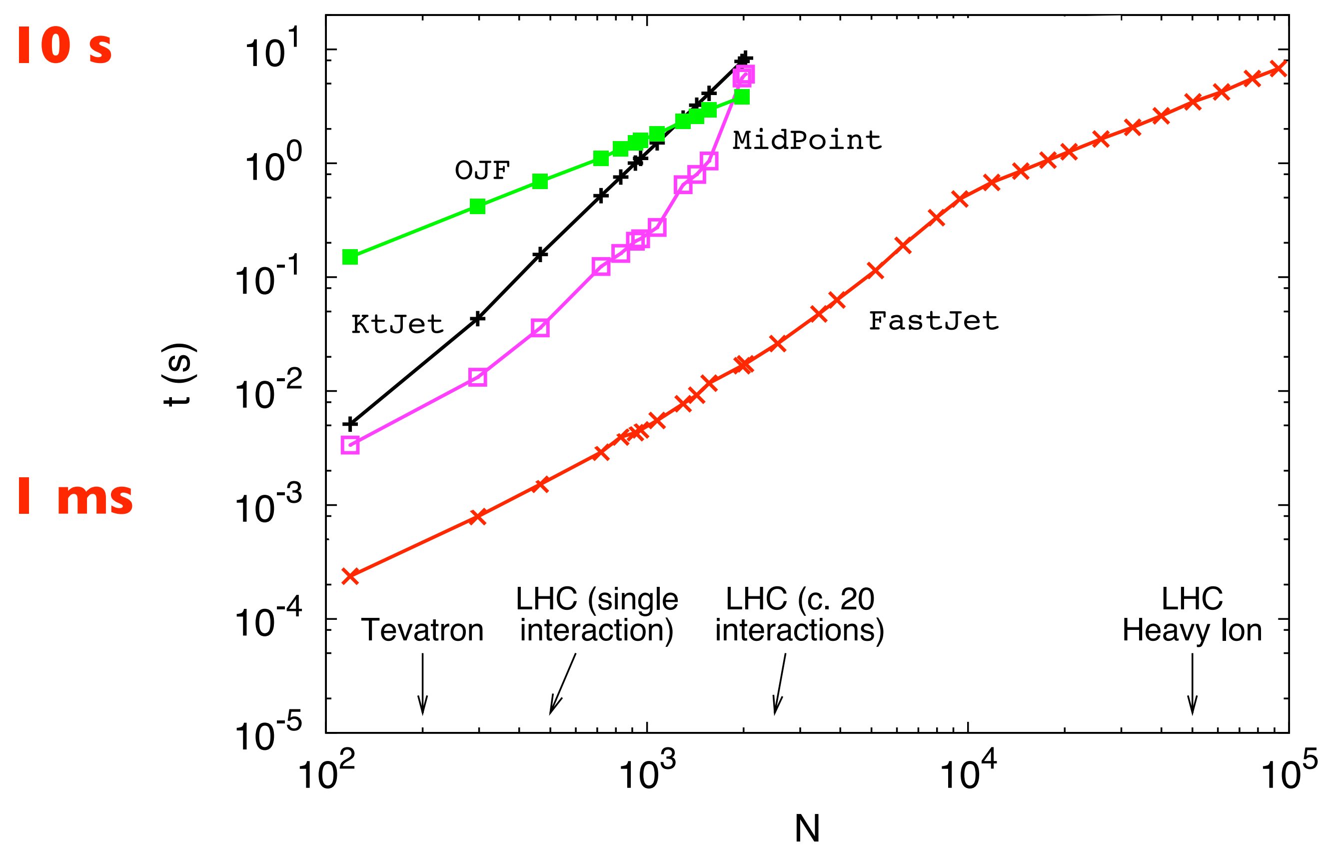
Update Voronoi diagram and distances' map — **O(lnN)**

repeat N times

<u>Overall, an O(N ln N) algorithm</u>

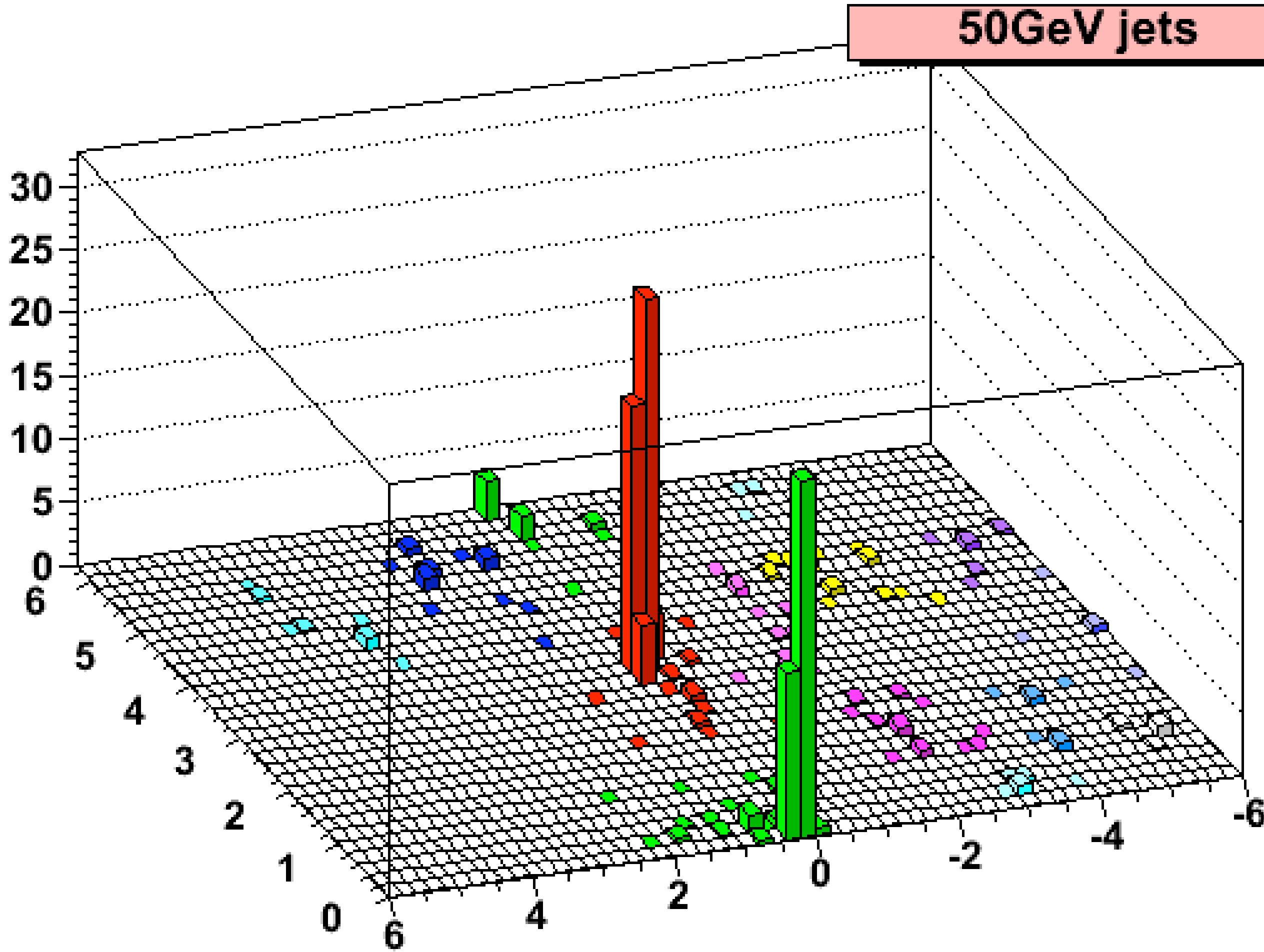
Why FastJet

Time taken to cluster N particles:

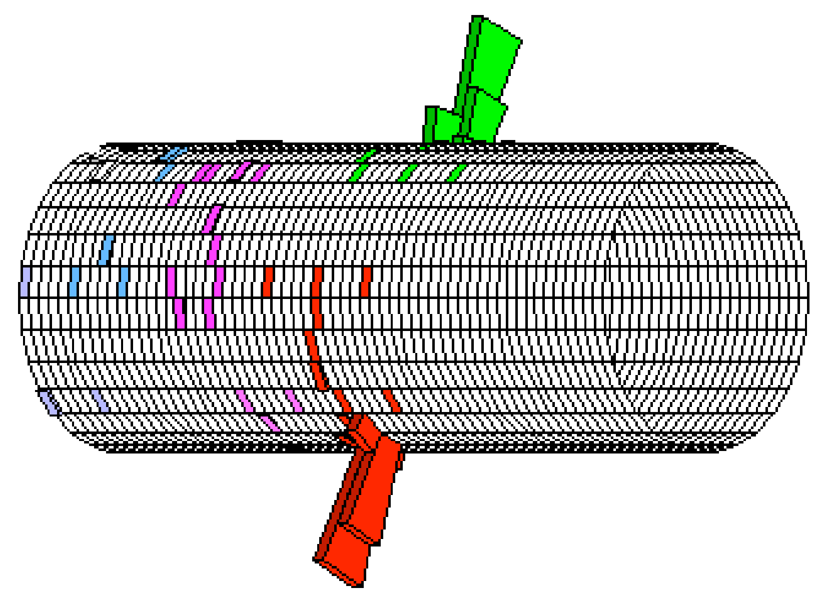


- Almost two orders of magnitude gain at small N (O(N^2) implementation)
- Large-N region now reachable

Why FastJet



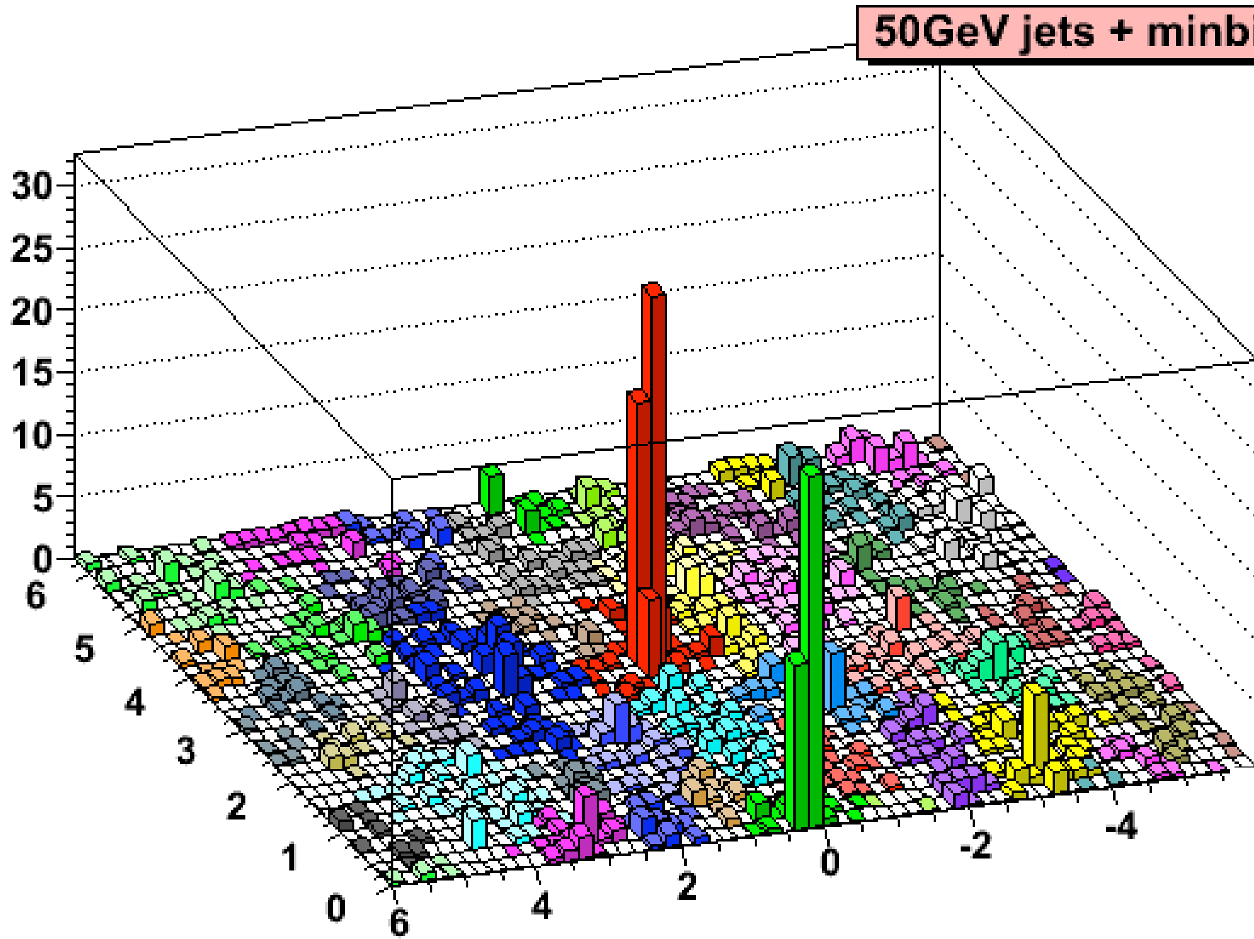
'Standard' hard event.
Two well isolated jets



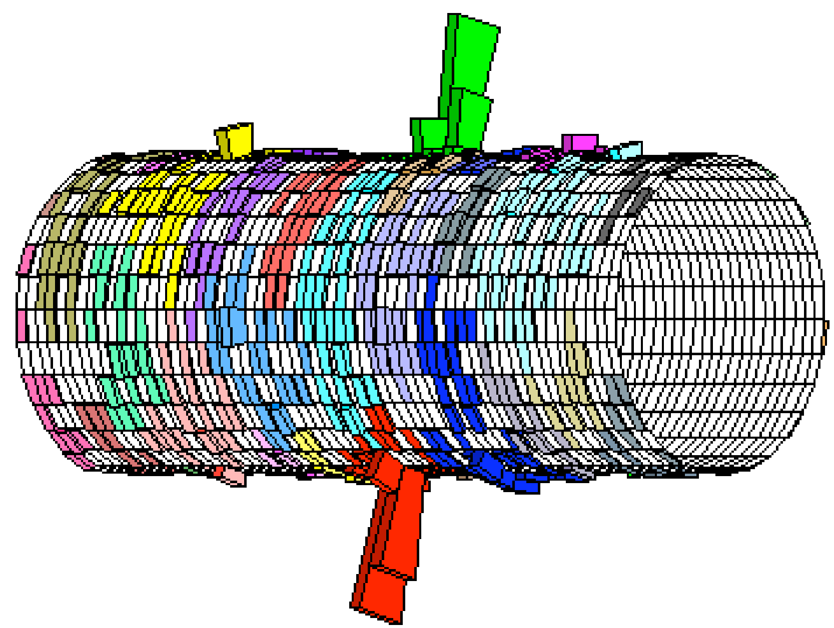
< 200 particles

Clustering easily doable even with standard algorithms

Why FastJet



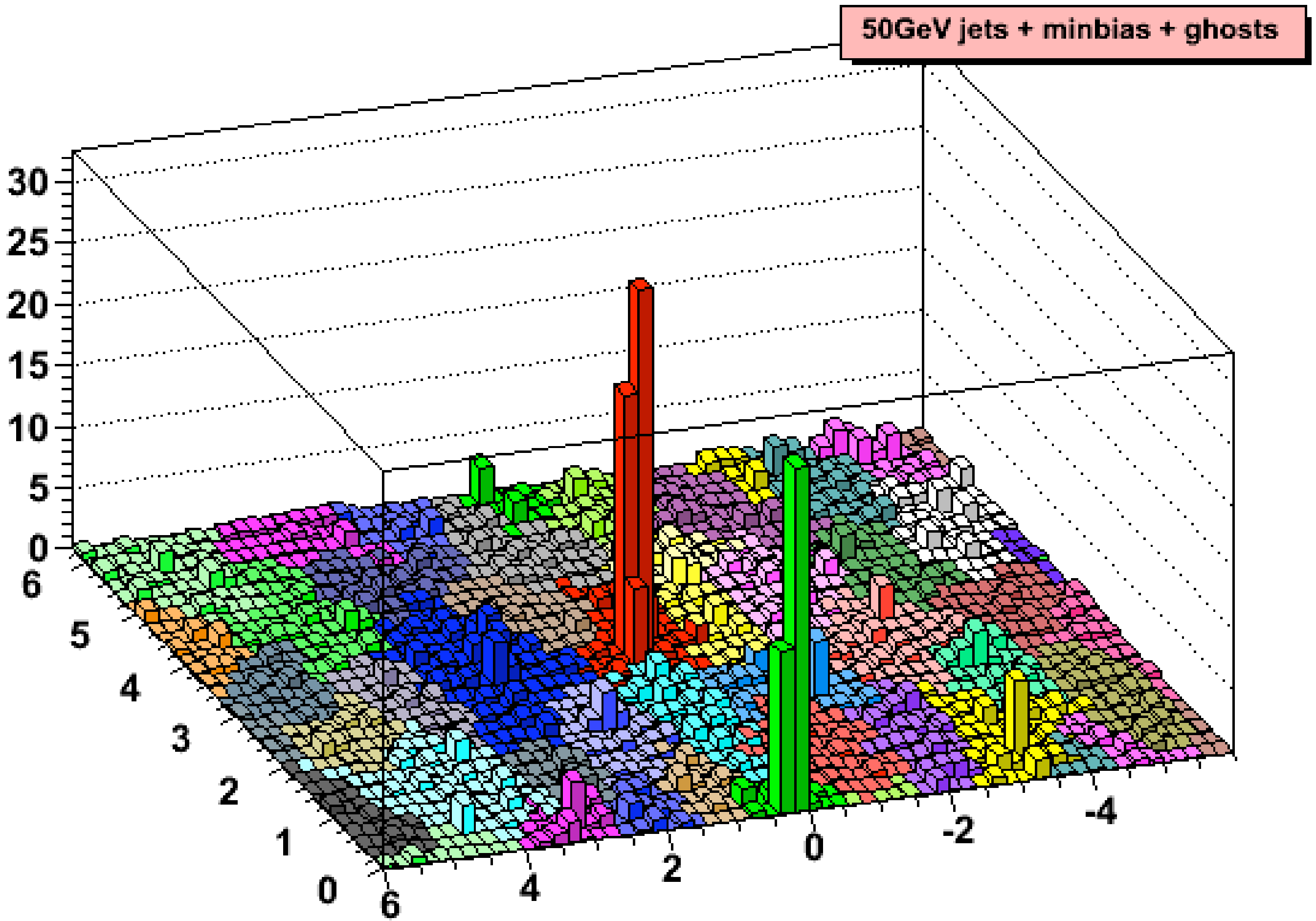
Add minimum bias



~ 2000 particles

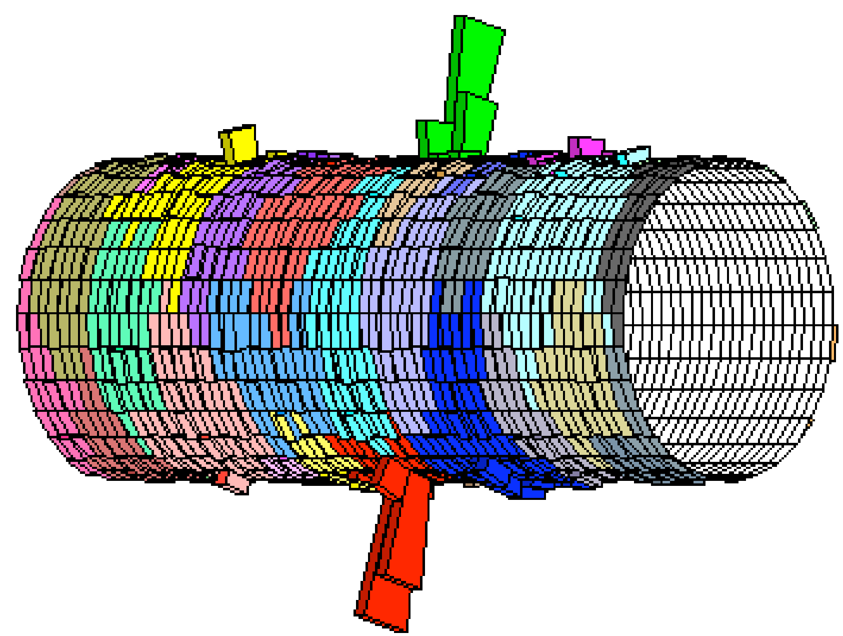
Clustering takes O(20 s) with standard algorithms, but only O(20 ms) with FastJet

Why FastJet



Try to estimate **area** of each jet

Fill event with many very soft particles, count how many are clustered into given jet

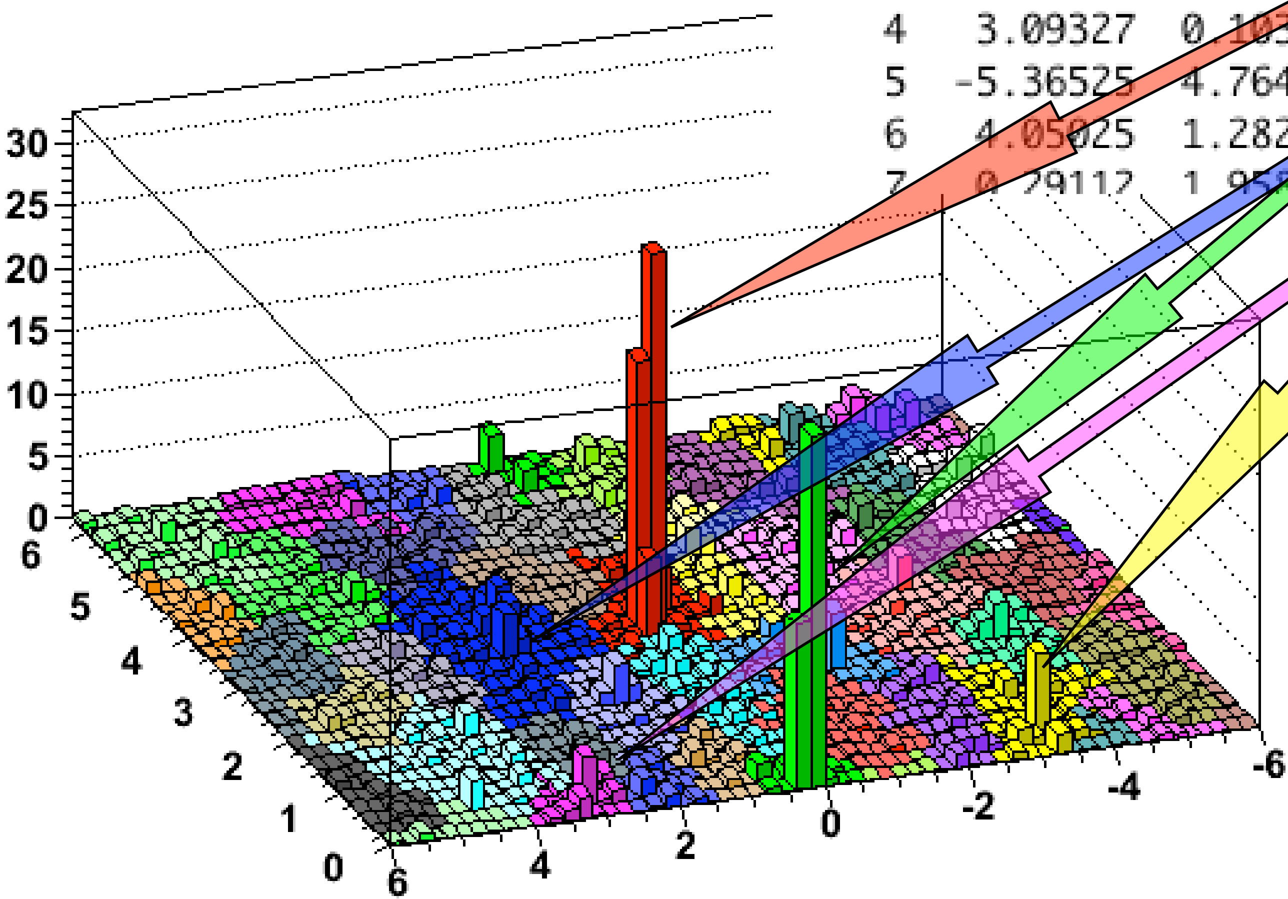


~ 10000 particles

Don't even think about it with standard algorithms, O(1 s) with FastJet

Why FastJet

```
iev 0 (irepeat 24): number of particles = 1428
strategy used =  NlnN
number of particles = 9051
Total area: 76.0265
Expected area: 76.0265
 ijet    eta        phi          Pt          area  +-   err
    0    0.15050    3.24498      69.970      2.625 +-   0.020
    1    0.18579    0.13150      59.133      1.896 +-   0.020
    2    2.33840    3.23960      31.976      4.749 +-   0.028
    3   -3.41796    0.52394      26.595      3.084 +-   0.021
    4    3.09327    0.10350      20.072      2.688 +-   0.023
    5   -5.36525    4.76491      19.598      2.780 +-   0.012
    6    4.05025    1.28279      15.561      3.592 +-   0.028
    7    0.79112    1.95845      14.566      2.114 +-   0.018
```



Approximate linear relation between Pt and area for minimum bias jets.

Can be used on an event-by-event basis to correct the hard jets

Conclusions

- `FastJet` written in C++ and available at **www.lpthe.jussieu.fr/~salam/fastjet**
- Extremely fast at small N, large N feasible (can cluster 50000 particles in O(3 s))
- Not a new clustering algorithm (results **IDENTICAL** to older and slower implementations of k_t)
- However, the high speed allows one to do **new** things. Among others, **cluster heavy ion events**, and study the **area of the jets**
- Full usefulness will only be clear with use. So, download it and run it!