QCD & Jets & MC Modeling

Matteo Cacciari
LPTHE Paris

Lecture 2

Many thanks to Guenther Dissertori, Rikkert Frederix, Fabio Maltoni, Paolo Nason, Gavin Salam, Gregory Soyez, Maria Ubiali, and probably others, from whose talks/lectures I have drawn inspiration, as well as extracted many slides

Outline

- Some basics of QCD
- Initial state
  - PDFs
- Hard scattering (and more)
  - higher order calculations and generators
    - Parton shower MCs
    - Merging
- Final state
  - Jets algorithms and jet areas
  - Jets as tools (jet substructure)

Lecture 1

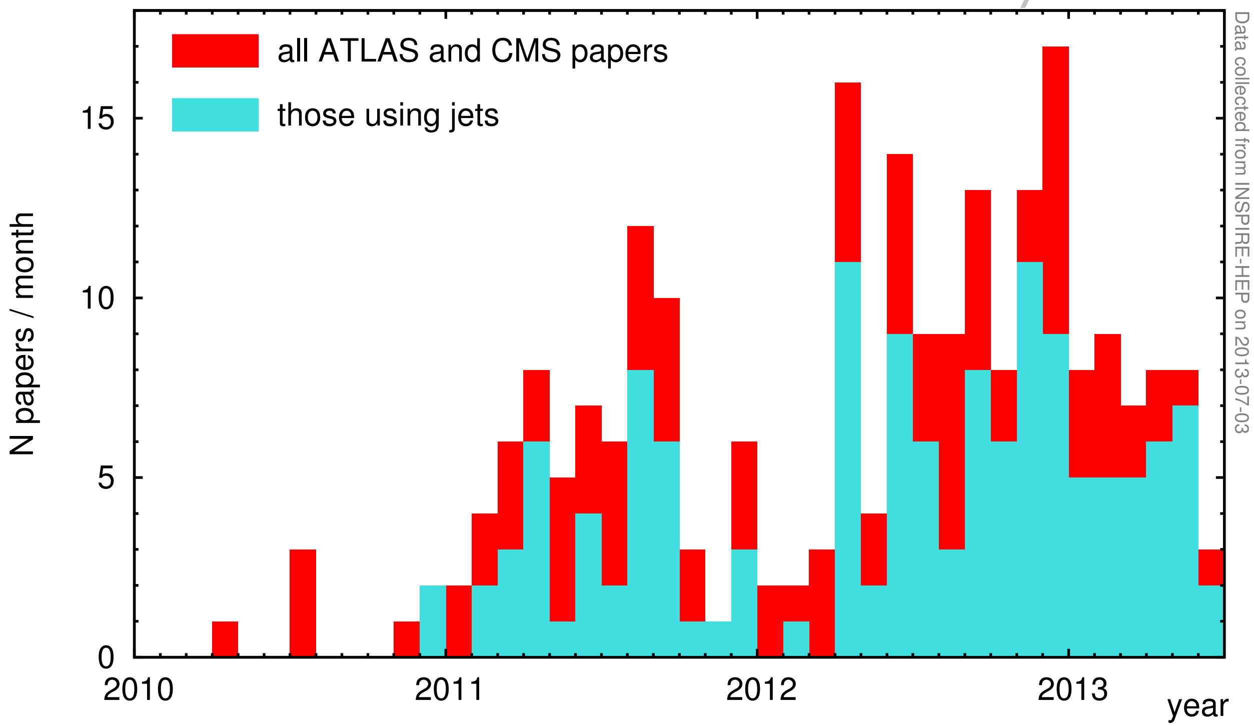
Lectures 2 and 3

[Subdivision in parts actually quite unreliable. Length/depth of descriptions varies quite a lot]

The pervasiveness of jets

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

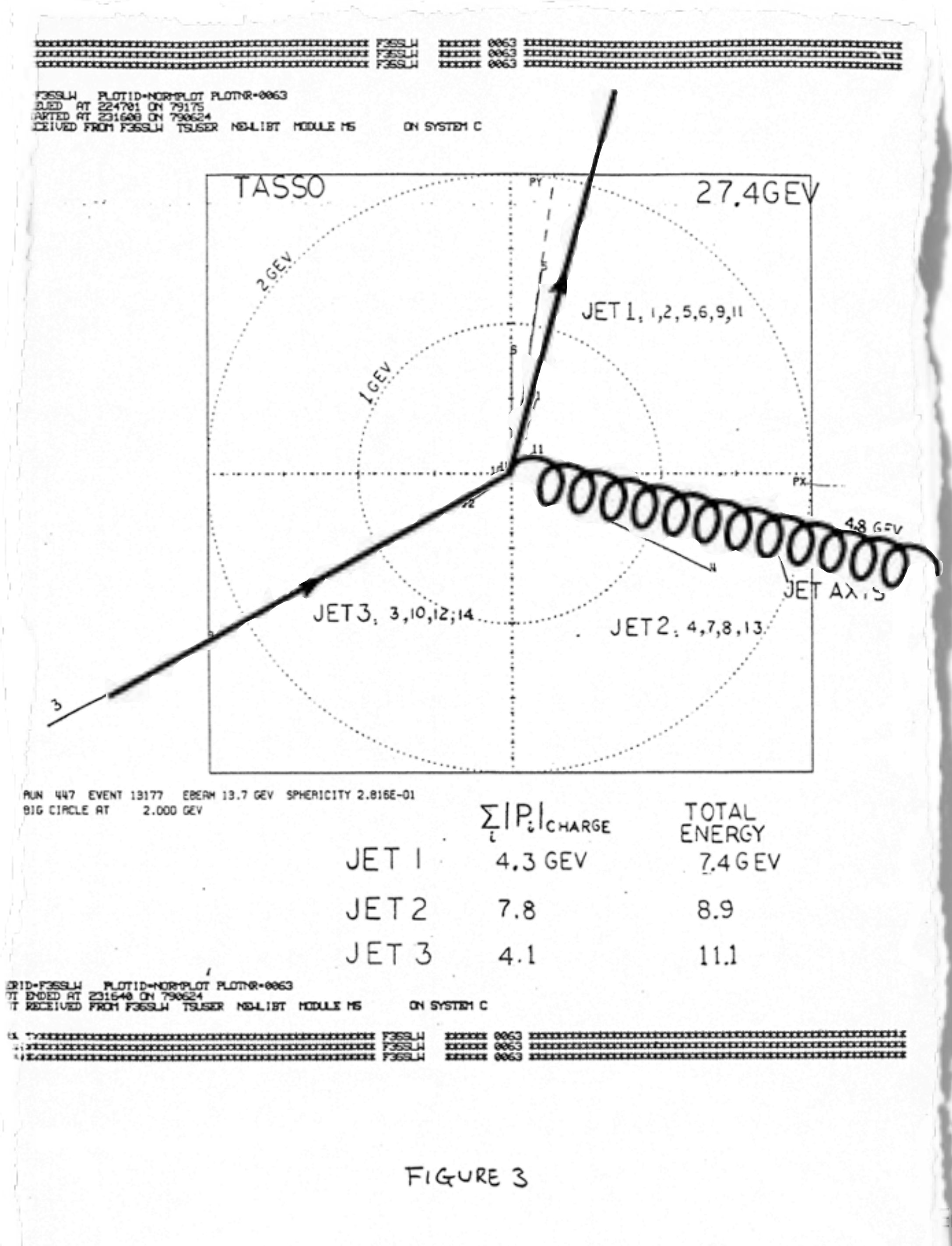
Plot by G. Salam



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

Why are jets so important?

Gluon ‘discovery’

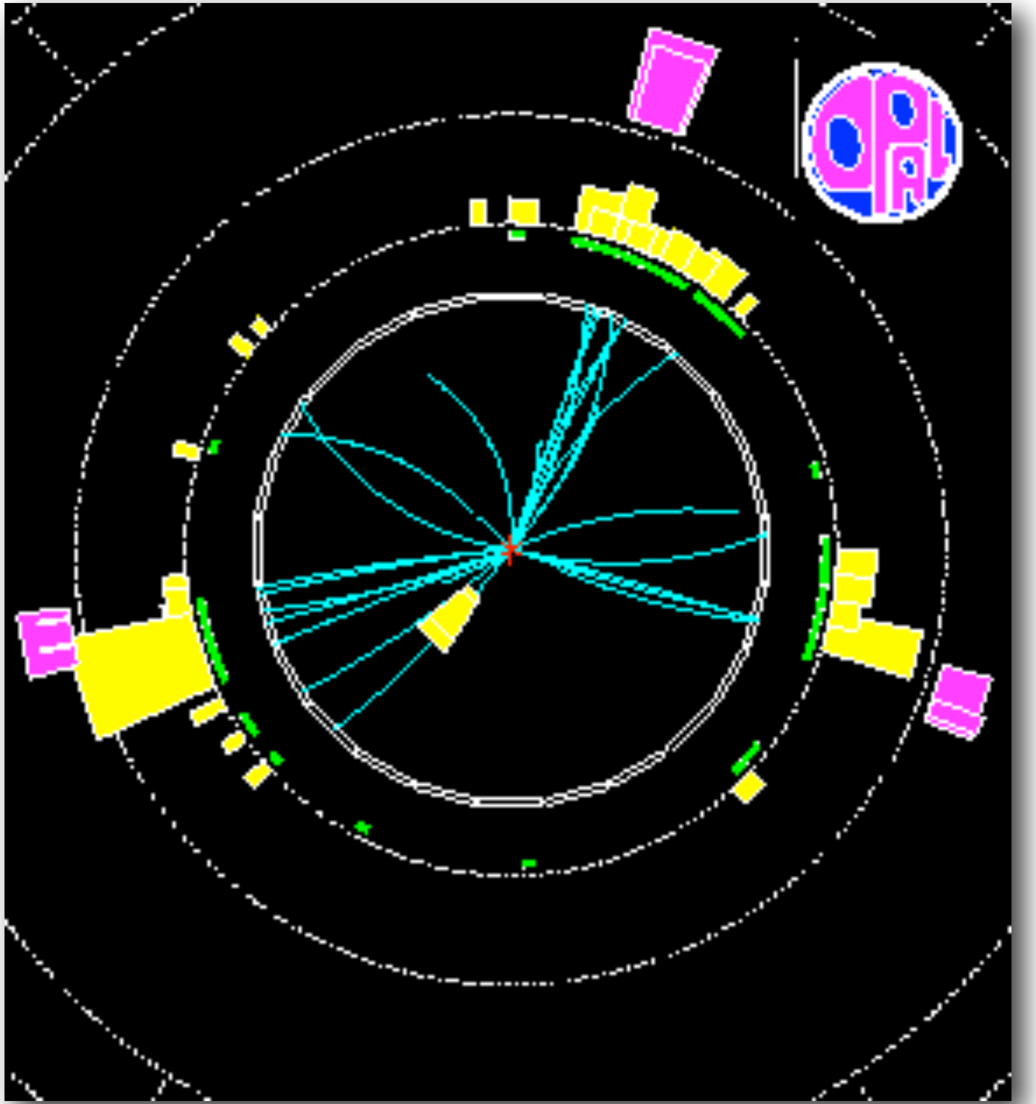


1979:

Three-jet events observed by TASSO, JADE, MARK J and PLUTO at PETRA in e^+e^- collisions at 27.4 GeV

Interpretation:
large angle emission of a hard gluon

Jets viewed as a proxy to the initial partons



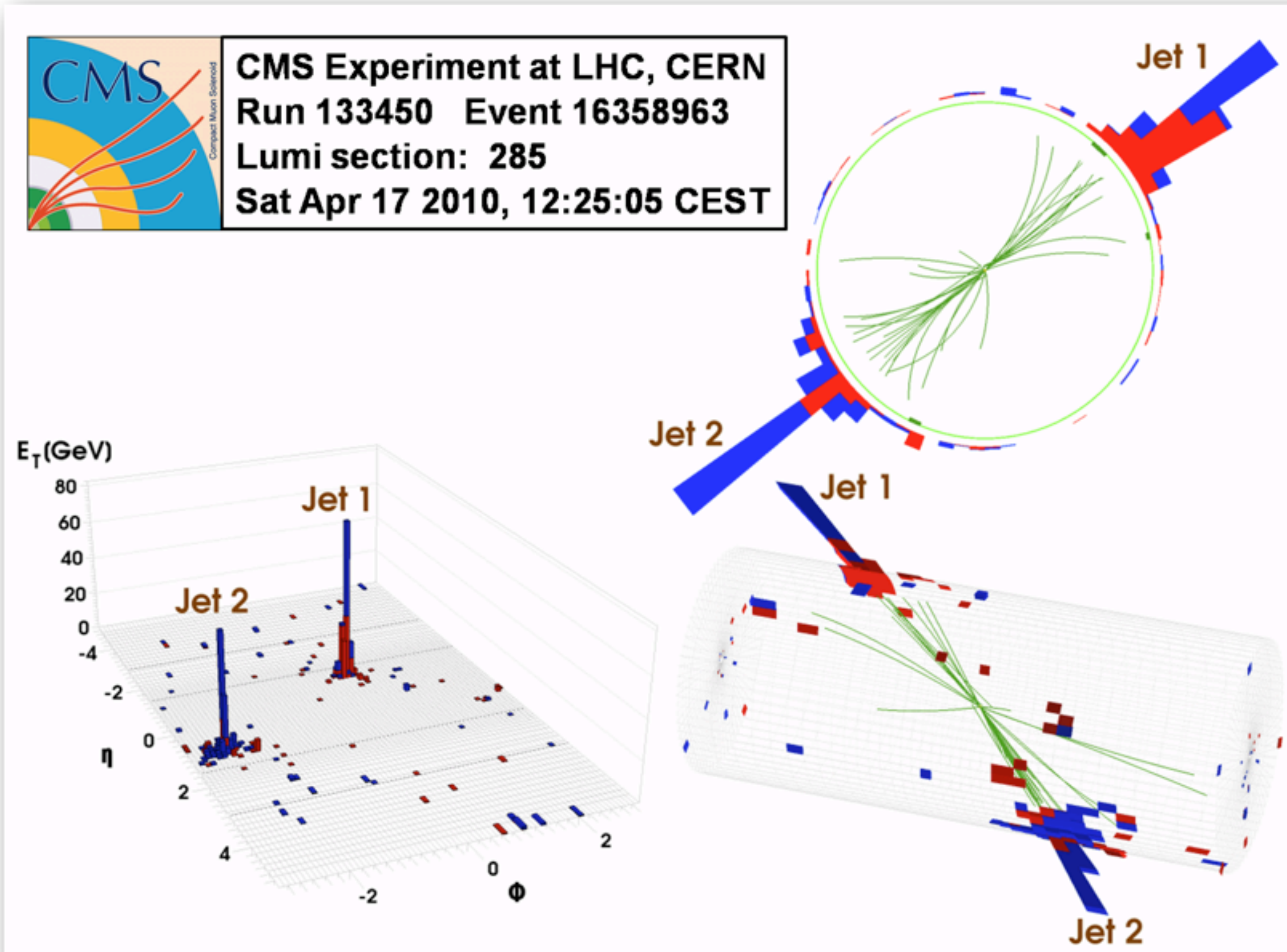
From PETRA to LEP

A **jet** is something that happens in high energy events: **a collimated bunch of hadrons flying roughly in the same direction**

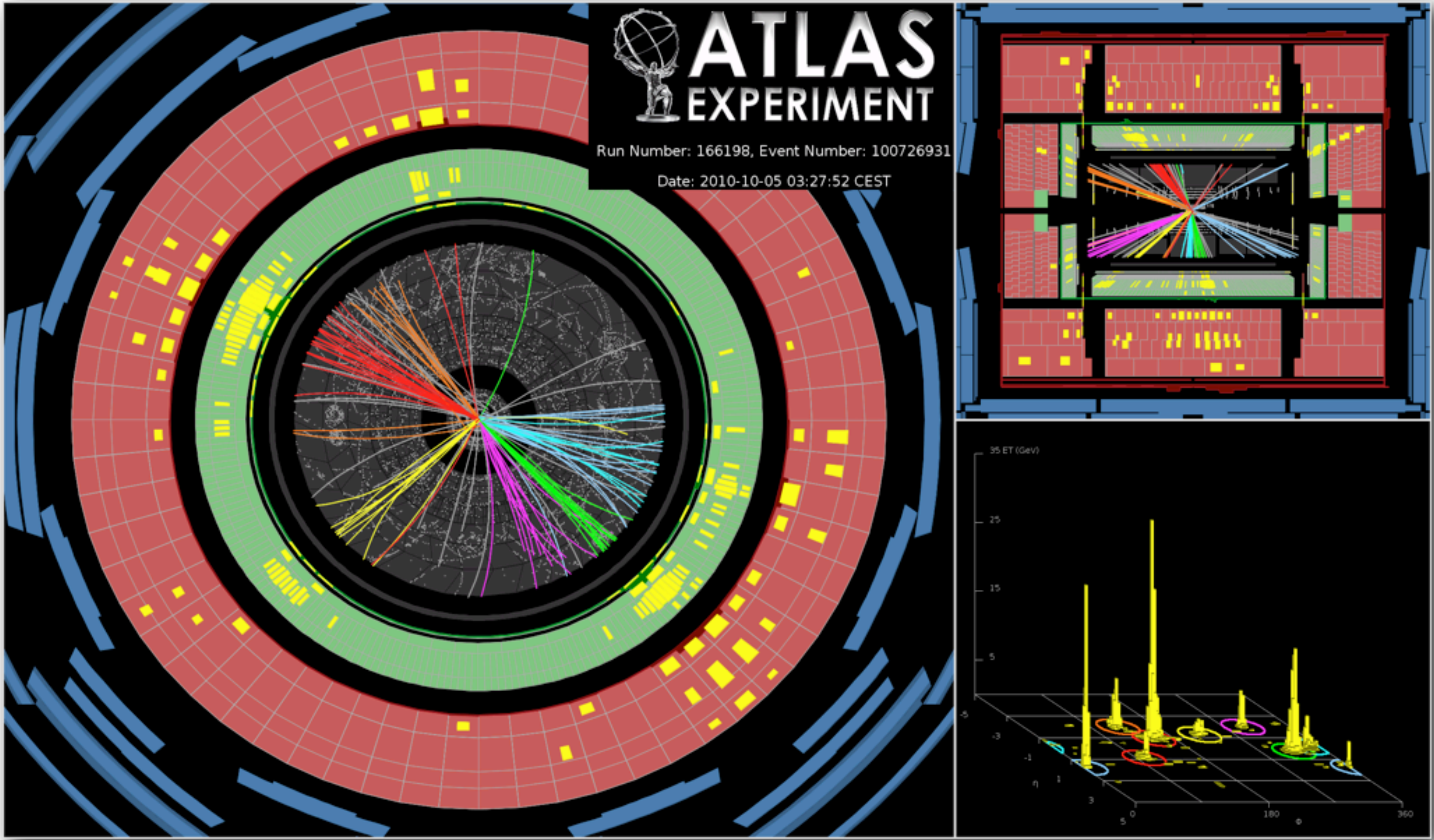
We could eyeball the collimated bunches, but it becomes impractical with millions of events

The classification of particles into jets is best done using a **clustering algorithm**

A few decades after PETRA and LEP, the event displays got prettier, but jets are still pretty much the same



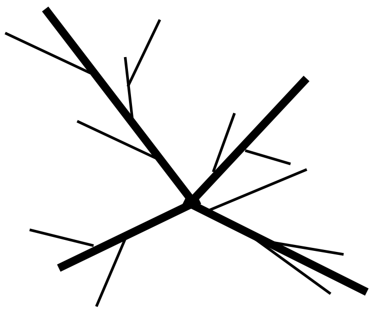
Dijet event from CMS



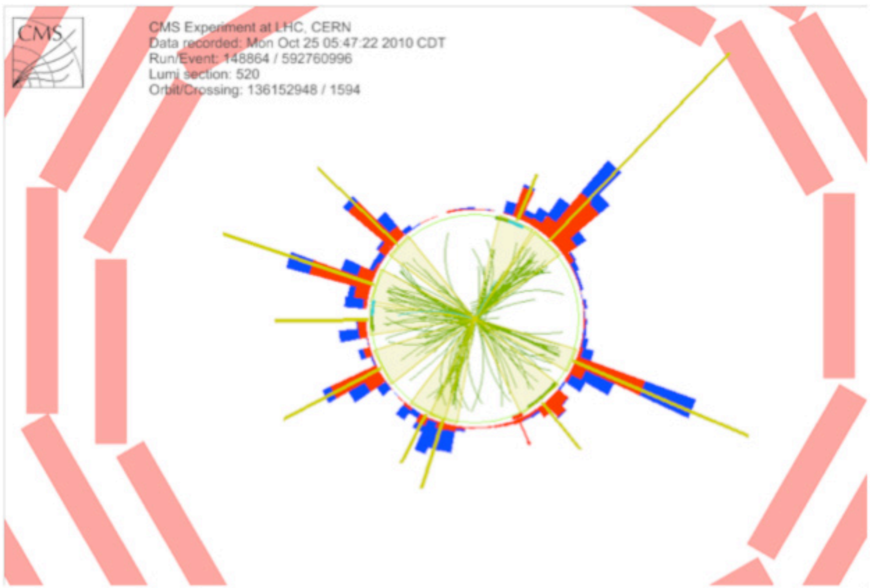
8(!) jets event from ATLAS

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**

Jets can serve two purposes

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

Different clustering algorithms have different properties and characteristics that can make them more or less appropriate for each of these tasks

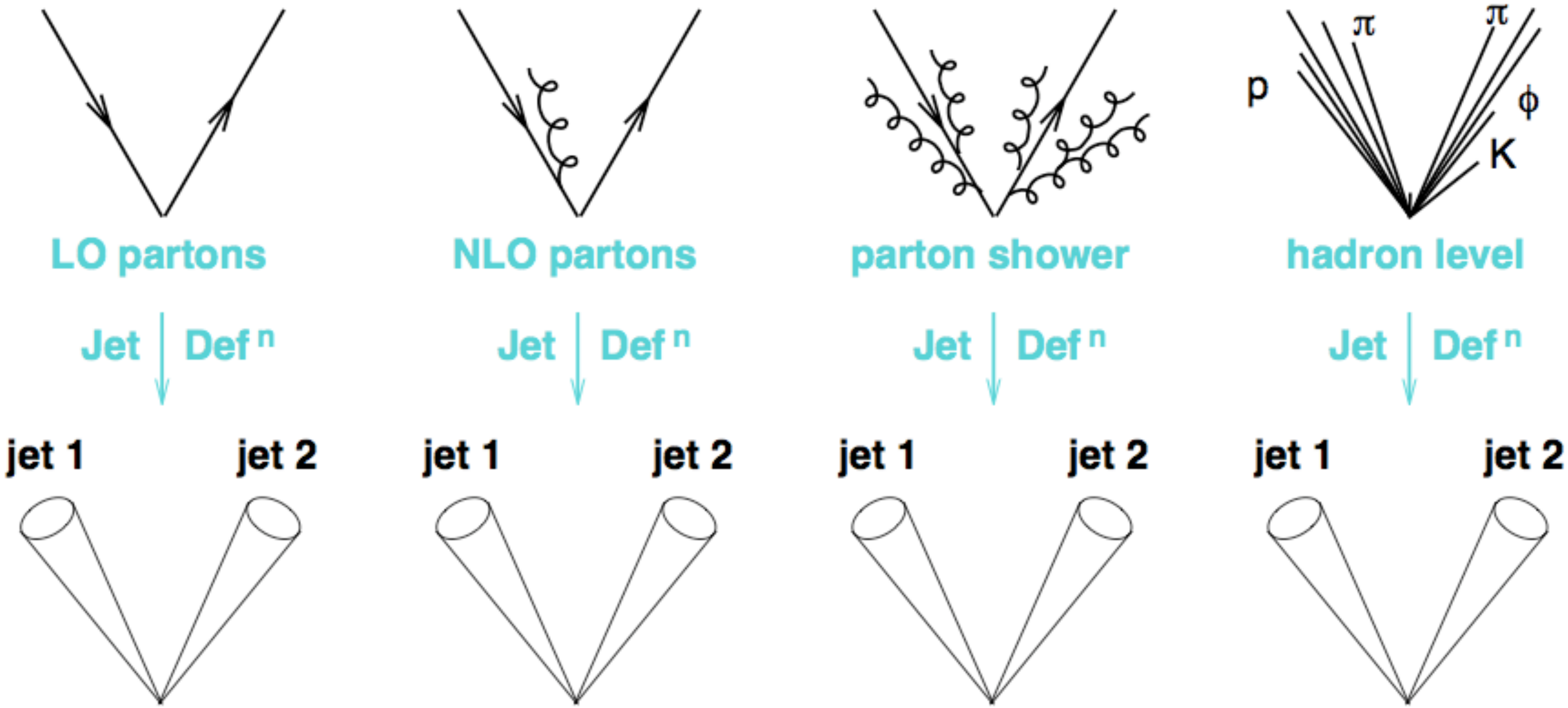
Jets in physics

- While we could take almost any clustering algorithm and, with a reasonable distance, use it to construct jets, i.e. clusters of hadrons, the result may not be particularly useful.
  We must also be **guided by physics**, so that
  - *the procedure leads to calculable results →infrared and collinear safety*
  - *the results are robust with respect to dynamics that we cannot calculate in detail → resiliency to hadronisation effects*

This puts strong constraints on the distances and algorithms that we can use

Resiliency to hadronisation effects

A good jet definition should be resilient to QCD effects



NB. 'Resiliency' does not mean 'total insensitivity'

A 'hadron jet' is **not** a parton

Most definitions will give very similar results (especially for inclusive observables), but it is important to be aware of potential differences, and not to compare apples with oranges.

IRC safety

Cancellation of singularities in total cross section

Born Virtual Real

$$\sigma_{tot} = \int_n |M_n^B|^2 d\Phi_n + \underbrace{\int_n |M_n^V|^2 d\Phi_n + \int_{n+1} |M_{n+1}^R|^2 d\Phi_{n+1}}_{\text{CANCELLATION}}$$

A generic observable

$$\frac{d\sigma}{dX} = \int_n |M_n^B|^2 O(X; p_1, \ldots, p_n) d\Phi_n$$
$$+ \int_n |M_n^V|^2 O(X; p_1, \ldots, p_n) d\Phi_n + \int_{n+1} |M_{n+1}^R|^2 O(X; p_1, \ldots, p_n, p_{n+1}) d\Phi_{n+1}$$

In order to ensure the same cancellation existing in σ_{tot}, the definition of the observable $O(X; p_1, \ldots p_{n+1})$ **must not affect** the soft/collinear limit of the real emission term $|M^R_{n+1}|^2$, because it is there that the real/virtual cancellation takes place

IRC safety

An observable is **infrared and collinear safe** if, in the limit of a **collinear splitting**, or the **emission of an infinitely soft** particle, the observable remains **unchanged:**

$$O(X;p_1,\ldots,p_n,p_{n+1}\to 0)\to O(X;p_1,\ldots,p_n)$$

$$O(X;p_1,\ldots,p_n\parallel p_{n+1})\to O(X;p_1,\ldots,p_n+p_{n+1})$$

If we wish to be able to calculate a jet rate in perturbative QCD the jet algorithm that we use must be IRC safe:
soft emissions and collinear splittings must not change the hard jets

IRC safety in real life

Strictly speaking, one needs IRC safety not so much to find jets, but to be able to calculate them in pQCD

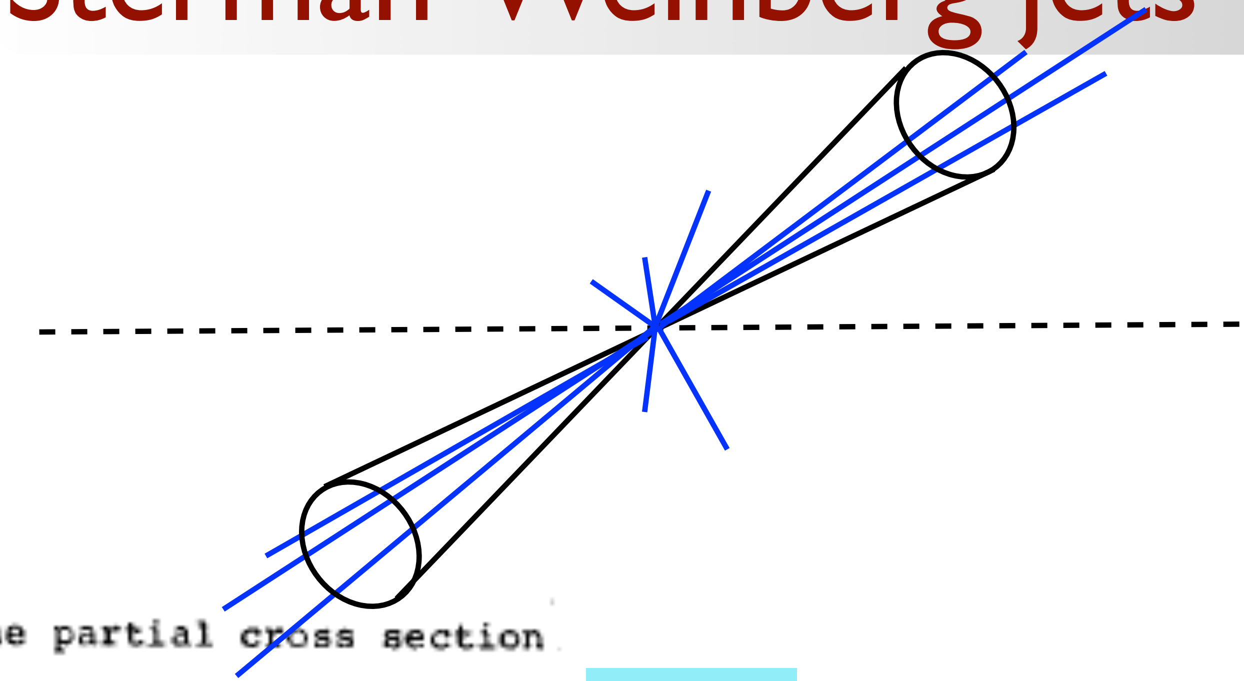
If you are not interested in theory/data comparisons, you may think of doing well enough with an IRC-unsafe jet algorithm

However

- Detectors may split/merge collinear particles, and be poorly understood for soft ones
- High luminosity (or heavy ions collisions) add a lot of soft particles to hard event

IRC safety provides resiliency to such effects
(plus, at some point in the future you may wish to compare your measurement to a calculation)

Sterman-Weinberg jets



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-

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

Calculable in pQCD (here is the result) but notice the soft and collinear large logs

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

$$\{p_i\} \xrightarrow{\text{jet algorithm}} \{j_k\}$$

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

jets

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

(more about this later on)

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

Two main classes of jet algorithms

- **Sequential recombination algorithms**

  Bottom-up approach: combine particles starting from **closest ones**

  How? Choose a **distance measure**, iterate recombination until few objects left, call them jets

  Works because of mapping closeness ⇔ QCD divergence

  Examples: Jade, k_t, Cambridge/Aachen, anti-k_t, …..

  → hierarchical clustering

- **Cone algorithms**

  Top-down approach: find coarse regions of energy flow.

  How? Find **stable cones** (i.e. their axis coincides with sum of momenta of particles in it)

  Works because QCD only modifies energy flow on small scales

  Examples: JetClu, MidPoint, ATLAS cone, CMS cone, SISCone…...

  → partitional clustering

Finding stable cones

In partitional-type algorithms (i.e. cones), one wishes to find the **stable configurations**:

axis of cones coincides with sum of 4-momenta of the particles it contains

The 'safe' way of doing so is to test **all possible combinations** of N objects

Unfortunately, this takes $N2^N$ operations:
the time taken is the age of the universe for less than 100 objects

An approximate way out is to use **seeds** (e.g. à la k-means)
However, the final result can depend on the choice of the seeds and, such jet algorithms usually turn out to be **IRC unsafe**

Finding stable cones

In partitional-type algorithms (i.e. cones), one wishes to find the **stable configurations**:

axis of cones coincides with sum of 4-momenta of the particles it contains

The 'safe' way of doing so is to test **all possible combinations** of N objects

Unfortunately, this takes $N2^N$ operations:
the time taken is the age of the universe for less than 100 objects

~~An approximate way out is to use **seeds** (e.g. à la k-means)~~
~~However, the final result can depend on the choice of the seeds and, such jet algorithms usually turn out to be **IRC unsafe**~~

SISCone is guaranteed to find **ALL stable cones**
(via a **fast** geometric implementation, G. Salam and G. Soyez 0704.0292)
and therefore leads to an **IRC-safe** cone jet algorithm

Recombination algorithms

These are hierarchical clustering algorithms

- First introduced in e^+e^- collisions in the '80s
- Typically they work by calculating a **'distance'** between particles, and then recombine them pairwise according to a given order, until some condition is met (e.g. no particles are left, or the distance crosses a given threshold)

IRC safety can usually be seen to be trivially guaranteed

JADE algorithm

distance:

$$y_{ij} = \frac{2E_iE_j(1-\cos\theta_{ij})}{Q^2}$$

- Find the minimum y_{min} of all y_{ij}
- If y_{min} is below some jet resolution threshold y_{cut}, recombine i and j into a single new particle ('pseudojet'), and repeat
- If no $y_{min} < y_{cut}$ are left, all remaining particles are jets

Problem of this particular algorithm:
two soft particles emitted at large angle get easily recombined into a single jet:
counterintuitive and perturbatively troublesome

e^+e^- k_t (Durham) algorithm

[Catani, Dokshitzer, Olsson, Turnock, Webber '91]

Identical to JADE, but with distance:

$$y_{ij} = \frac{2\min(E_i^2, E_j^2)(1-\cos\theta_{ij})}{Q^2}$$

In the collinear limit, the numerator reduces to the **relative transverse momentum** (squared) of the two particles, hence the name of the algorithm

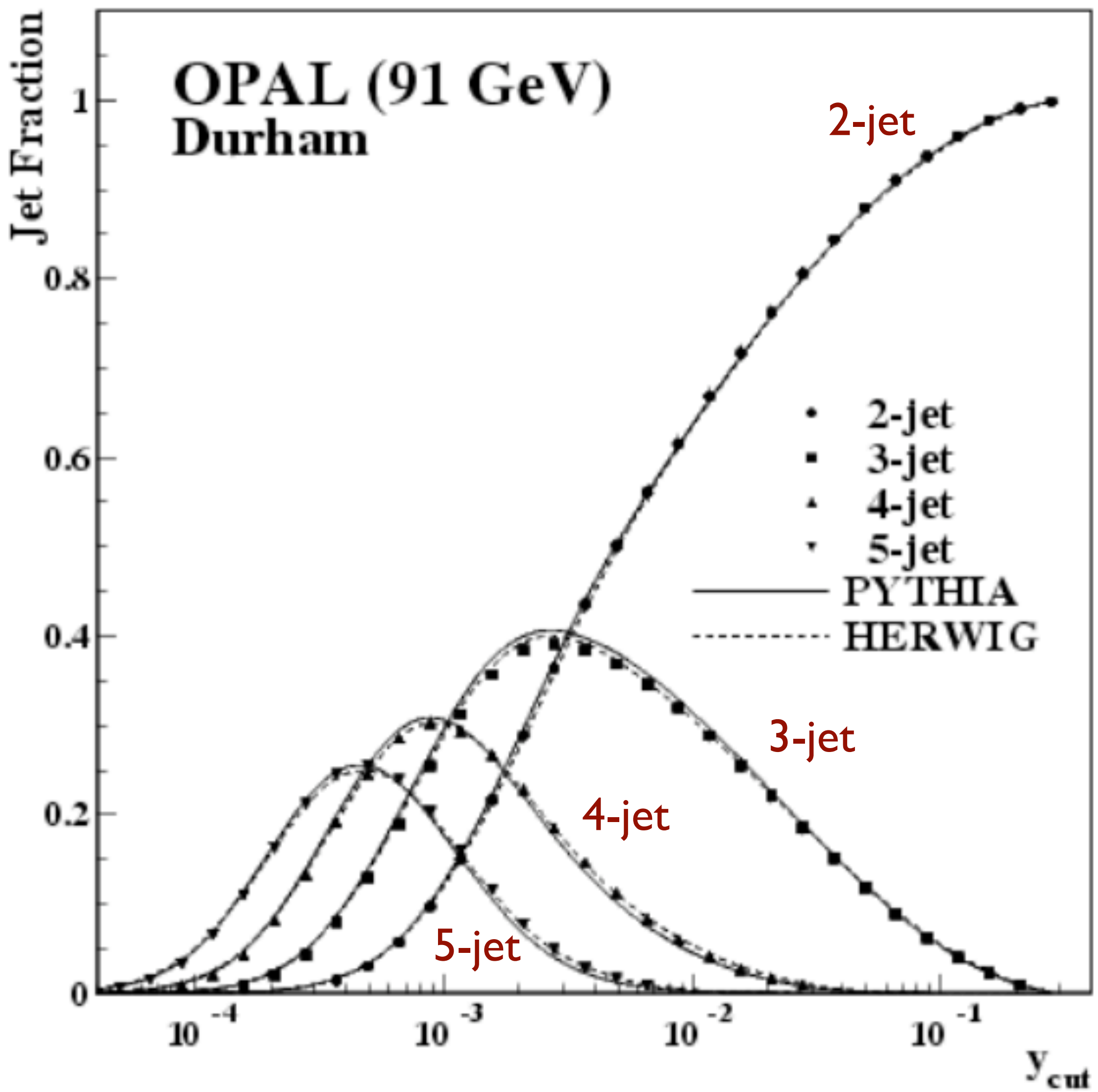
The use of the min() avoids the problem of recombination of back-to-back particles present in JADE: a soft and a hard particle close in angle are 'closer' than two soft ones at large angle

One key feature of the k_t algorithm is its relation to the structure of QCD divergences:

$$\frac{dP_{k\to ij}}{dE_i d\theta_{ij}} \sim \frac{\alpha_s}{\min(E_i, E_j)\theta_{ij}}$$

The k_t algorithm inverts the QCD branching sequence (the pair which is recombined first is the one with the largest probability to have branched)

e^+e^- k_t (Durham) algorithm in action



Characterise events
in terms of number of jets
(as a function of y_{cut})

The smaller ycut is,
the more 'jetty' the event looks

Resummed calculations for distributions of y_{cut} doable with the k_t algorithm

k_t algorithm in hadron collisions

(Inclusive and longitudinally invariant version)

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

- Calculate the distances between the particles: $\mathbf{d_{ij}}$
- Calculate the beam distances: $\mathbf{d_{iB}}$
- Combine particles with **smallest distance** d_{ij} or, if d_{iB} is smallest, call it a jet
- Find again smallest distance and repeat procedure until no particles are left (this stopping criterion leads to the ***inclusive*** version of the k_t algorithm)

- Given N particles this is, naively, an $O(N^3)$ algorithm: calculate N^2 distances, repeat for all N iterations. 1 second to cluster 1000 particles: too slow for practical use.
- An $O(N^2)$ implementation (the 'FastJet algorithm') exists: 1ms for 1000 particles. Can even use it in the trigger.
  - The same algorithm can also be implemented with only O(NlnN) complexity.

The speed 'problem'

Given N particles the k_t algorithm is, naively, an $O(N^3)$ algorithm: calculate N^2 distances, repeat for all N iterations

With 1000 particles (typical LHC event), this takes 10^9 operations, i.e. about a second on a modern GHz CPU

Clustering such an event would take significantly more than generating it in a MonteCarlo, not to speak about trying to use the algorithm at the trigger level, where the time budget is of the order of tens of milliseconds

This, together with the tendency of the k_t algorithm to 'scoop up' soft radiation quite far from the hard partons, and to give jets with ragged borders, difficult to correct for, had led people to prefer cone algorithms in a hadronic environment

The FastJet algorithm

Sequential recombination algorithms are computationally heavy because one naively calculates **all distances between all particles** ($O(N^2)$ step), before recombining them ($O(N)$ step)

Considering the problem from a **geometrical** point of view, one realizes that, in the k_t algorithm, when a particle gets combined with another, and has the smallest k_t, **its partner is its geometrical nearest neighbour on the cylinder spanned by y and ϕ**

This means that we need to look for partners only among the near neighbours of all particles:
a few neighbours each × N particles = O(N) operations

The FastJet algorithm

The FastJet algorithm is at least an **O(N²) algorithm**: find a few neighbours for each of the N particles, and repeat N times to recombine them all.

One can however do even better by resorting to computational geometry techniques (e.g. Voronoi diagrams) that allow one to find the near neighbours of N particles in **NlnN** time and update this situation while clustering in **lnN** time

FJ: the Voronoi implementation

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

Construct the Voronoi diagram of the N particles
(e.g. 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 priority queue (i.e. a C++ map)

O(N lnN)

Merge/eliminate particles appropriately

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

repeat N times

Overall, an O(N ln N) algorithm

The k_t algorithm and its siblings

One can generalise the k_t distance measure:

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

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**

Quite ironically, a sequential recombination algorithm is the 'perfect' cone algorithm

IRC safe algorithms

k_t	SR $d_{ij} = \min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2/R^2$ hierarchical in rel p_t	Catani et al '91 Ellis, Soper '93	NlnN
Cambridge/Aachen	SR $d_{ij} = \Delta R_{ij}^2/R^2$ hierarchical in angle	Dokshitzer et al '97 Wengler, Wobish '98	NlnN
anti-k_t	SR $d_{ij} = \min(k_{ti}^{-2}, k_{tj}^{-2})\Delta R_{ij}^2/R^2$ gives perfectly conical hard jets	MC, Salam, Soyez '08 (Delsart, Loch)	$N^{3/2}$
SISCone	Seedless iterative cone with split-merge gives 'economical' jets	Salam, Soyez '07	$N^2 \ln N$

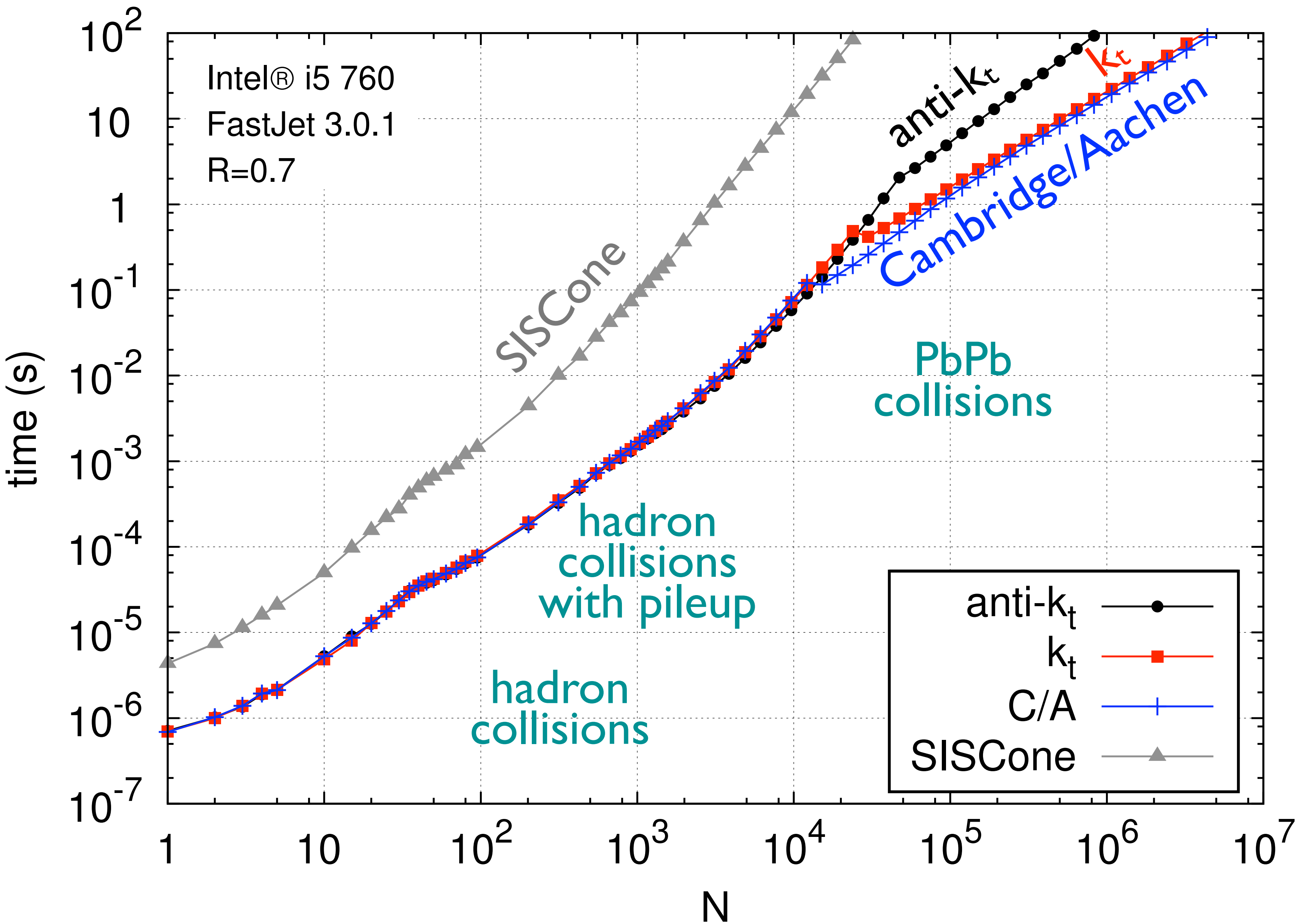
'second-generation' algorithms

All are available in FastJet, http://fastjet.fr

(As well as many IRC unsafe ones)

FastJet speed test

Time needed to cluster an event with N particles



The IRC safe algorithms

	Speed	Regularity	UE contamination	Backreaction	Hierarchical substructure
k_t	☺ ☺ ☺				
Cambridge /Aachen	☺ ☺ ☺				
anti-k_t	☺ ☺ ☺				
SISCone	☺				

Jets 'reach'

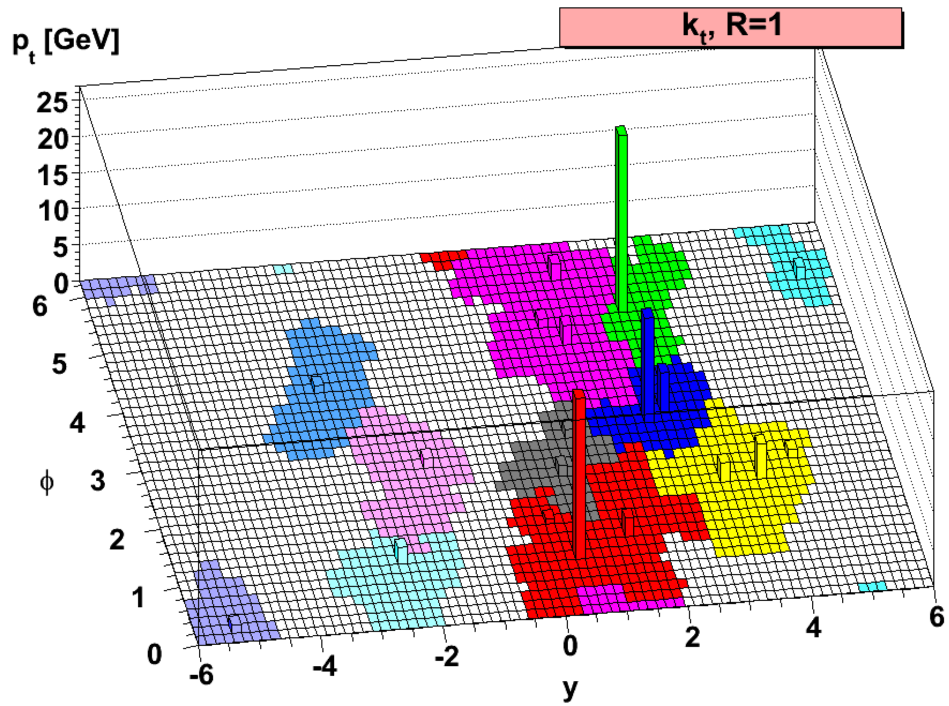
Algorithmically, a jet is simply a collection of particles

For a number of reasons, it is however useful to consider its **spatial extent**, i.e. given the position of its axis, up to where does it collect particles? What is its shape?

These details are important for a number of corrections of various origin: perturbative, non-perturbative (hadronisation), pileup, detector related, etc

Note that the intuitive picture of a jet being a cone (of radius R) is **wrong.** This is what k_t jets can look like:

(more later about what this plot really means)



From jet 'reach' to jet areas

MC, Salam, Soyez, 0802.1188

Not one, but three **definitions** of a jet's size:

- **Passive** area

  Place a single very soft particle (a '**ghost**') in the event, measure the extent of the region where it gets clustered within a given jet

  Reach of jet for **pointlike** radiation

- **Active** area

  Fill the events with many very soft particles ('**ghosts**'), cluster them together with the hard ones, see how many get clustered within a given jet

  Reach of jet for **diffuse** radiation

- **Voronoi** area

  Sum of areas of intersections of Voronoi cells of jet constituents with circle of radius R centred on each constituent

  Coincides with passive area for k_t algorithm

(In the large number of particles limit all areas converge to the same value)

Active Area

Add **many** ghost particles in random configurations to the event.
Cluster many times. *Allow ghosts to cluster among themselves too.*
Count how many ghosts on average get clustered into a given jet J.

Number of ghosts in jet J

$$A(J\,|\,\{g_i\}) = \frac{N_g(J)}{\nu_g} \equiv A_g N_g(J)$$

Active area of jet J for a single ghosts configuration

Ghost density

Area of a single ghost

$$A(J) = \lim_{\nu_g \to \infty} \langle A(J\,|\,\{g_i\}) \rangle_g$$

Active area of jet J

Jet active area

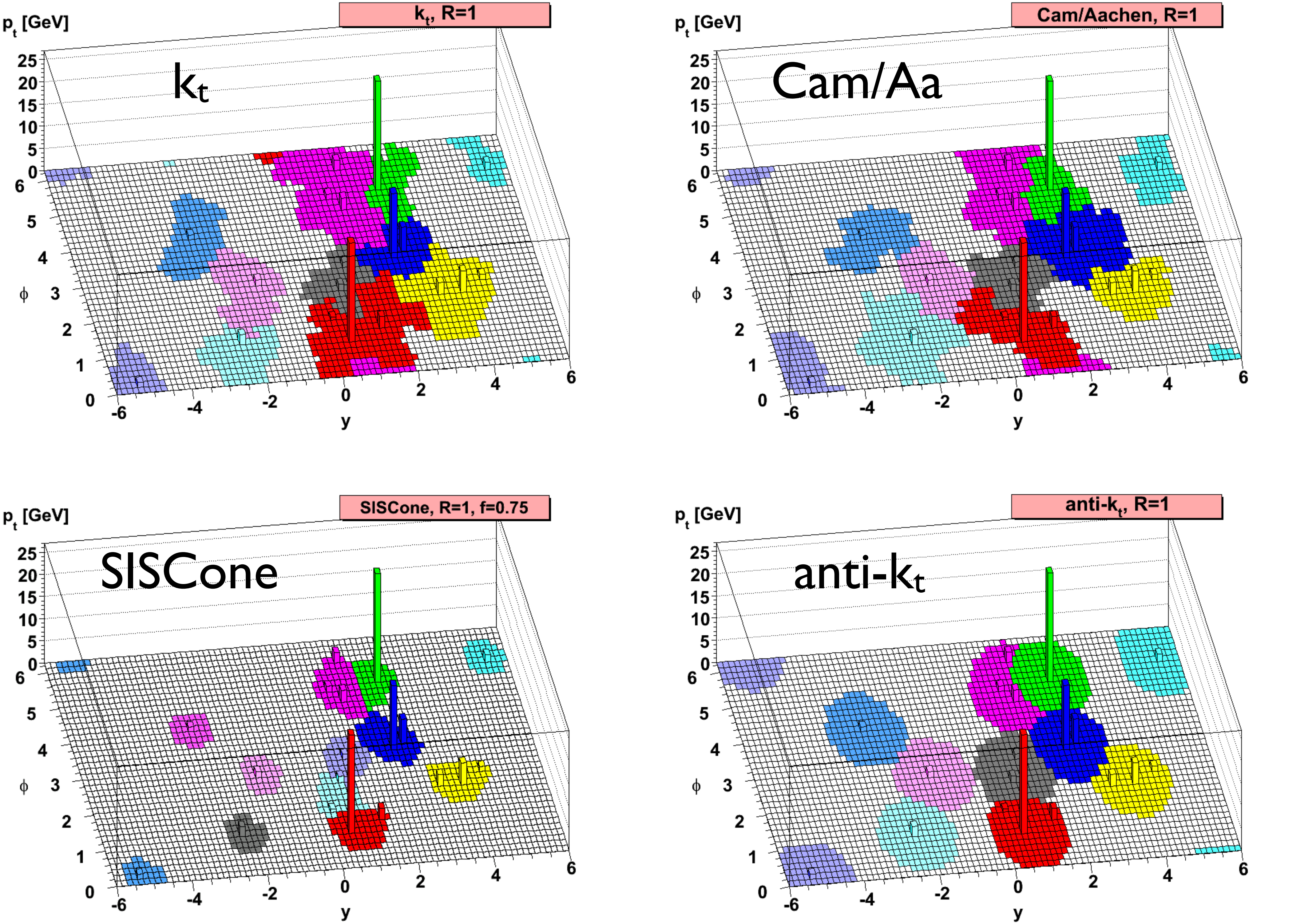
The definition of **active area** mimics the behaviour of the jet-clustering algorithms in the presence of a **large number of randomly distributed soft particles,** like those due to **pileup or underlying event**

Tools needed to implement it

1. An **infrared safe jet algorithm** (the ghosts should not change the jets)
2. A reasonably **fast implementation** (we are adding thousands of ghosts)

Both are available

As a bonus, active areas also allow for a **visualisation** of a jet's reach


kt, R=1
kt
Cam/Aachen, R=1
Cam/Aa
SISCone, R=1, f=0.75
SISCone
anti-kt, R=1
anti-kt
pt [GeV]
φ
y

Jet areas: the single hard particle case

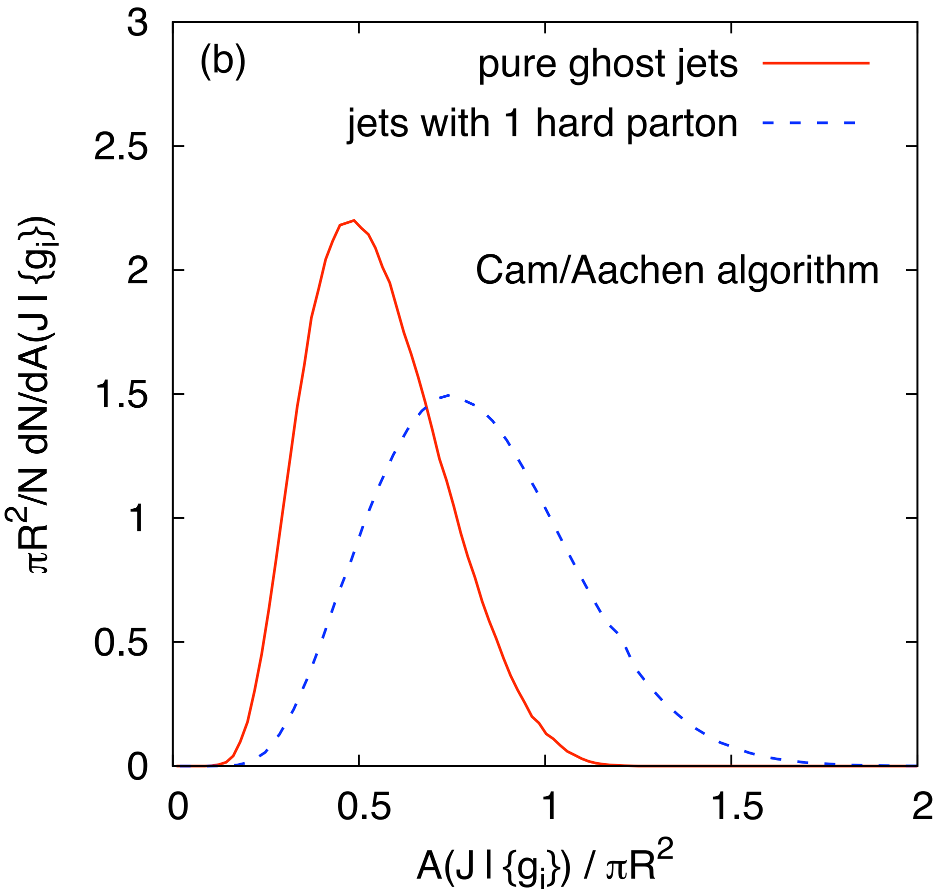
It is worth noting that, for a jet made of a **single hard particle**, while **passive** areas are indeed **πR^2, active** areas are **not**

Active areas	k_t	Cam/Aa	SISCone	anti-k_t
$\langle A\rangle/\pi R^2$	0.81	0.81	1/4	1

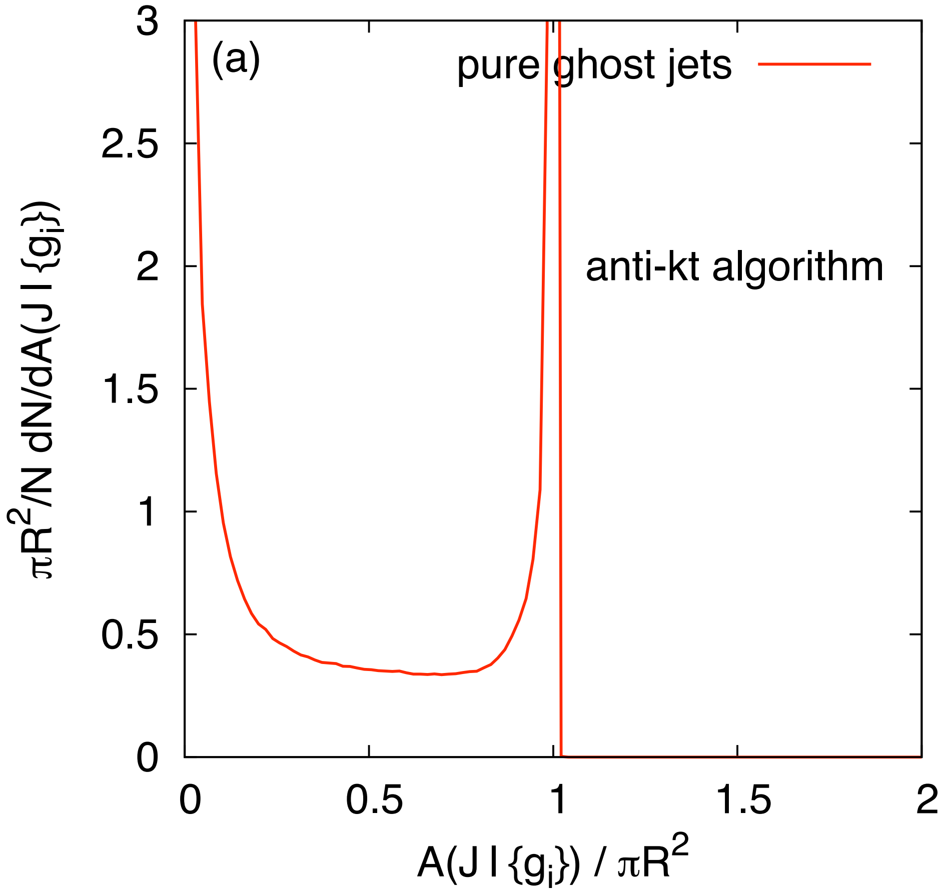
Only anti-k_t has the behaviour one would naively expect, i.e. area = πR^2

Active area distributions

k_t and Cam/AA



anti-k_t



For a roughly uniformly soft background, anti-k_t gives many small jets and many large ones

(you can't fill a plane with circles!)

Jet areas physical meaning

A jet's active area expresses the susceptibility of that jet's transverse momentum to contamination from a uniform background

Consider a jet of transverse momentum p_t, made up of N_g ghosts, each with transverse momentum $p_{t,g}$.

It holds
$$\frac{\partial p_t}{\partial p_{t,g}} = \frac{\partial (N_g p_{t,g})}{\partial p_{t,g}} = N_g$$

Recalling the definition of active jet area, $A_J = A_g N_g$, we can then rewrite

$$A_J = A_g \frac{\partial p_t}{\partial p_{t,g}} = \frac{\partial p_t}{\partial \rho}$$

Transverse momentum density, $\rho = p_{t,g}/A_g$

The jet area is therefore the susceptibility of a jet's p_t to contamination, because for a generic background density ρ it will hold

$$\Delta p_t = \rho \frac{\partial p_t}{\partial \rho} = \rho A_J$$

Susceptibility

Jet area: summary

- Jets CAN have an area, but one must define it

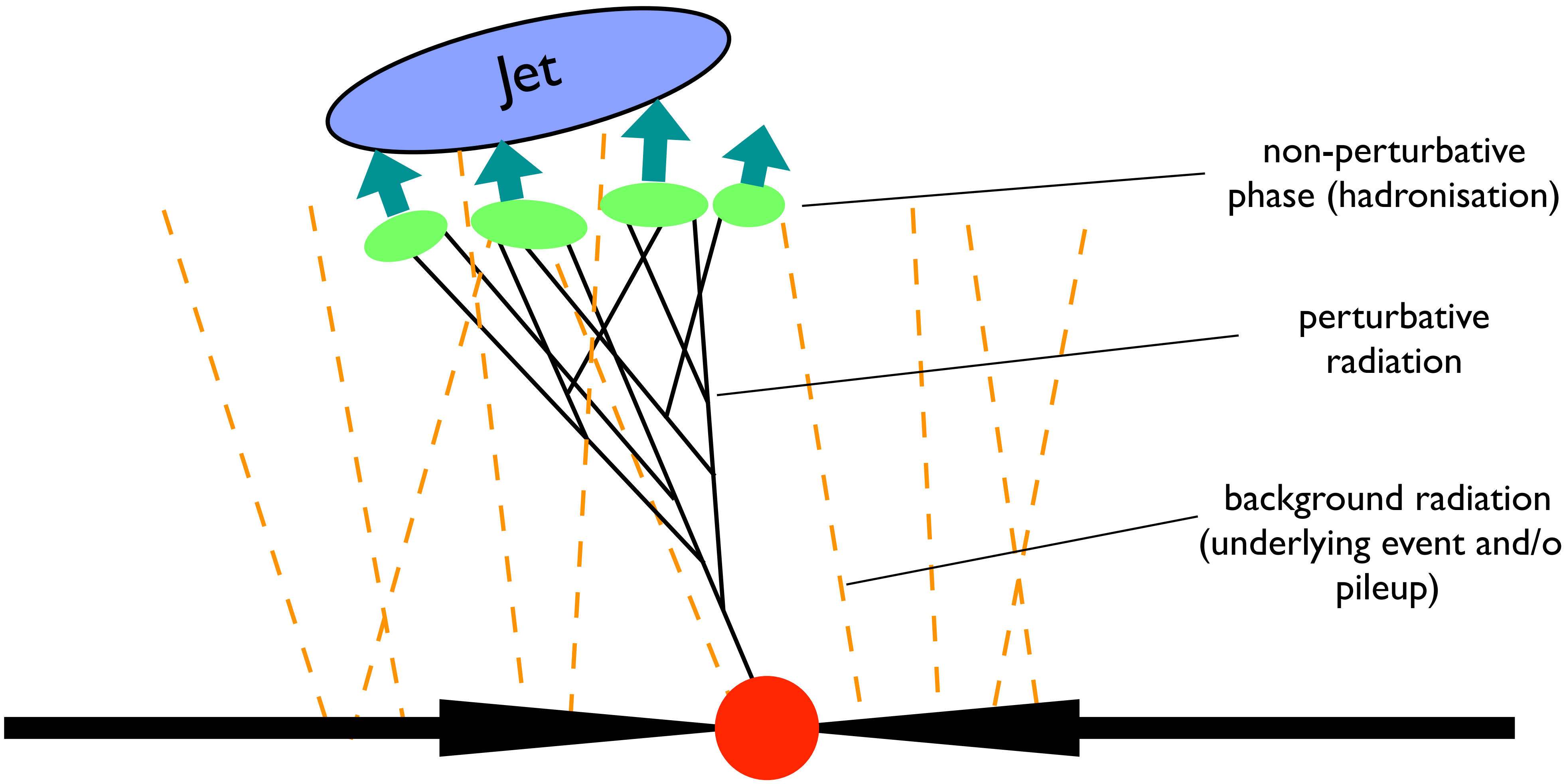
- The **jet (active) area expresses the susceptibility of a jet's transverse momentum to contamination from a uniform background**
  - Given a background transverse momentum density ρ, the jet's p_t will be modified by a quantity $A\rho$

- Different jet algorithms can have very different area properties:

  - Jet areas in many algorithms can fluctuate significantly from a jet to another. *Isolated hard jets in anti-k_t are an exception*

  - Jet areas can depend on a jet's p_t, driven by a (calculable) anomalous dimension that is specific to each jet algorithm. *Anti-k_t jets are again an exception: the anomalous dimension is zero*

What contributes to a jet's transverse momentum?



Effect of background

How are the hard jets modified by the background?

(Can be underlying event and/or pileup)

Susceptibility (how much bkgd gets picked up)	**Jet areas**
Resiliency (how much the original jet changes)	**Backreaction**

Hard jets and background

Modifications of the hard jet

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

Background momentum density (per unit area)

background

'susceptibility'

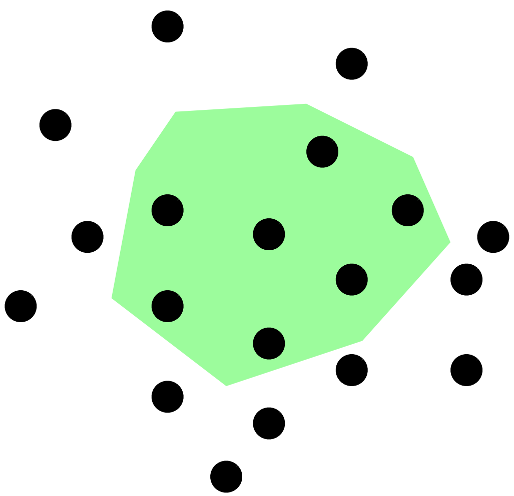
back-reaction

'resiliency'

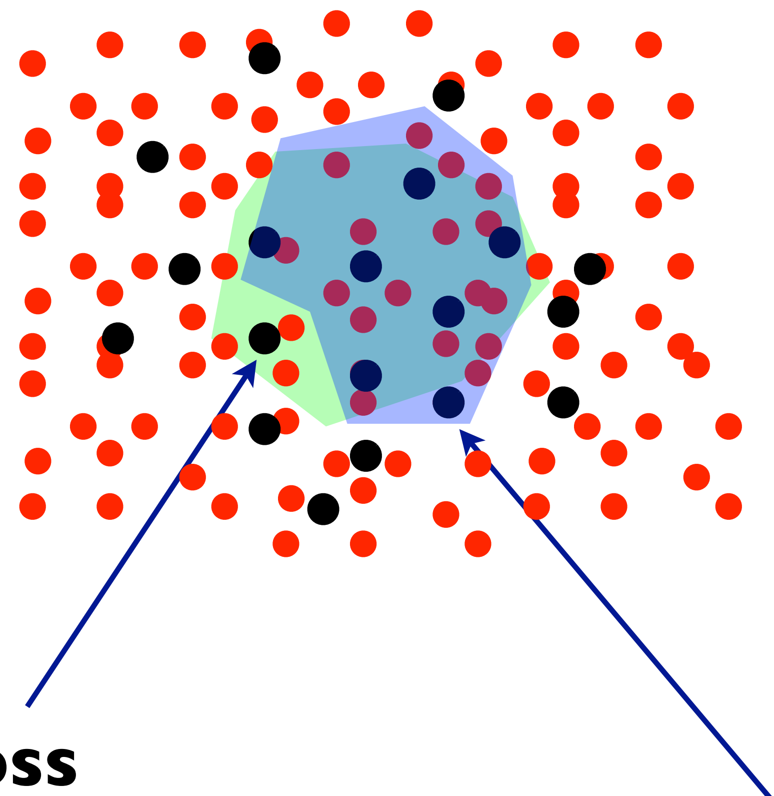
Resiliency: backreaction

"How (much) a jet changes when immersed in a background"

Without background



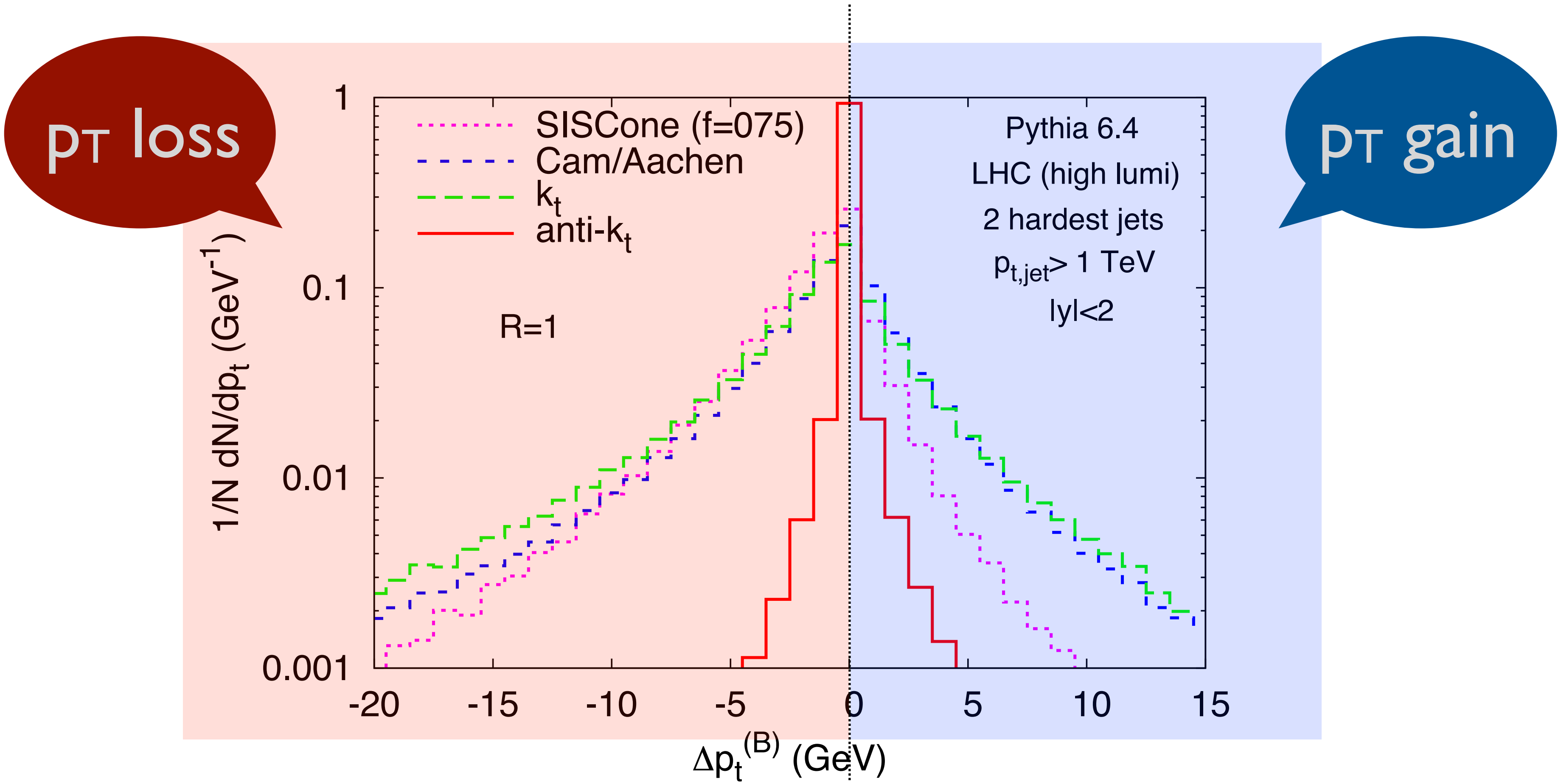
With background



Backreaction **loss**

Backreaction **gain**

Resiliency: backreaction



Anti-k_t jets are much more resilient to changes from background immersion

(NB. Backreaction is a minimal issue in pp background and at large pt.
Can be much more important in Heavy Ion collisions)

Background subtraction

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

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

MC, Salam, 0707.1378

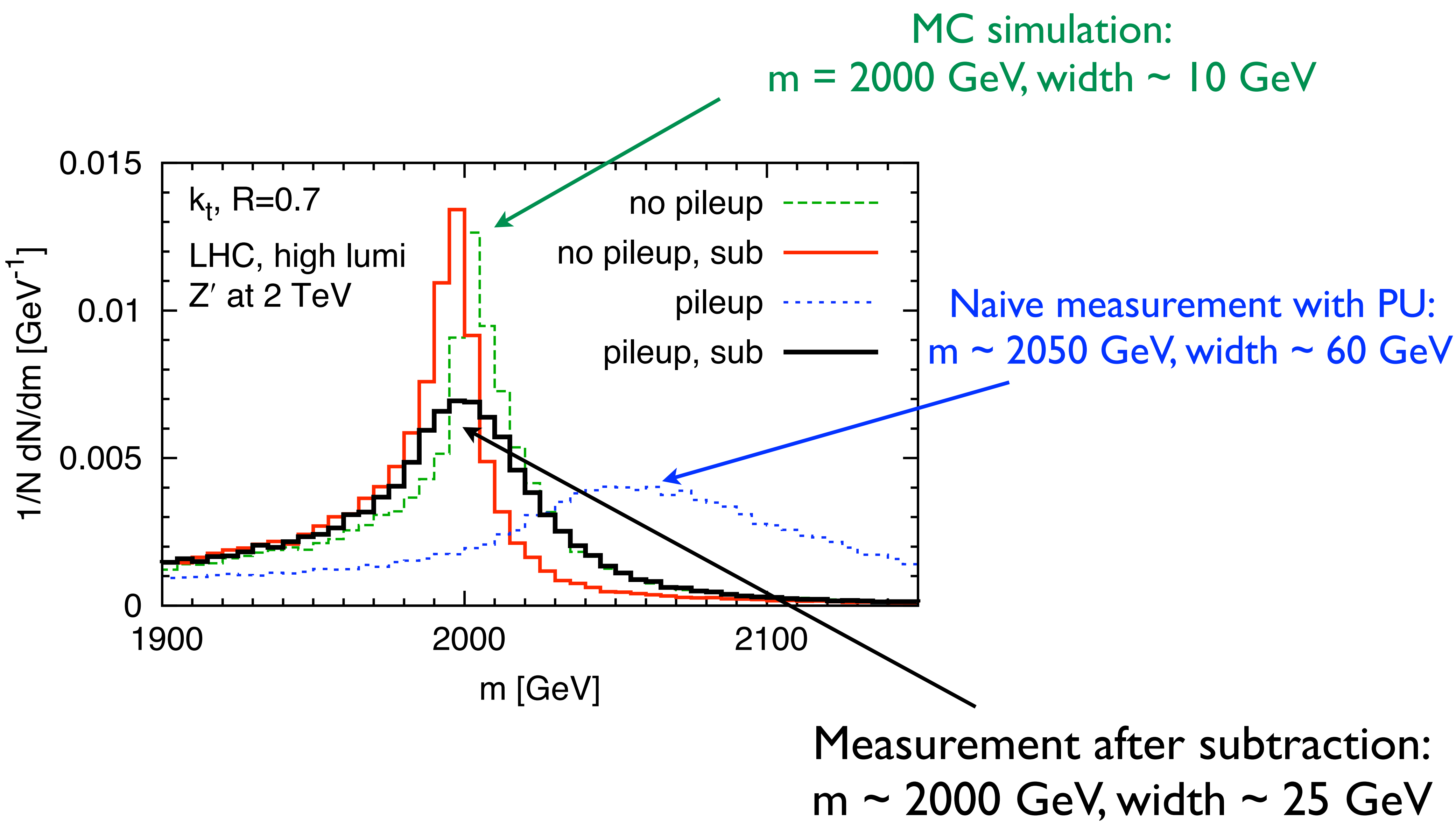
If ρ is measured on an event-by-event basis, and each jet subtracted individually, this procedure will remove many fluctuations and generally improve the resolution of, say, a mass peak

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

Irreducible fluctuations:
uncertainty of the subtraction

Example of pileup subtraction

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



The IRC safe algorithms

	Speed	Regularity	UE contamination	Backreaction	Hierarchical substructure
k_t	☺ ☺ ☺	☂	☂☂	☁☁	☺ ☺
Cambridge /Aachen	☺ ☺ ☺	☂	☂	☁☁	☺ ☺ ☺
anti-k_t	☺ ☺ ☺	☺ ☺	☁ → ☺ ☺	☺ ☺	✗
SISCone	☺	☁	☺ ☺	☁	✗

Backup

Areas as a dynamical jet property

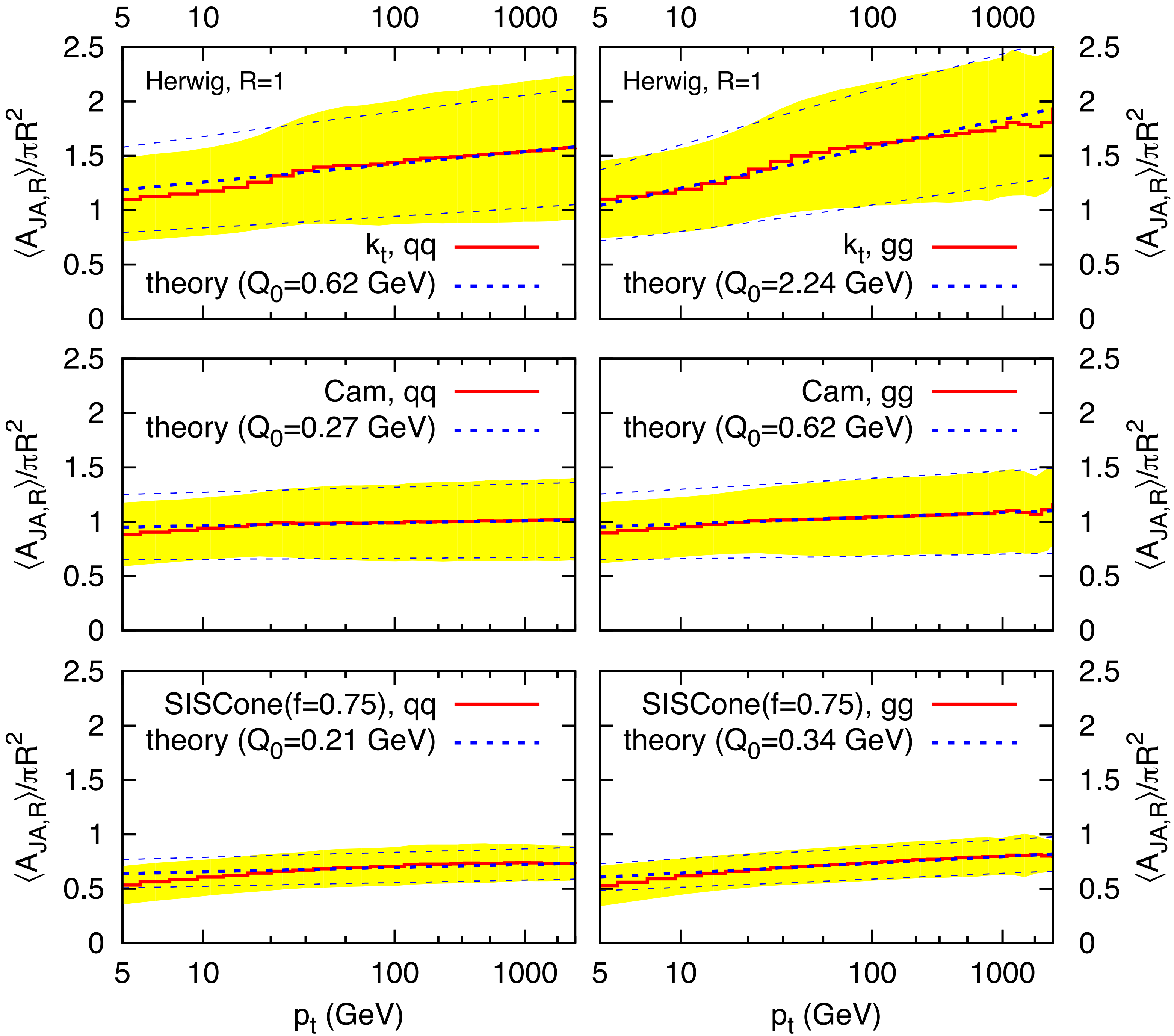
The average area of a jet can change with its p_t:

$$\langle \Delta A \rangle = \mathrm{D}\, \frac{C_1}{\pi b_0} \ln \frac{\alpha_s(Q_0)}{\alpha_s(R p_{t1})}$$

	k_t	Cam/Aa	SISCone	anti-k_t
D	0.52	0.08	0.12	0

Again, only anti-k_t has a typical area that does **not** increase with p_t

Jet areas scaling violations



Averages and dispersions evolution from Monte Carlo simulations (dijet events at LHC) in good agreement with simple LL calculations

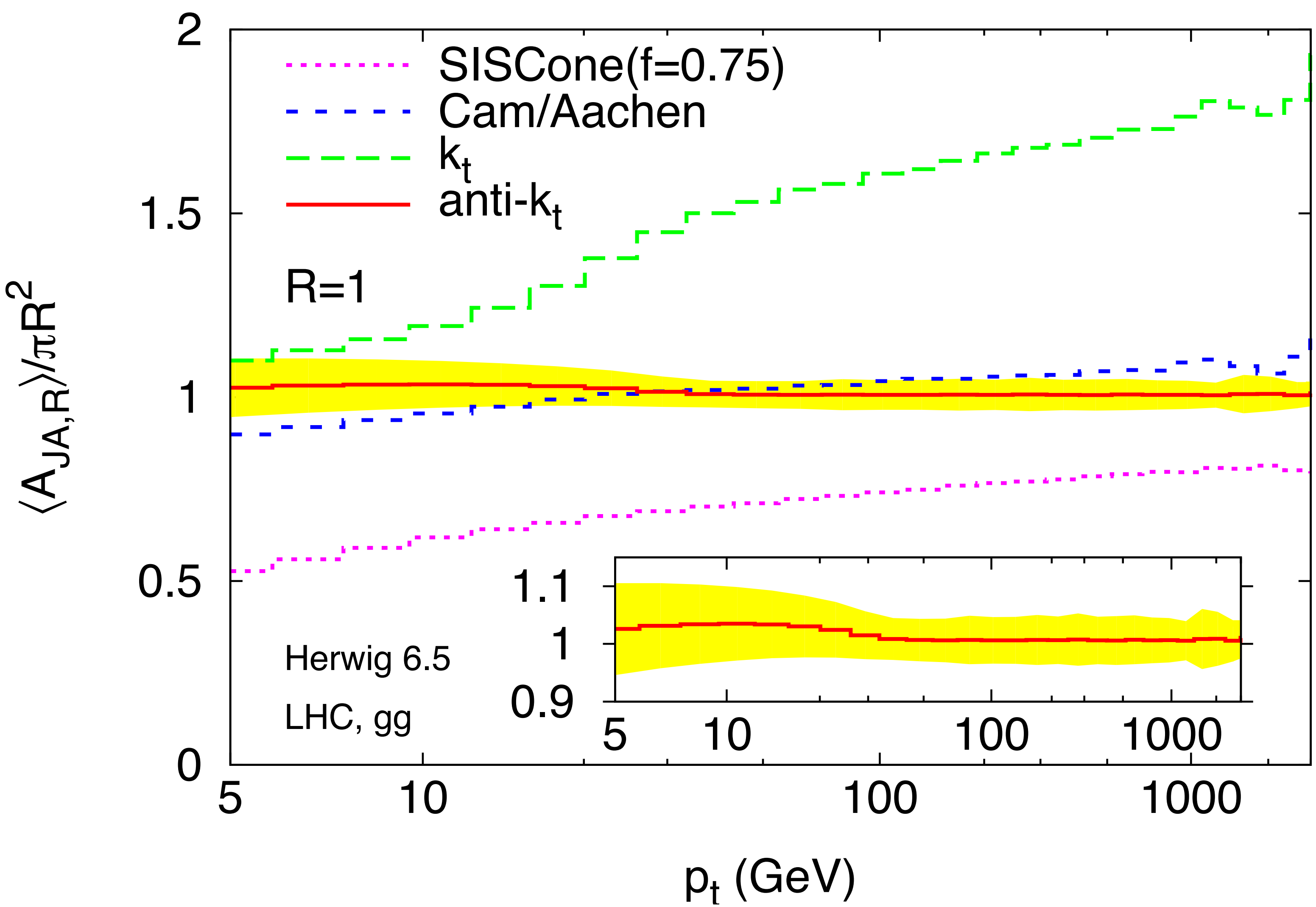
Area scaling violations are a legitimate observable.

(Though they might not be the best place where to measure α_s ....)

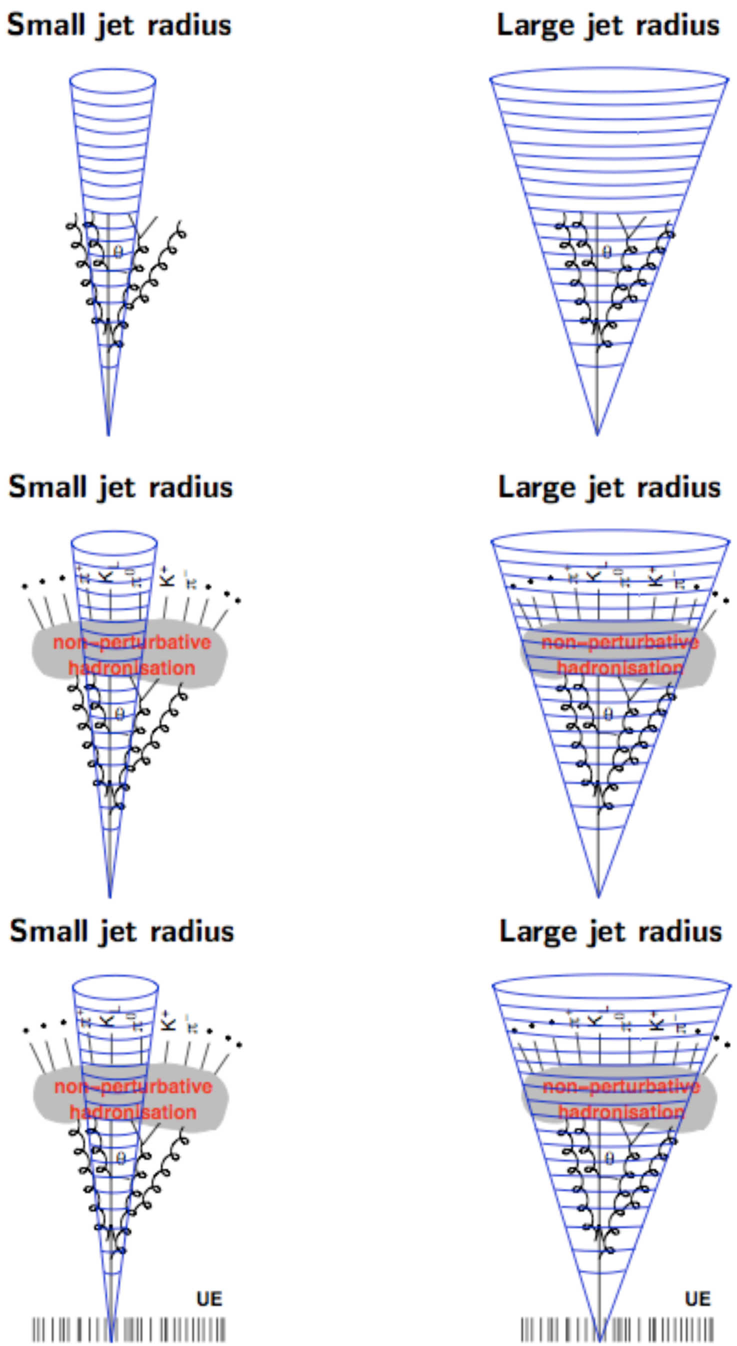
Jet areas scaling violations

MC, Salam, Soyez, arXiv:0802.1189

Check anti-k_t behaviour: scaling violations indeed absent, as predicted



Effects of jet 'radius'



perturbative radiation:
large radius better (lose less)

non-perturbative hadronisation:
large radius better (lose less)

underlying event:
large radius worse (capture more)

R-dependent effects

Perturbative radiation: $$\Delta p_t \simeq \frac{\alpha_s(C_F, C_A)}{\pi} p_t \ln R$$

Hadronisation: $$\Delta p_t \simeq -\frac{(C_F, C_A)}{R} \times 0.4 \text{ GeV}$$

Underlying Event: $$\Delta p_t \simeq \frac{R^2}{2} \times (\underset{\text{Tevatron}}{2.5} — \underset{\text{LHC}}{15} \text{ GeV})$$

(small-R limit results)

Analytical estimates: Dasgupta, Magnea, Salam, arXiv:0712.3014