

eic-smear

Workshop on EIC Detector R&D Simulations

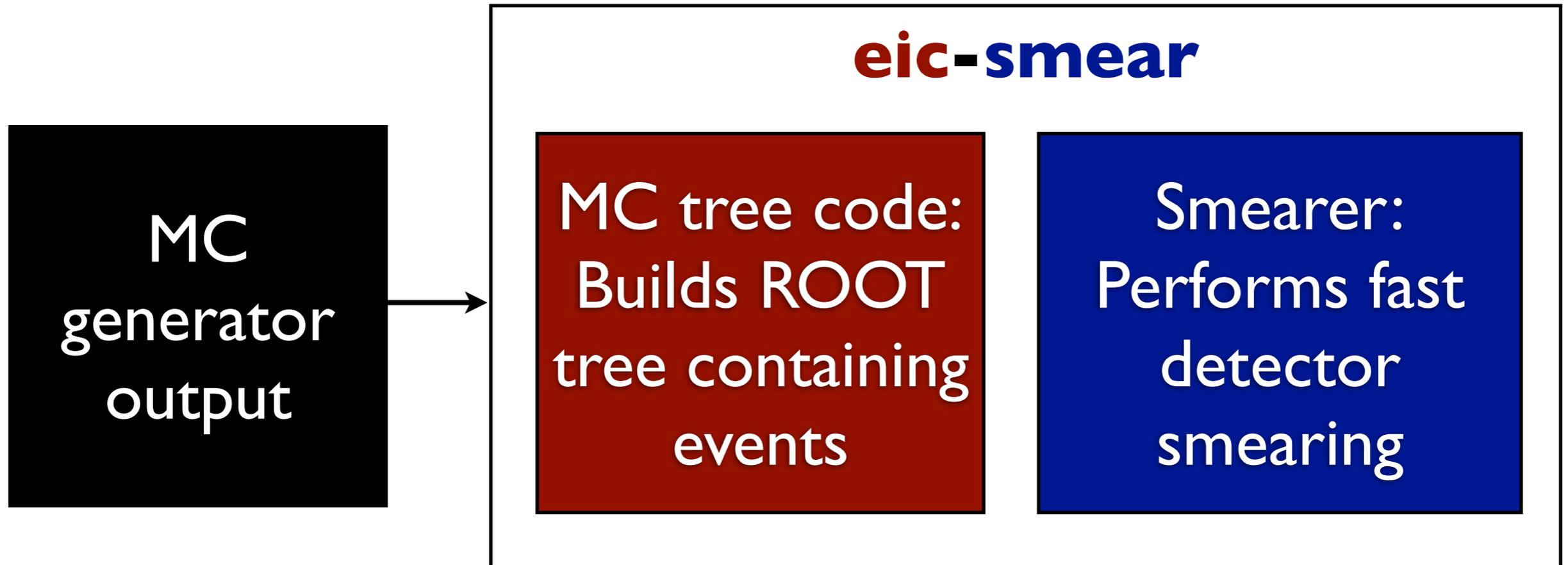
Thomas Burton

8th October 2012

Overview

- What **is** it?
- ... and what is it **not**?
- How to **use** it
- Example **results**

Overview



- **C++** code running in **ROOT**
- Builds with **configure/Make**
- Single **libeicsmear.so** to load in ROOT

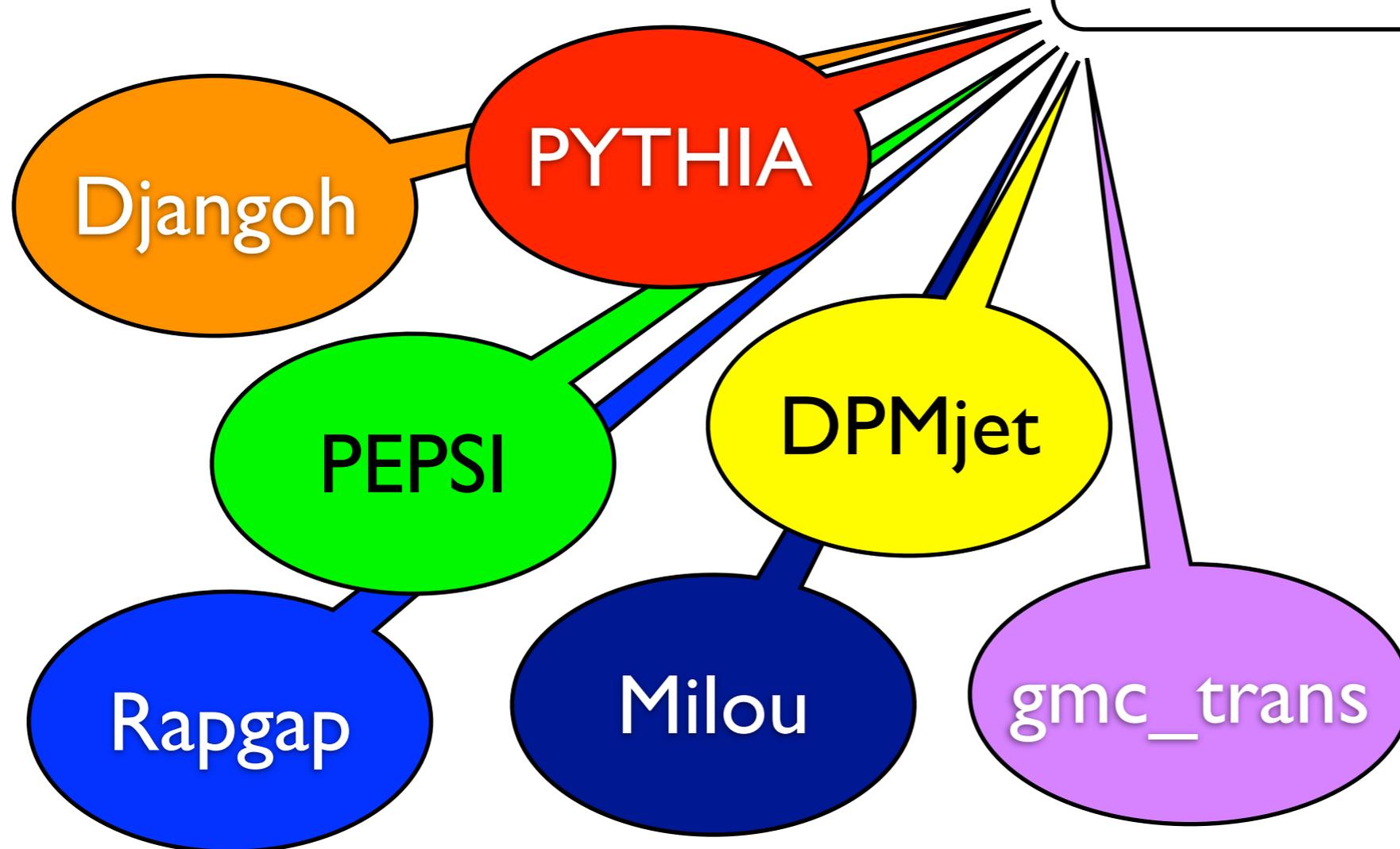
MC tree code

MC tree code

Base DIS event

$x, Q^2, y, W^2, v,$
track list

defines
common
format



Event class
for each
generator
adds specific
data

standard format: macros can analyse
different generators **without change**

Running

ASCII file
in standard format:
header + tracks*

Event class
defines how to
process header

Simple "build" routine:

```
BuildTree(  
  "file.txt",  
  "file.root");
```

Processes event
header/tracks
+
Does file I/O

ROOT file

**This is all
the end-
user has to
run**

*For example see

https://wiki.bnl.gov/eic/index.php/PYTHIA#Output_file_structure

Smearing

What is(n't) it?

- Utility for smearing of MC output
- It's **NOT** a replacement for Geant!
- But, if you are asking...

“Given a (known) detector performance, how well can I measure some physics observable(s)?”

or

“If I need to measure X with to some precision, what detector performance do I need?”

... then maybe it is for you

Architecture

- Written by Michael Savastio (student)
- Fast - thousands of events/second
- Smears
 - ▶ tracks: p , E , angle, ID
 - ▶ DIS kinematics: x , Q^2 , y
- **Not** specific to any generator
 - ▶ Same smearing specification works for **all** generator output

The idea

(single) quantity,
 X , to smear:
 E, p, θ, φ

+

Function defining
 $\sigma(X) =$
 $f([E, p, \theta, \varphi])$

+

Acceptance
for X in
 $E, p, \theta, \varphi, p_T, p_Z$

||

NOT a
“physical
detector”:
represents the
**overall
performance**
in measuring a
quantity.

“Device”

“Device”

“Device”

“Device”

“Device”

“Detector”

How to use it

- Write a ROOT script:

```
Smear::Detector createDetector() {  
    // Resolution in momentum, sigma(P).  
    // sigma(P) = 0.4%P + 0.3%P^2.  
    Smear::Device tracking("P", "0.004 * P + 0.003 * pow(P, 2)");  
    // Resolution in energy, sigma(E) = 14% * sqrt(E)  
    // 3rd argument == 1 -> smear only photons & electrons.  
    Smear::Device emcal("E", "0.14 * sqrt(E)", 1);  
    // Add devices to a Detector.  
    Smear::Detector detector;  
    detector.AddDevice(tracking);  
    detector.AddDevice(emcal);  
    return detector;  
}
```

Simple devices
define $\sigma(X)$ via
text string

- Smear your ROOT tree:

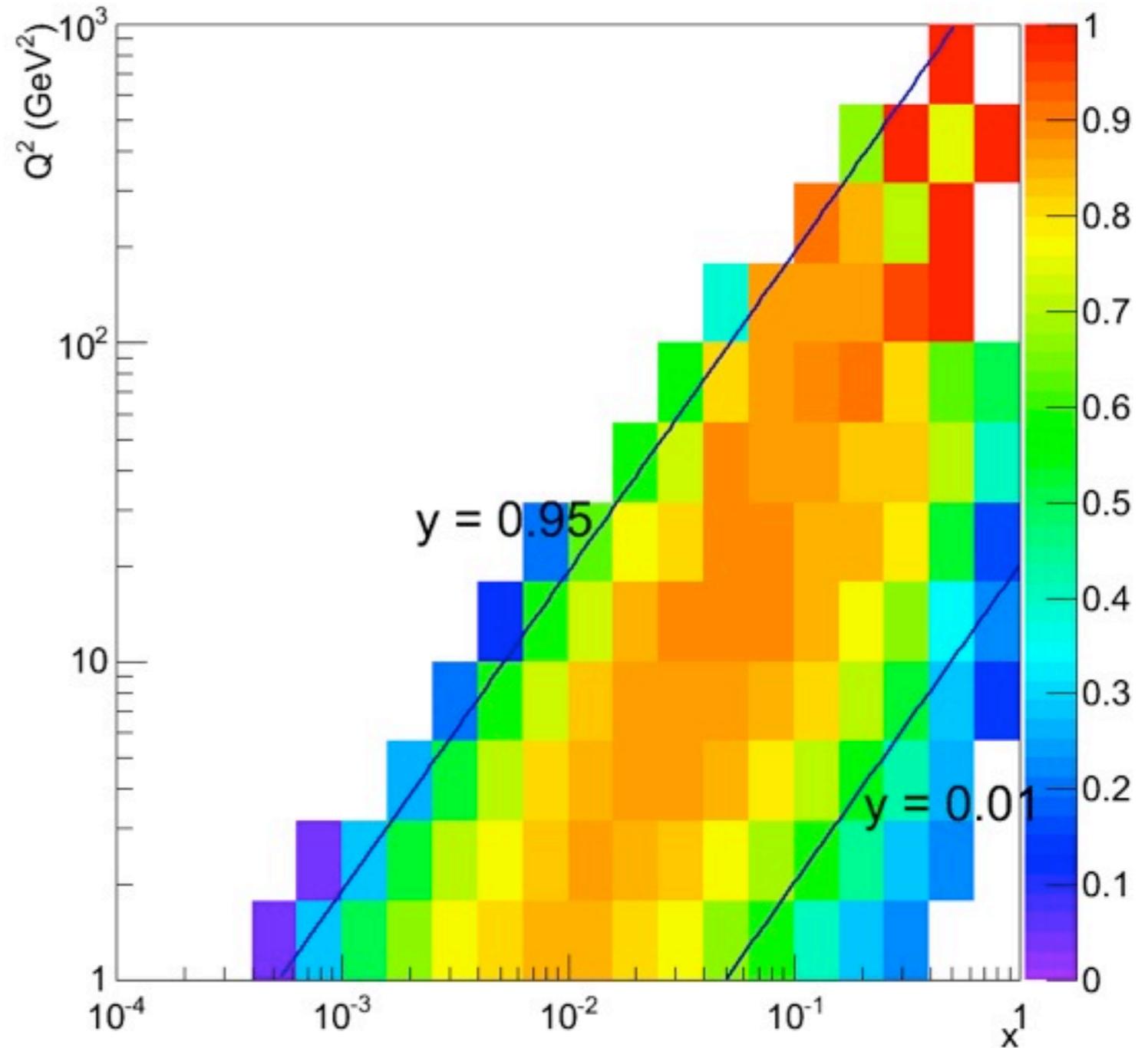
```
root[0] SmearTree(createDetector(), "mc.root", "smear.root");
```

Handles event
loop, file I/O

Example: Q^2 - x bin migration

- Q^2 - x bin migration due to p resolution using hadron calculation

η	$\sigma(P)$
$ \eta < 1$	0.2% P
$1 < \eta < 3$	1% P
$3 < \eta < 5$	3% P



Other “Devices”

- Extensible to more specialised devices
 - ▶ “Bremmstrahlung” class
 - ▶ “Tracker” class, implementing

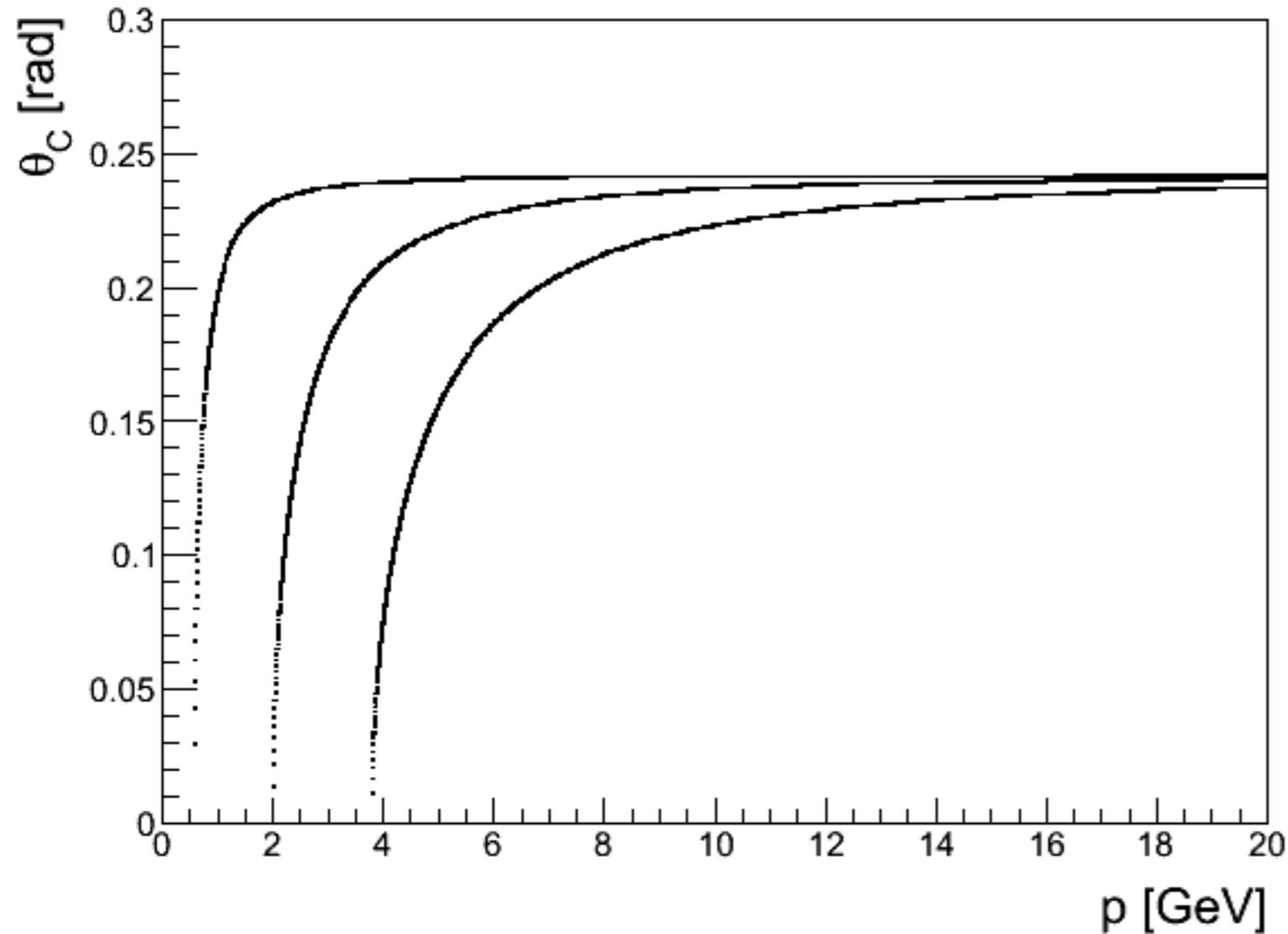
intrinsic resolution

$$\left| \frac{dp}{p} \right|_i = \frac{p}{0.3B_T} \frac{\sigma_{r\phi}}{(L')^2} \sqrt{\frac{720}{n+4}}$$

multiple scattering

$$\left| \frac{dp}{p} \right|_{MS} = \frac{1}{0.3B_T} \frac{0.0136}{L\beta \cos^2(\gamma)} \sqrt{n_{rl}}$$

Example: $\sigma(p)$ and PID

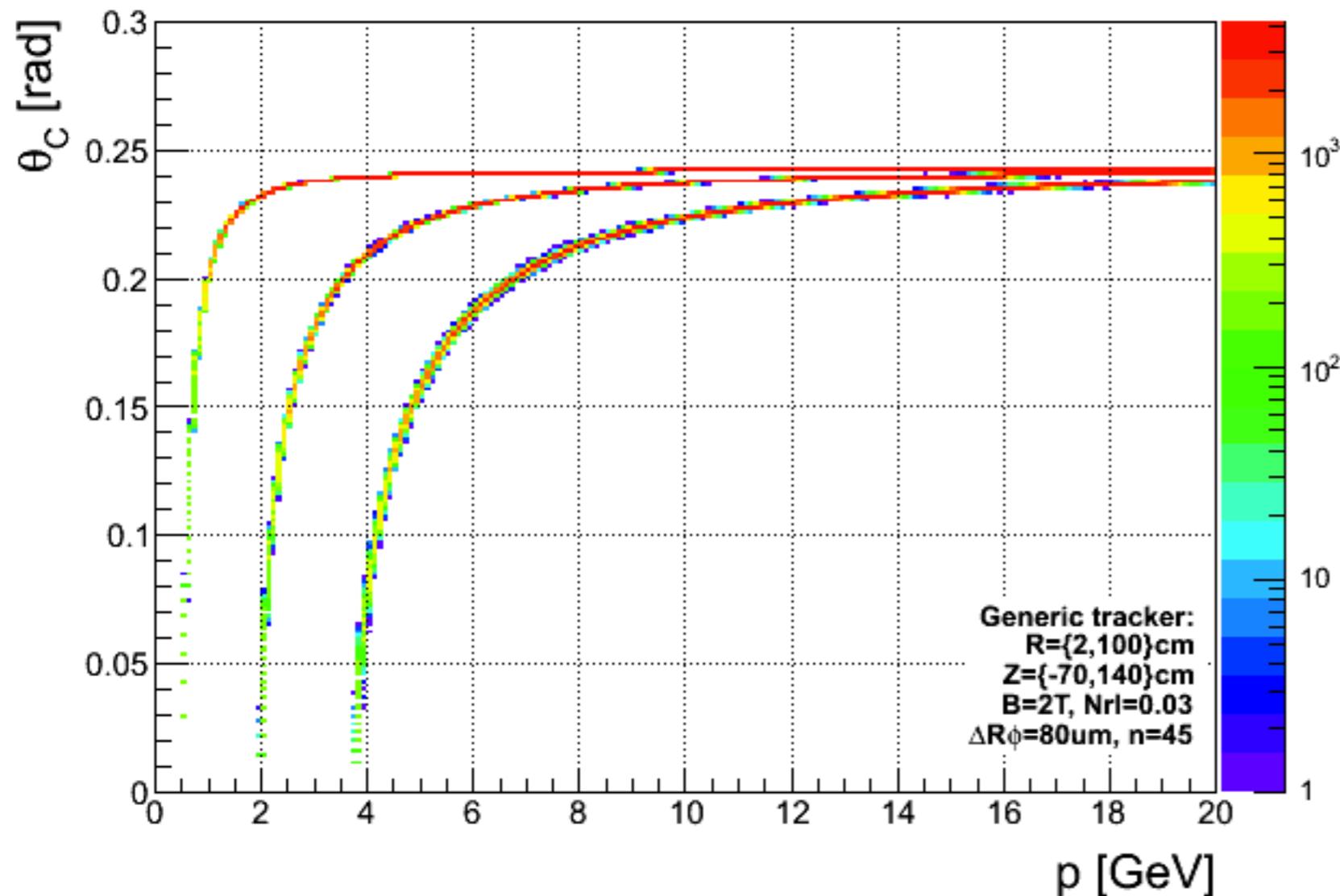


Aerogel $n = 1.03$

Will Foreman

Example: $\sigma(p)$ and PID

Cherenkov angle vs. momentum ($-1 < \eta < 1$)



Aerogel $n = 1.03$

R = [2, 100] cm

Z = [-70, 140] cm

B = 2T

0.03 rad lengths

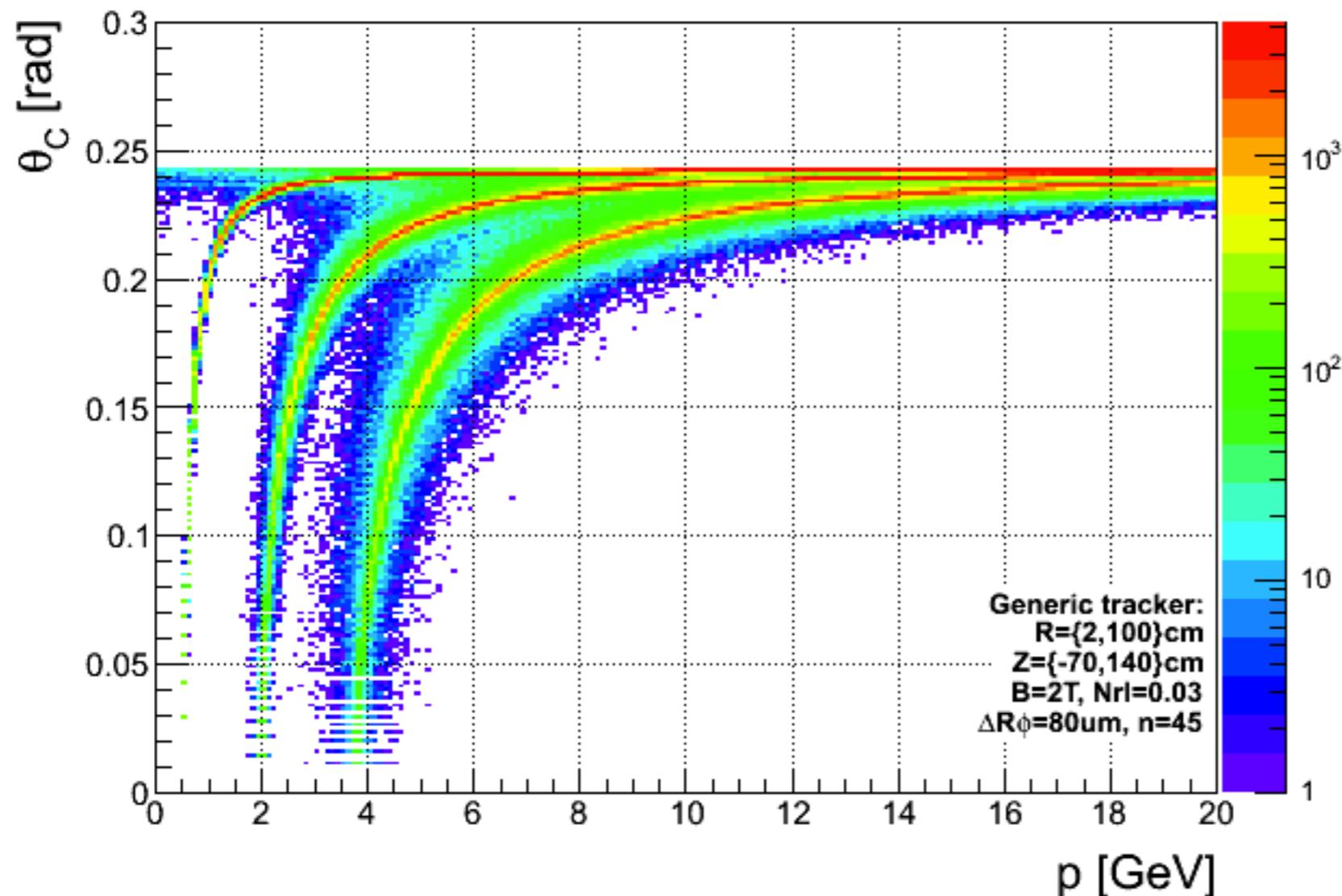
$\Delta R\phi = 80\mu\text{m}$

Max. 45 points

Will Foreman

Example: $\sigma(p)$ and PID

Cherenkov angle vs. momentum ($1 < \eta < 3$)



Aerogel $n = 1.03$

$R = [2, 100]$ cm

$Z = [-70, 140]$ cm

$B = 2\text{T}$

0.03 rad lengths

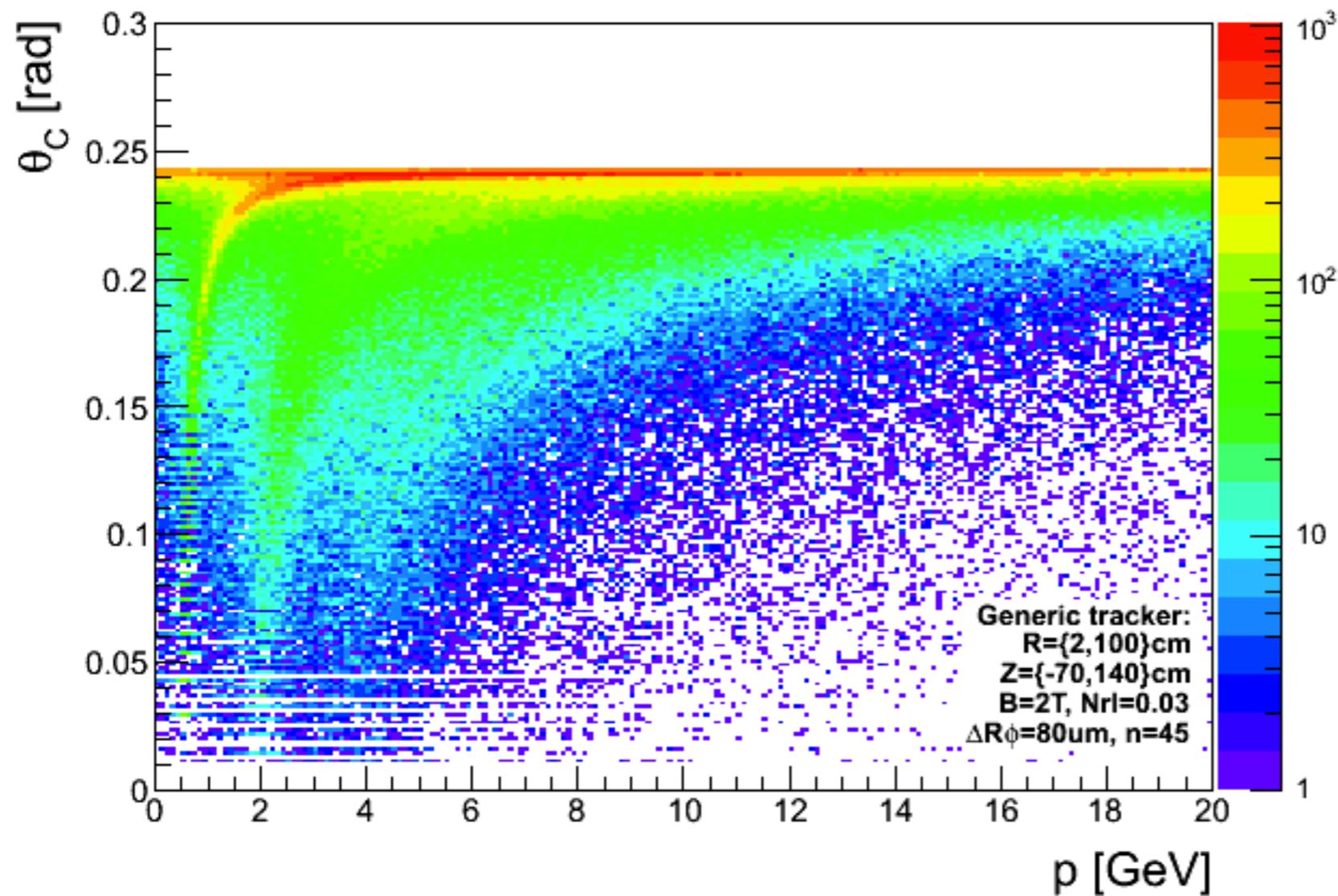
$\Delta R\phi = 80\mu\text{m}$

Max. 45 points

Will Foreman

Example: $\sigma(p)$ and PID

Cherenkov angle vs. momentum ($3 < \eta < 5$)



Aerogel $n = 1.03$

R = [2, 100] cm

Z = [-70, 140] cm

B = 2T

0.03 rad lengths

$\Delta R\phi = 80\mu\text{m}$

Max. 45 points

Will Foreman

Access and documentation

Read about it:

<https://wiki.bnl.gov/eic/index.php/Eic-smear>

Get it:

svn checkout <http://svn.racf.bnl.gov/svn/eic/Utilities/eic-smear/trunk> eic-smear

Build it (see the README):

```
autoconf --force --install  
configure  
make
```

Run it:

```
root[0] gSystem->Load("/path/to/libeicsmear");
```