Argonne NATIONAL LABORATORY

"GRETINA/AUX Gamma Ray Tracking software"

from a
practical data analysis
point of view

MSU 10/21/2012

torben lauritsen, ANL

The way I use ROOT...

- Mostly just use ROOT to store and display spectra... sad but true
- Have some utilties that are shared with GSSort: GSUtil_cc.so [simple display, make 2d windows, energy calibrate, time align (for GS), read root file, write to Radford matrix...]
- Just know enough to be dangerous, I'm no expert...
- Root spectra very portable!
- Quite often export to gf3 spectra

My ‘steps’ for tracking data

- [1] offline: Make coincidences (using the time stamps in the GT and AUX data)
- [2] Track the GT data (‘MY MAIN TASK’); based on I-Y code (Fortran ~1.25)
- [3] Analyze the data aided by ROOT (this is where the Doppler correction is applied and cuts are made on the FOM values)

Dual use tracking code

- Can be used offline
- Build in such a way that the code can also be used in the compute cluster producing the mode-1 data (what we originally envisioned we would produce)
- ---------
- The root sorter, ctkana, is '*incidental*', but could be used as a starting point for offline data processing (it is relatively simple, no trees...)

Two access avenues

‘Contributers’: svn.anl.gov

‘Users’: http://www.phy.anl.gov/gretina

The ANL web site also has the “GT_aux.pdf” documents that describes how we (tl, I-Y & Dirk) think we should handle GT+AUX data in general. Also has an intro to tracking by I-Y

Kinda following that document...

Reminder of GT data format:

```
struct gebData {
  int type;
  int length;
  long long timestamp;
};
```

```
#define GEB_TYPE_DECOMP       1
#define GEB_TYPE_RAW          2
#define GEB_TYPE_TRACK        3
#define GEB_TYPE_BGS          4
#define GEB_TYPE_S800_RAW     5
#define GEB_TYPE_NSCLnonevent 6
#define GEB_TYPE_GT_SCALER    7
#define GEB_TYPE_GT_MOD29     8
#define GEB_TYPE_S800PHYSDATA 9
```

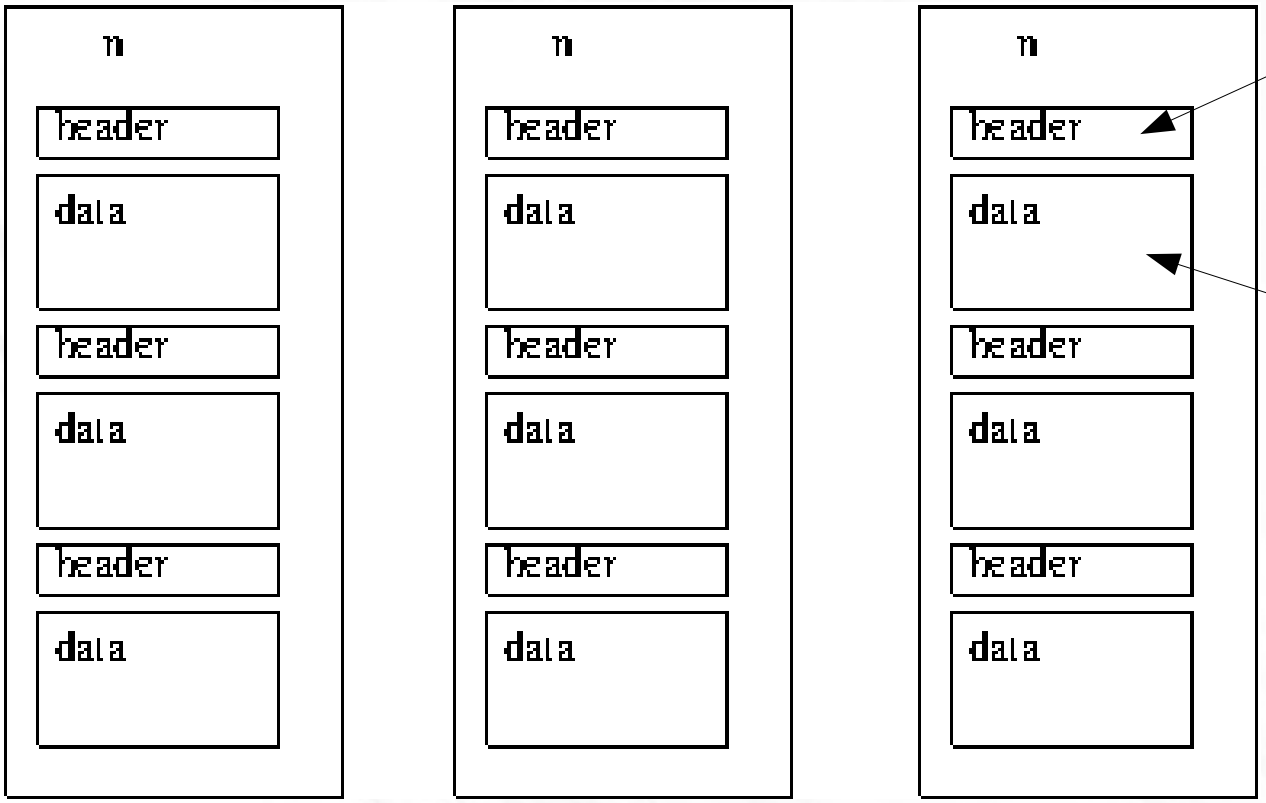
payload

Can inspect timestamp without knowing what the payload data is

Step1:

- Mode 1 data from gretina already has the events time correlated, but offline data is just a long stream of GT and AUX data
- ‘combdata3’ will make coincidences based on the timestamps in the headers of the gebData headers. It does not care about the data payloads.
- Makes coincidence events and filter a little on what it writes out
- Can energy calibrate too if needed

Data for the tracking code:



More precise time gate applied later in ROOT sorter

The tracking task

- Program: trackMain, --- ***more details later***!
- *Result: Adds an array of tracked data information (== gamma rays) to the data stream:*

```
typedef struct CLUSTER_INTPTS {
  int   valid; /* alway 1 in output, may be zero internally */
  int   ndet; /* # interaction points */
  int   tracked; /* ==1 if we managed to track */
  float fom;  /* fom value for the tracking */
  float esum;   /* gamma ray energy */
  int   trackno;
  int   bestPermutation;
  int   processsed; /* not used now */
  CL_INTPTS intpts[MAX_NDET];
} CLUSTER_INTPTS;
```

One gamma ray

The interaction points for the gamma ray:

```
typedef struct CL_INTPTS {
    float xx, yy, zz;
    float edet;
    int   order;   /* 0 == first interaction point */
    long long int timestamp;
    int   shellHitPos; /* internal use only */
    int   detno; /* not sure it is used or meaningful */
    } CL_INTPTS;
```

The one with order==0 is the first interaction point according to the trackMain program

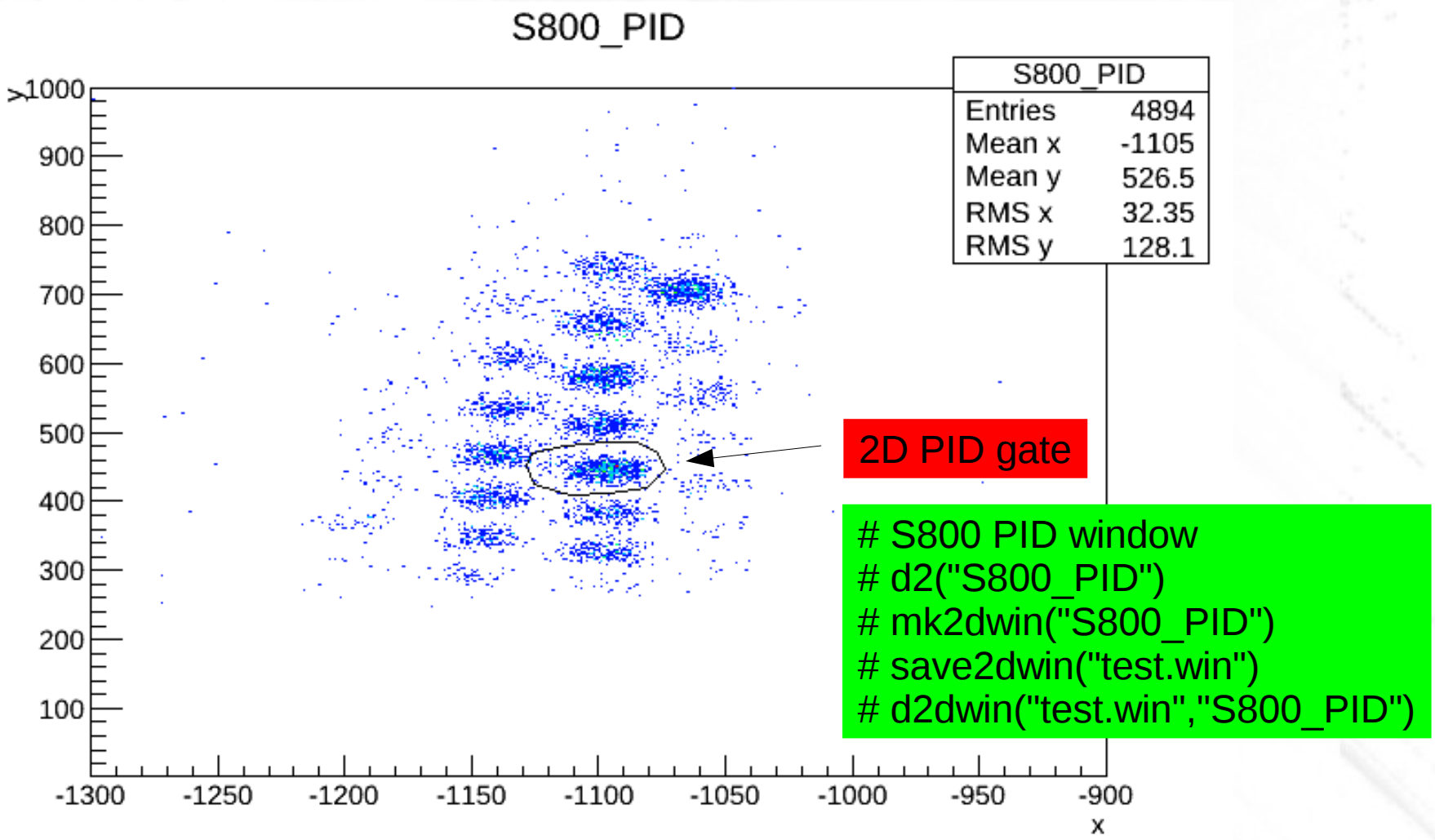
- The trackMain program only seeks out the GT data (GEB_TYPE_DECOMP [==1]), all other types of data is just pass on
- The tracking information is simply ADDED to the data
- The Mode 2 data is still there
- You can re-track, you have lost nothing in the tracking task
- You can do simple on-line tracking and ‘better’ tracking off-line

ctkana, the ROOT sorter

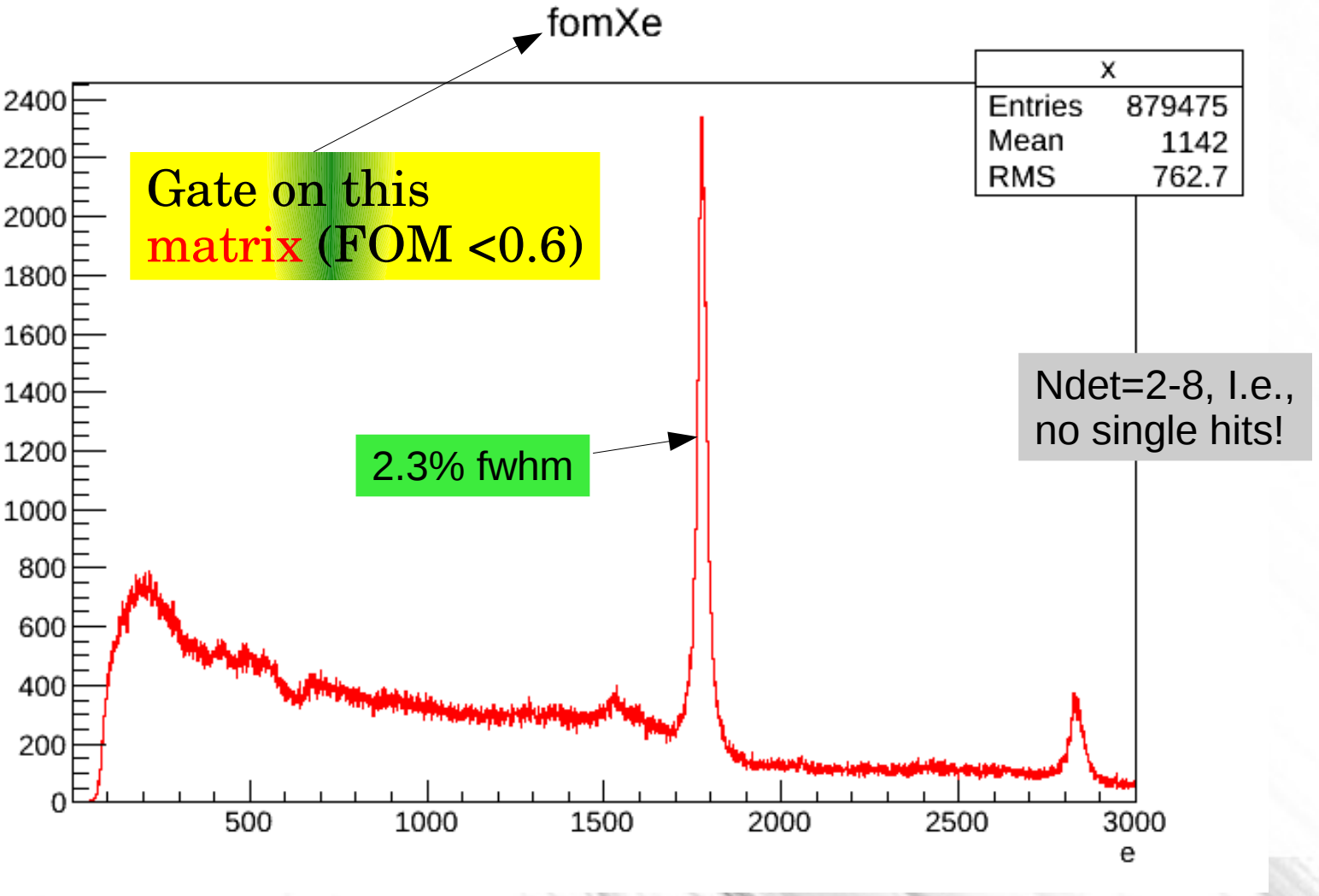
- Reads the tracked data format
- Has access to tracked data, mode 2 data and ext (non-GT) data
- Built the same way as GSSort, but is not nearly as refined
- Uses GSUtils programs
- No trees or anything fancy (yet?)

ctkana and BGS/S800

- For BGS data, extract Si energies (timestamp is already in GEB header)
- Construct TAC spectrum [GT-AUX] and sets narrow time gates
- For S800, applies 2D PID gate (if needed)
- For S800, applies Dirk's fancy Doppler correction by interpreting the GEB_TYPE_S800PHYSDATA [==9] data
- For S800: uses GEB_TYPE_MOD29 [==8] for the TAC spectrum


S800_PID
S800_PID
Entries 4894
Mean x -1105
Mean y 526.5
RMS x 32.35
RMS y 128.1
y
1000
900
800
700
600
500
400
300
200
100
-1300
-1250
-1200
-1150
-1100
-1050
-1000
-950
-900
x
2D PID gate
S800 PID window
d2("S800_PID")
mk2dwin("S800_PID")
save2dwin("test.win")
d2dwin("test.win","S800_PID")

S800 28Si spectrum



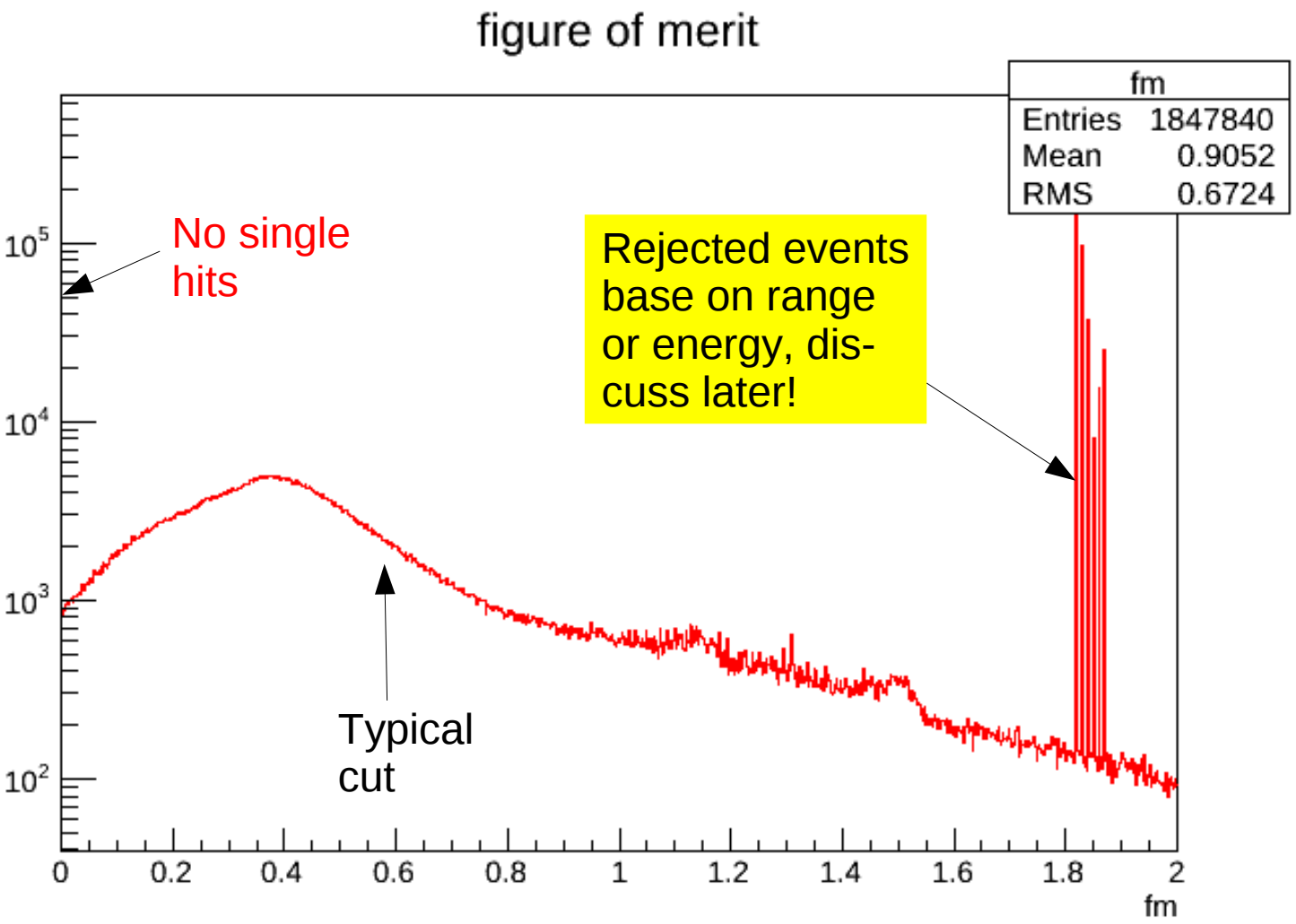
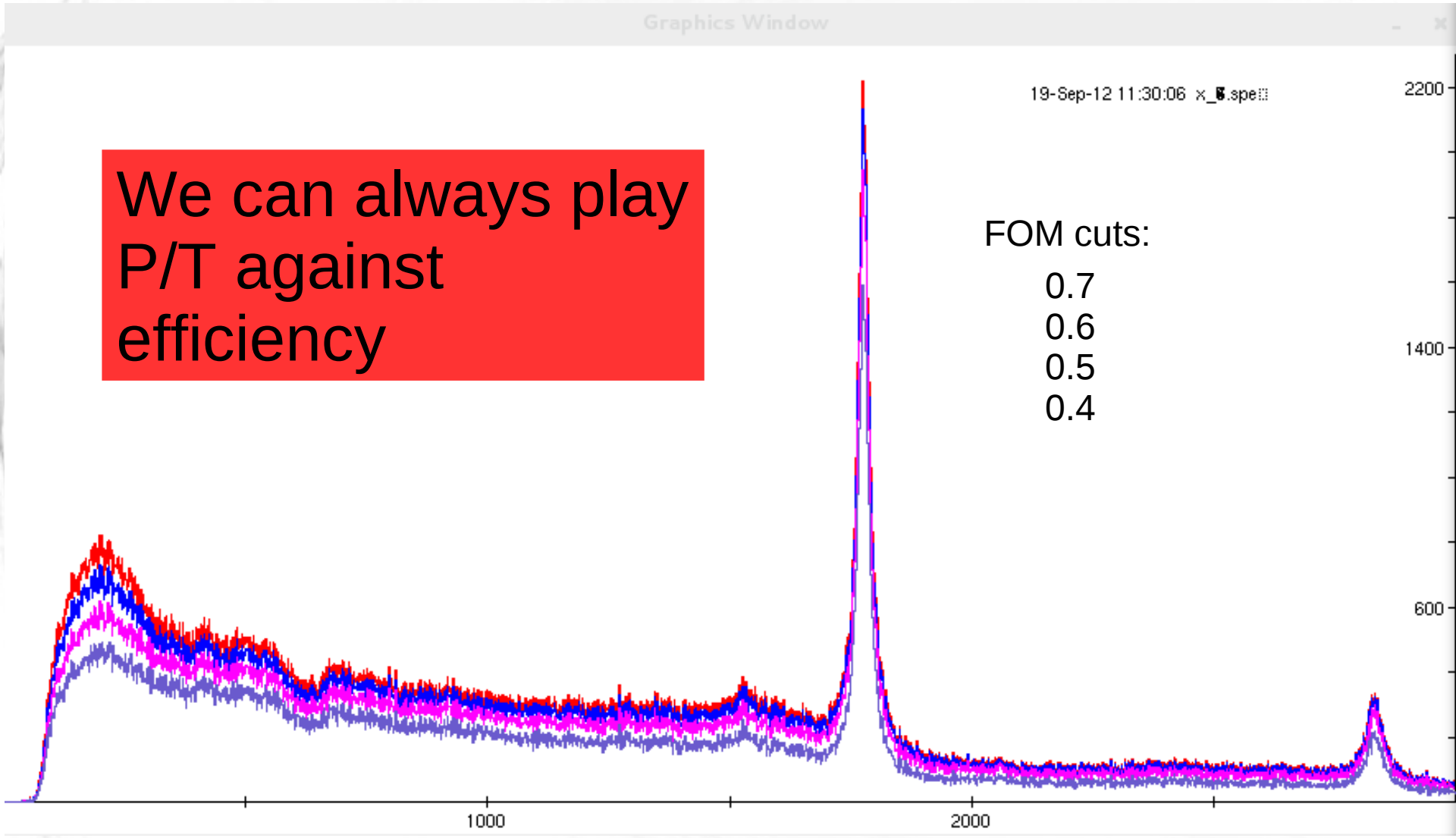
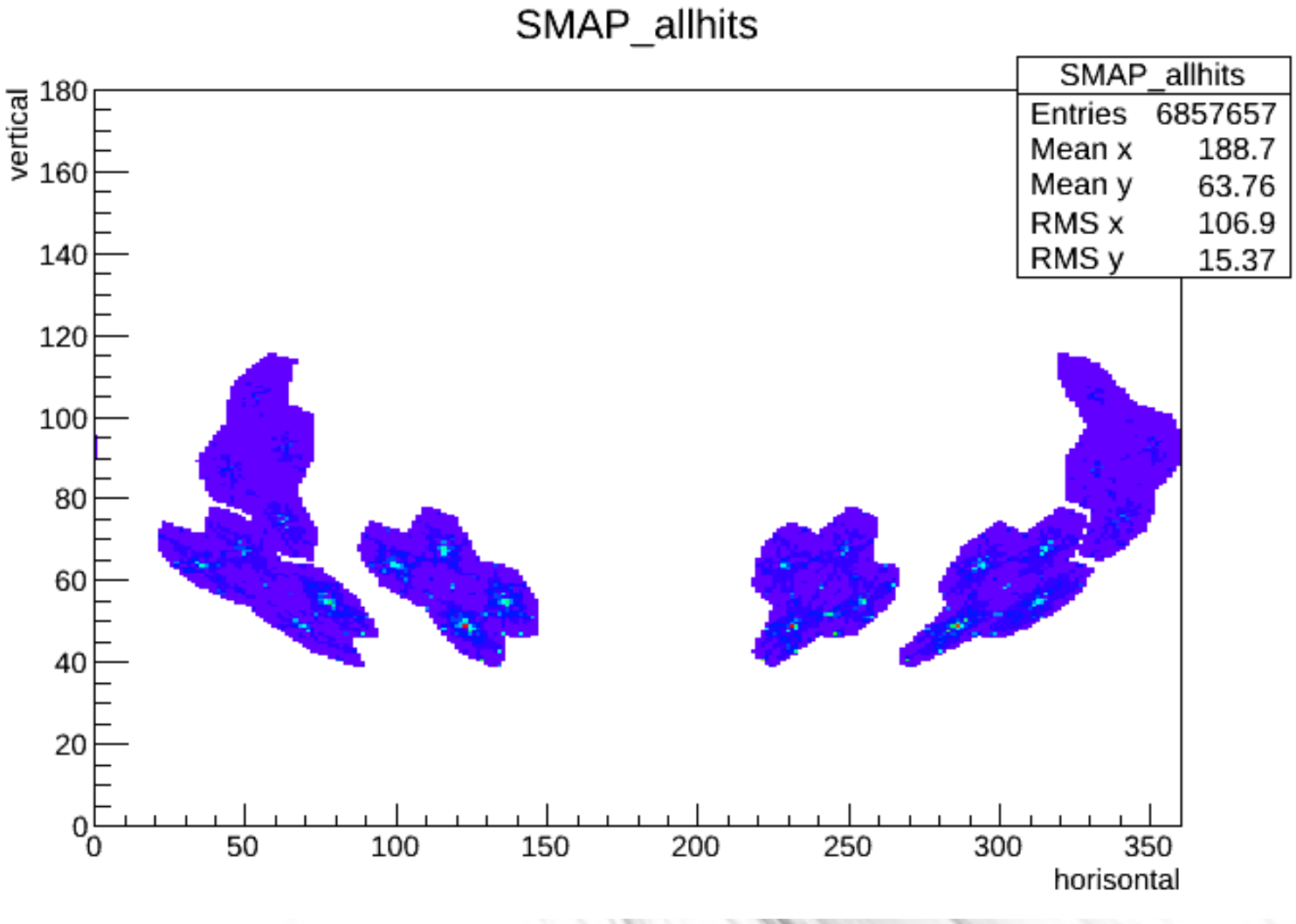


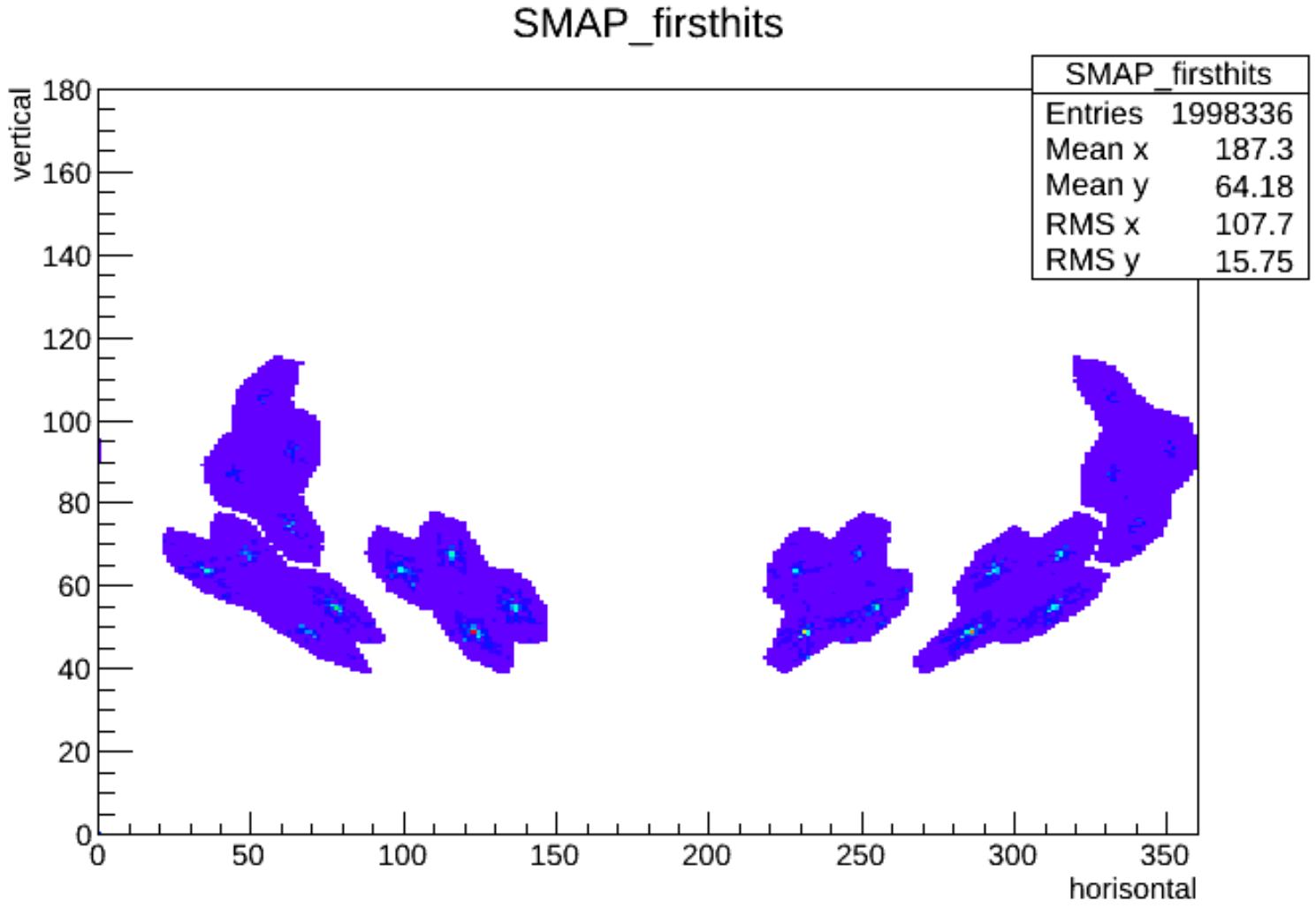
figure of merit
fm
Entries 1847840
Mean 0.9052
RMS 0.6724
No single hits
Rejected events base on range or energy, dis-cuss later!
Typical cut
10^5
10^4
10^3
10^2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
fm


Graphics Window
We can always play P/T against efficiency
19-Sep-12 11:30:06 x_B.spe
FOM cuts:
0.7
0.6
0.5
0.4
2200
1400
600
1000
2000

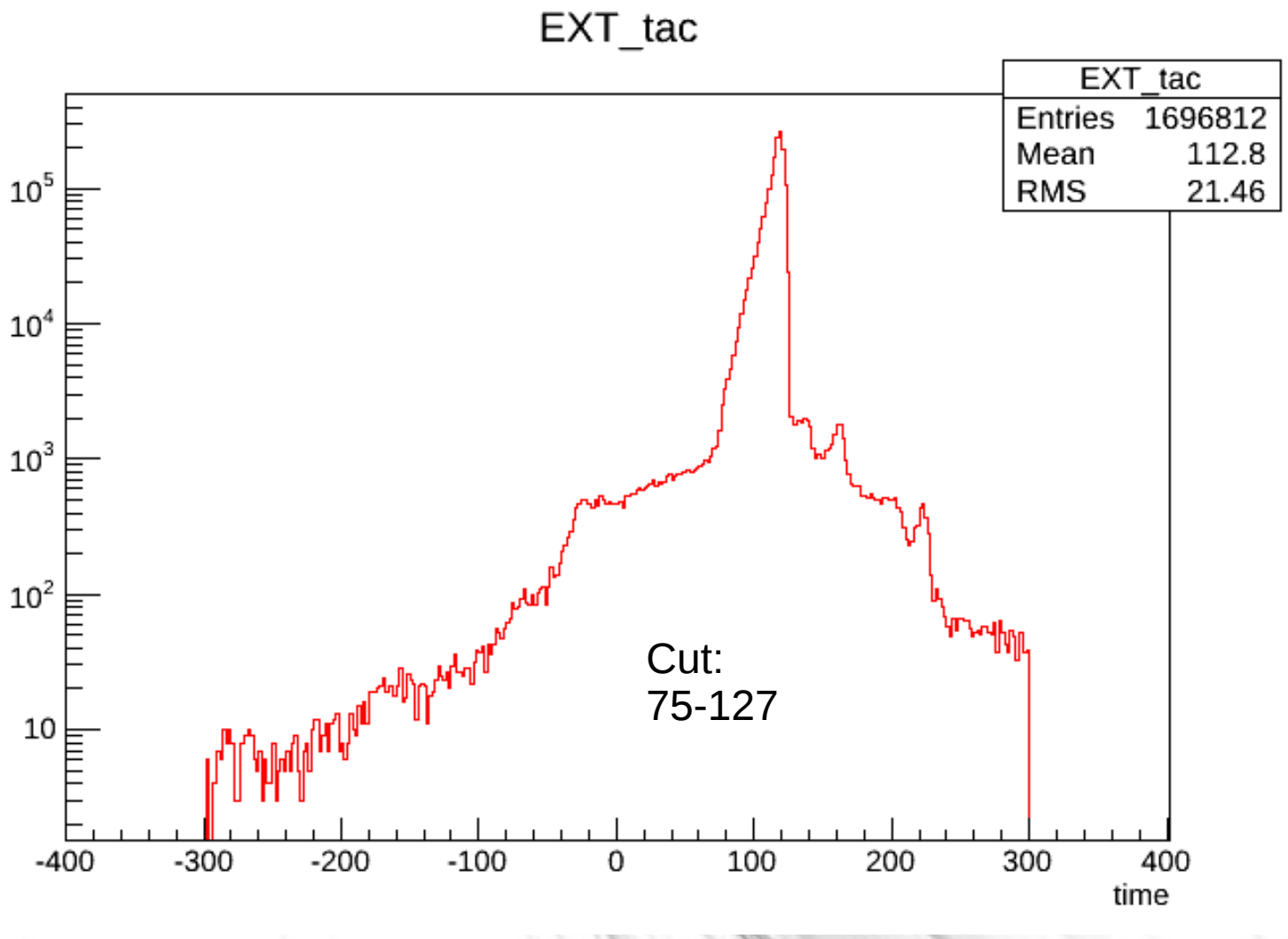
World map, all interaction points



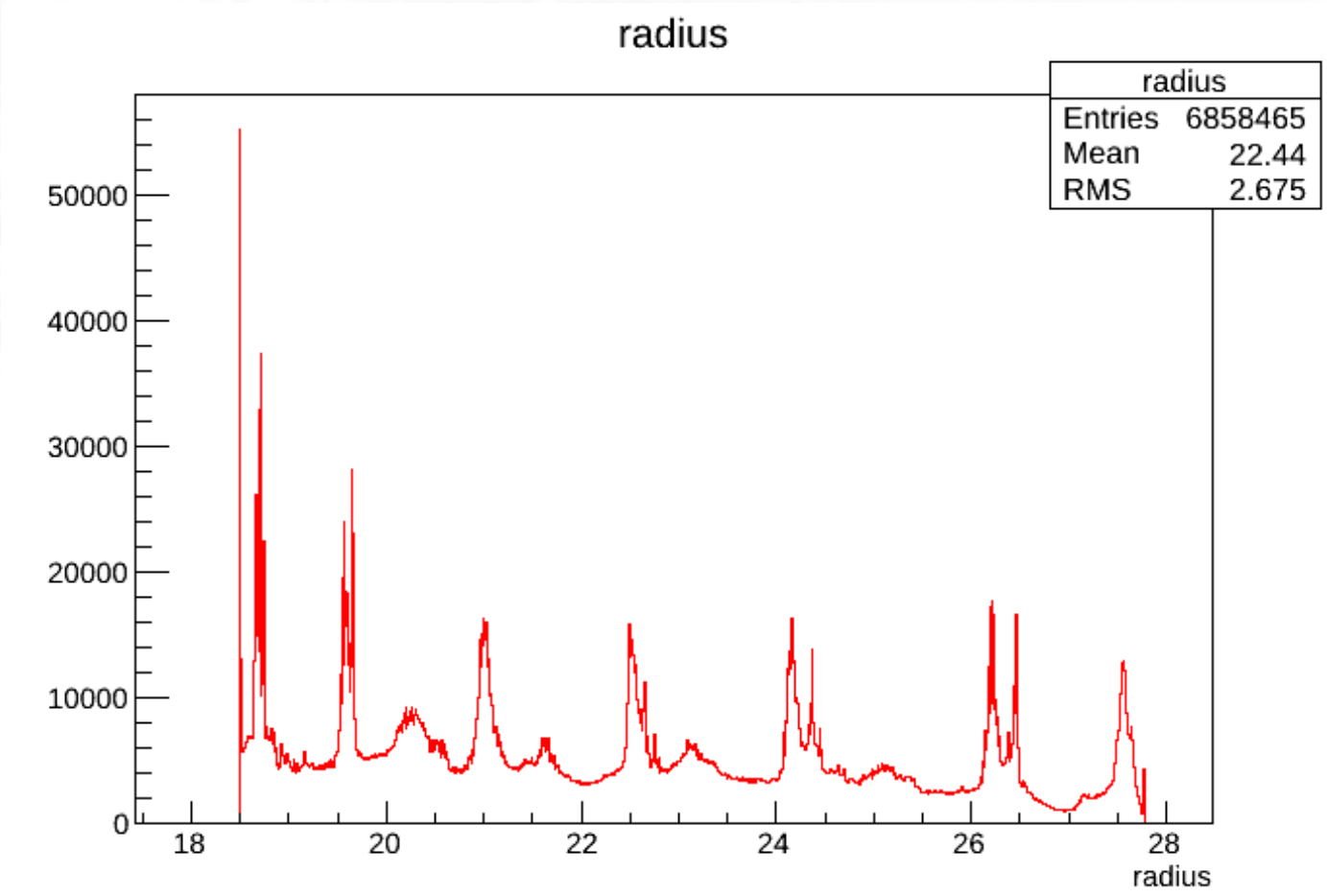
World map, first interaction points



Mod29 – GT spectrum



Not all is well with the decomposition:



Chat script inputs:

```
#------------------------------
# v/c for Doppler correction

# for MSU
beta 0.378
#------------------------------
# specify beam direction in GT cord system

;-- normal setup
beamdir 0 0 1

;-- for BGS setup
;beamdir 0.2764 0.8507 -0.4472
#------------------------------
# target position (in centimeters)

target_x -0.05
target_y  0.38
target_z  0.12
```

analyze.chat excerpts

```
greta_ata  0.247
greta_bta -0.277

#------------------------------
# S800 PID window
# d2("S800_PID")
# mk2dwin("S800_PID")
# save2dwin("test.win")
# d2dwin("test.win","S800_PID")

PIDwin test.win



#------------------------------
# debug print this many events

nprint 200
```

GSUtil can generate the 2D PID window

Prints events in ‘nice’ format

more trackedEvent_001.txt

Can specify a number of events to be written out in glory details

```
trackedEvent_001.txt
**Tracked data::
[n/m] n: tracked pos, m: original pos

 0:  (valid) ndet= 2 esum= 3.5851/bestperm=  0/FOM= 1.8200; (tracked)
 IP-> (-10.56 -15.46   5.13) r= 19.41  { 0}; e=  3.287/s=  3.287;  0, ts=33711783917;
        (-10.97 -15.57   5.97) r= 19.96  { 1}; e=  0.298/s=  3.585;  1, ts=33711783917;

We found  1 gamma rays

**input data to tracking::
Event Builder event # 1, #tracks=3
  num (crystalno/detpos) tot_e t0 chisq norm_chisq timestamp
ptgd->type = 1, ptgd->length = 464
 0> GEB_TYPE_DECOMP with timestamp 33711783917
#interactions= 2 ID=(03/17) [id=0x0047(71)] [detno=071] 3585.088 88.621 0.089 38.589
33711783917 pad= 0
  [0] (-10.56 -15.46   5.13);  r= 19.4cm  3287 {sum  3287/ 3585}
  [1] (-10.97 -15.57   5.97);  r= 20.0cm   298 {sum  3585/ 3585}
ptgd->type = 8, ptgd->length = 808
 1> GEB_TYPE_GT_MOD29 with timestamp 33711784037
ptgd->type = 9, ptgd->length = 72
 2> GEB_TYPE_S800PHYSDATA with timestamp 33711784064
```

To do

- Write tracked data out in standard gebData format (but, what do I set the timestamp to...)
- Add more refinements to ctkana, the root sorter (trees anyone!?)
- Play more with the tracking parameters when we have improved the detector basis
- Better documentation
- User input?
- User contributions?
- **Who is using this package?**

Could write data out in simpler format

- Number of gamma rays, for each:
- Egam
- FOM
- 1’th interaction point
- 2’nd interaction point