Tracker SAS Analysis

Objective: To expose the tools and an approach to understanding the GLAST Tracker

Initially the charge was to delineate what the data requirements were for the Tracker Analysis

This was by way of contributing to the "Run Plan" during I&T

The "End-to-End" committee and I&T detailed a sets of runs -

Let's evaluate this to see if its enough or too much!

Key to most HEP detector systems is the Tracking

- Knits together not only itself - but by projection to other systems affords a "built-in" test beam.

Elemental Tracker Data -

- Hit SSD channels
- Groups of adjacent hits = TkrClusters
  (Its from these that Pat. Rec. and Track Fits are made)
- Trigger Primative - 3-in-a-Row Information
- Time-over-Threshold (TOT) (once per layer)

Tracker Self Consistency -

- Reconstructed Trajectories match expectations.
  - Geometry
  - Efficiency (Reminder: expect ~ 100%)

A Trial Balloon: Image the individual SSD's with a goal of verifying

- Location
- Integrity (Wire Bonds)
- Efficiency

Idea was hatch with Eduardo in planning for this meeting

Est. Require ~ 1 Hit/mm^2

Implementation: Recover the local trajectory location where there are missing SSD layers (gaps).

Problem: No such variables available EITHER in the TDS or the Merit nTuple.

Solution: Write a little code in the Analysis Package

Analysis Package

Purpose: Provide subsystem and system wide parameters for Event Analysis

leit-motif: Provide a collection of templates and examples on how to use the system (TDS & Services & Tools).

A set of classes where each class focuses on calculating variables associated with a subsystem - or several together
All have names like McValsTool, CalValsTool, TkrValsTool,...

Start by looking at TkrValsTool.cxx as this involves only Tkr Data.

```
// Section to dig out the TOT information
    ..........................................................
 int gapId = -1;
int lastLayer = -1;
while(pln_pointer != track_1->end()) {
      Event::TkrFitPlane plane = *pln_pointer;
      int thisPlane = plane.getIDPlane();
      int thisIview = Event::TkrCluster::viewToInt(plane.getProjection());
     int thisLayer = 0;
      if(thisPlane%2 != 0) thisLayer = 2.*thisPlane+thisIview;
      else                 thisLayer = 2.*thisPlane+((thisIview+1)%2);

       if(lastLayer < 0) { //First Hit
       lastLayer = thisLayer;
       }
       else {
```

(I added the gap stuff to this loop)

Begin Loop Planes

Computer Layer# from Plane & View

(Probably should be added to TkrFitPlane)

Continued on the next page....

```
if(gapId < 0 && lastLayer+1 != thisLayer){
      gapId = lastLayer+1;
      Event::TkrFitPlane lastPlane = *(--pln_pointer);
     pln_pointer++;
      Point lastPoint = lastPlane.getPoint(Event::TkrFitHit::FIT);
     Event::TkrFitHit lastHit = lastPlane.getHit(Event::TkrFitHit::FIT);
      double xSlope = lastHit.getPar().getXSlope();
      double ySlope = lastHit.getPar().getYSlope();
      Vector localDir = Vector(-xSlope,-ySlope,-1.).unit();
      Ray localSeg(lastPoint, localDir);
      int view;
      int layer;
      pTkrGeoSvc->planeToLayer (gapId, layer, view);
      double gapZ = pTkrGeoSvc->getReconLayerZ(layer,view);
     double arcLen = (gapZ-lastPoint.z())/localDir.z();
     Point gapPoint = localSeg.position(arcLen);
      Tkr_1_GapX = gapPoint.x();
      Tkr_1_GapY = gapPoint.y();
     }
}
lastLayer = thisLayer;
.....    // The Loop Goes On...
```

First Gap Trigger

Setup a local Track Segment

(Probably should be added to KalmanFilterUtil)

Compute ArcLength

Project & Store Results

Next Step: Setup a run and execute:

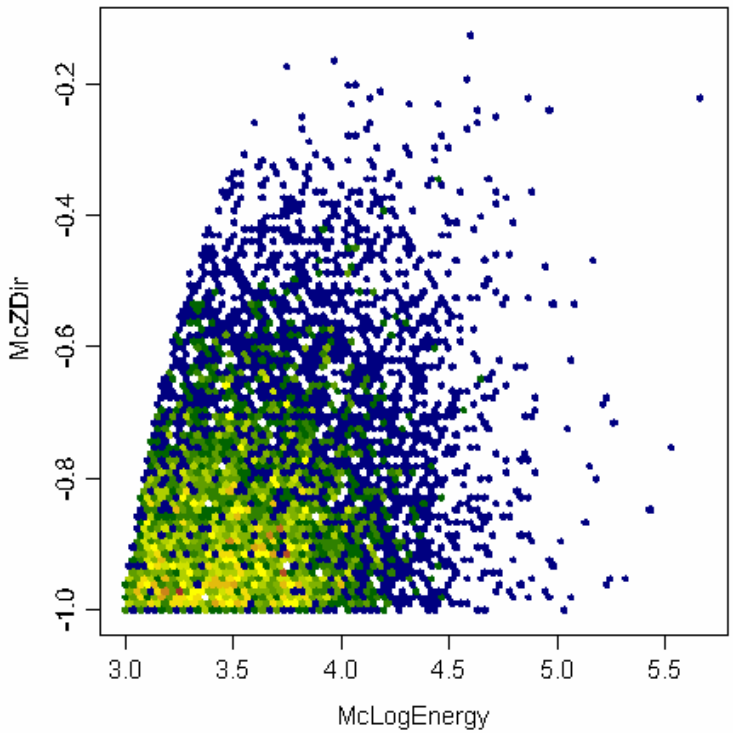
Use Flux Source "Surface Muons"
Run 50K Events
Set min. Energy for Finding/Fitting = 300 MeV

TkrInitSvc.TkrMinEnergy = 300;

Results: 10.3 K Events in Tuple

Cuts:
TkrNumTracks > 0
CalCsIRLn > 2

Leaves 6.8 K Events

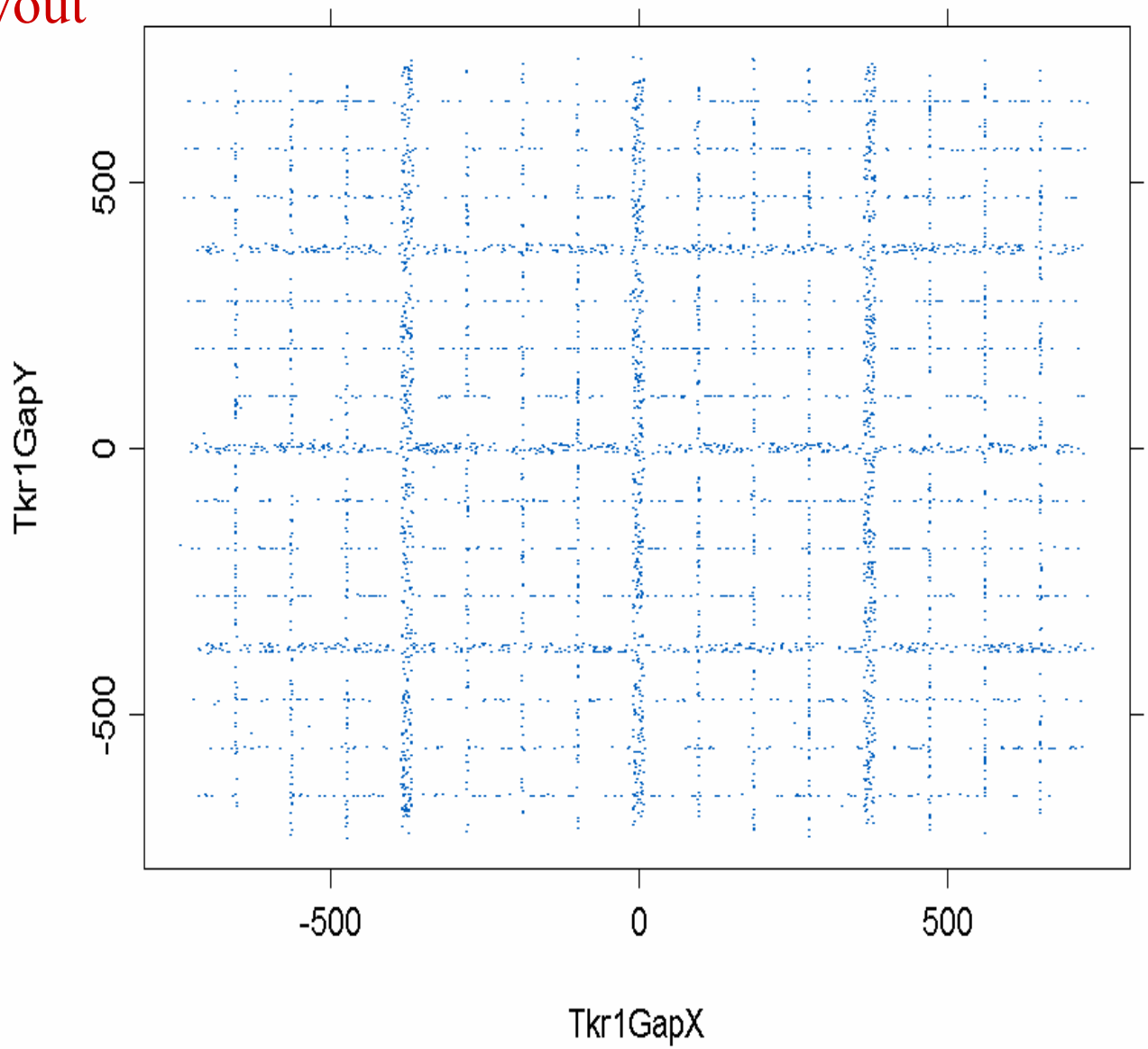


Require: Tkr1Gaps > 0

This is was the original idea

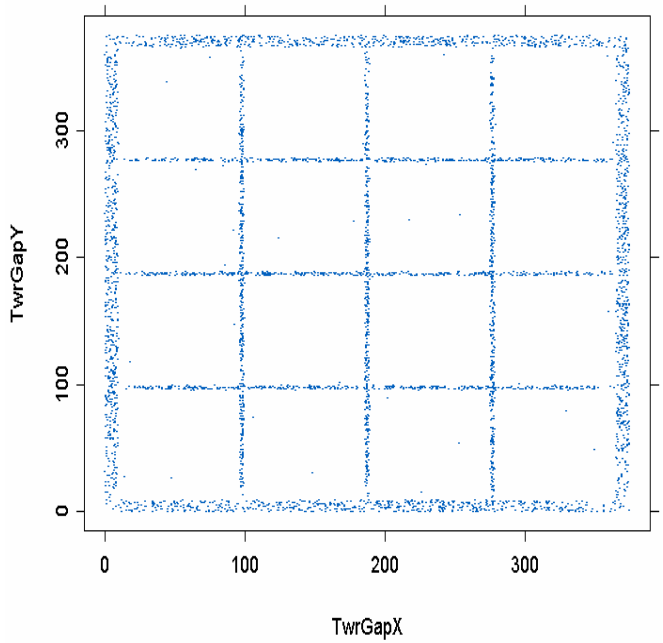
Image the Tracker Layout

Leaves 4.7 K Events



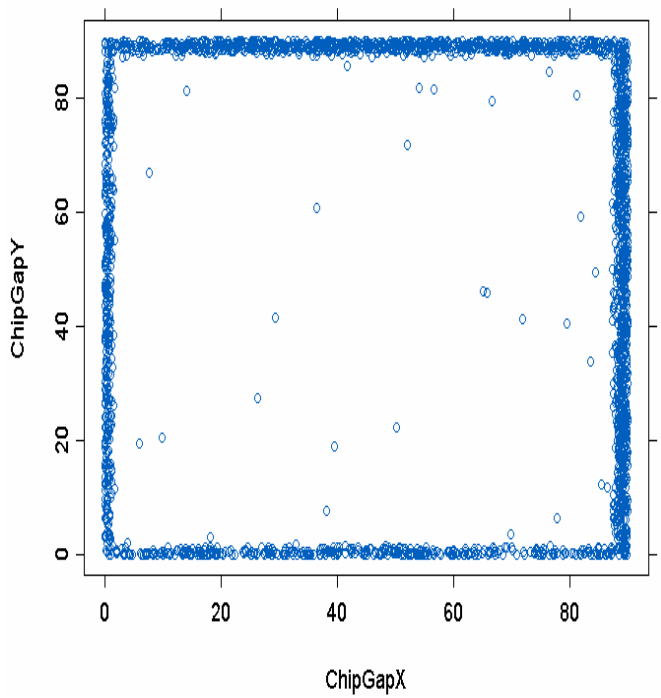
Collapse down onto one Tower

Cut off the edges
(8 mm)



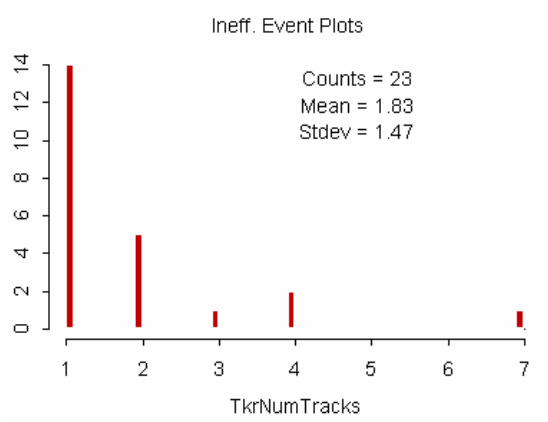
Collapse down onto one SSD

Cut off the edges ...
(4 mm)

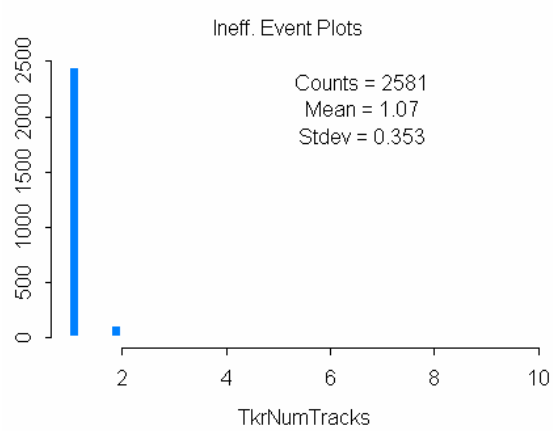


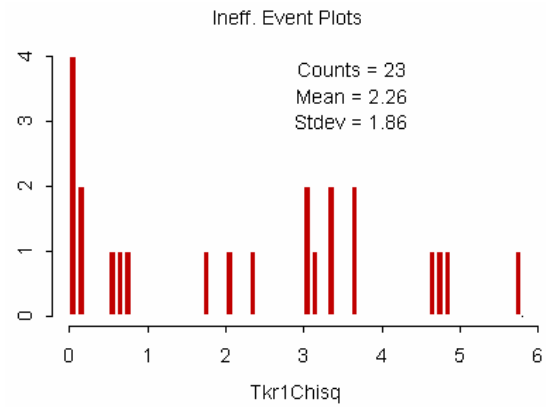
These are the events that shouldn't be... Lets Explore!

InEff. Events



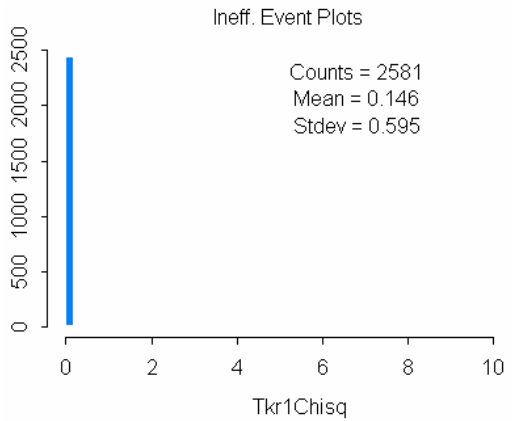
Require 1 Track

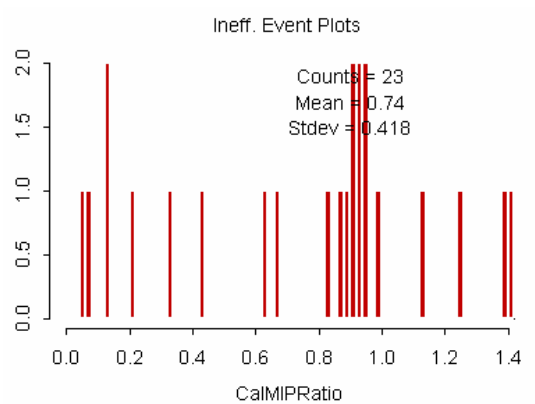




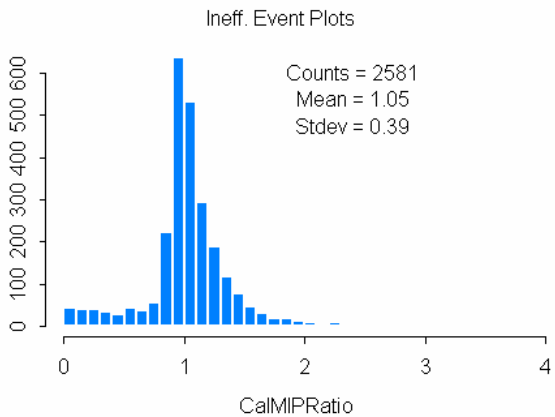
Require Small χ^2

Good Events





Require a MIP

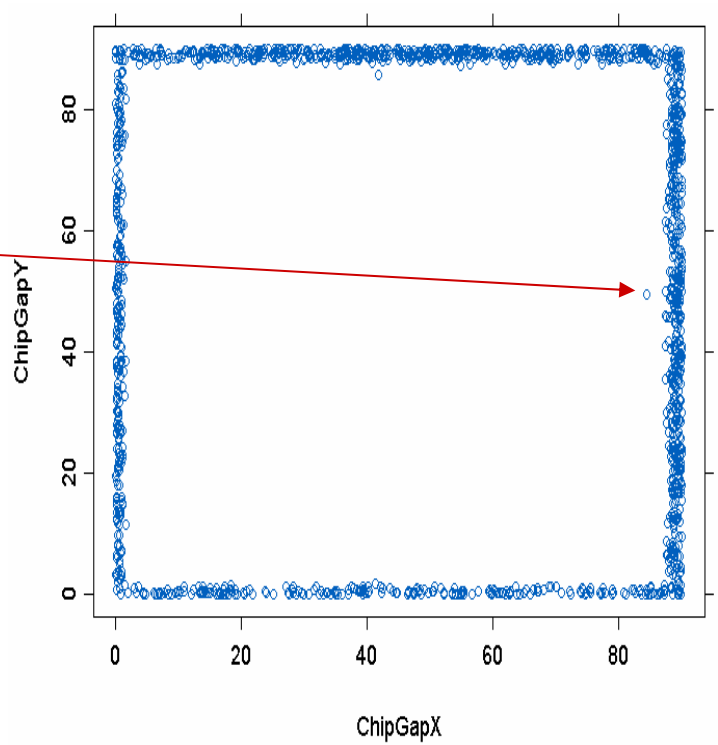


This suggests we require:

1) Tkr1Chisq < .2
2) .8 < CalMIPRatio < 1.2
3) TkrNumTracks == 1

Leaves: 4.1 K Events (~ 40% Efficiency)

This leaves 1 Ineff. Event



What's Been Learned ?

1) The cosmic source of particles can be "Cleaned Up" thus affording highly efficient tests. Suspect the real world will be worse however!

2) The efficiency of this is ~ 40% (See above)

3) Broken wire bonds, etc. will be readily visible at 1 Hit/mm^2 . In this example that would translate into

375^2 x 16 x 2.5 (Eff.) x 2 (Layer Factor) = 11.3 M Triggers (Full Lat)

375^2 x 2 x 2.5 (Eff.) x 4 (Layer Factor) = 2.8 M Triggers (2 Towers)

(or about 1/2 day - 60 Hz Rate Trig. Rate assumed)

Where to go from here?

1) Repeat analysis with 2 Tower Data and Updated Surface Cosmic Rays Source

2) Find out what that remaining event was!

Mystery Event

TkrThinHits = 24
TkrThickHits = 8
TkrBlknHits = 4

Why didn't all the hits wind up on the track????