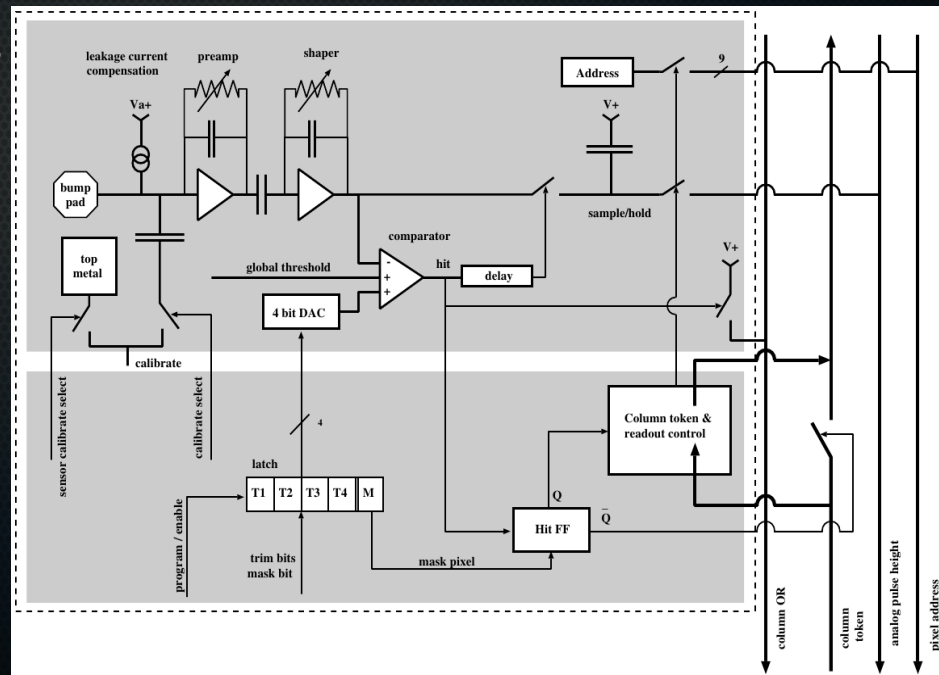
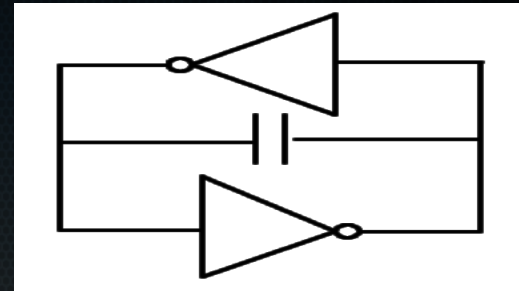


# Automatic Detection of Single Event Upsets In the Read Out Chips of the CMS Pixel Detector

Luigi Calligaris (DESY)

# Effects of Single Event Upsets

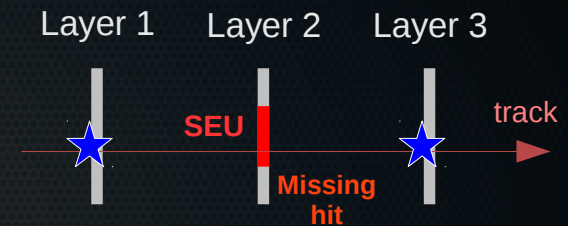
- Pixel Detector: high rate of ionizing particles
- A bit can be flipped (SEU) by injected charge
- ROCs design protects them against SEUs
- Still, sometimes a SEU will happen in a critical bit register
  - Unexpected behaviour
  - Often the chip goes totally/partially silent
- Not possible to read ROC state (design choice)
  - Need for data-driven monitoring



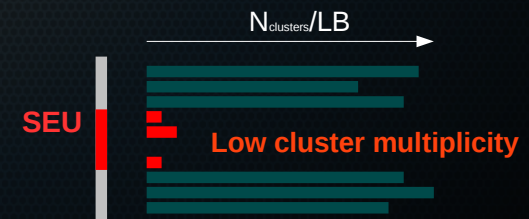
# Indicators for a silent ROC

- Hit efficiency for tracks, multiplicity of digis (pixels above threshold), multiplicity of clusters
- Advantages of using clusters
  - Many clusters compared to tracks
  - Do not depend on tracking
  - Computationally cheap compared to tracks
- Aim: determine bad ROCs with a fine time granularity
  - Good data produced by them is not wasted
  - Useful to understand the phenomenon
- Observe: cluster occupancy for each ROC in each LumiBlock
  - Flag LumiBlocks where a ROC was bad

Hit efficiency



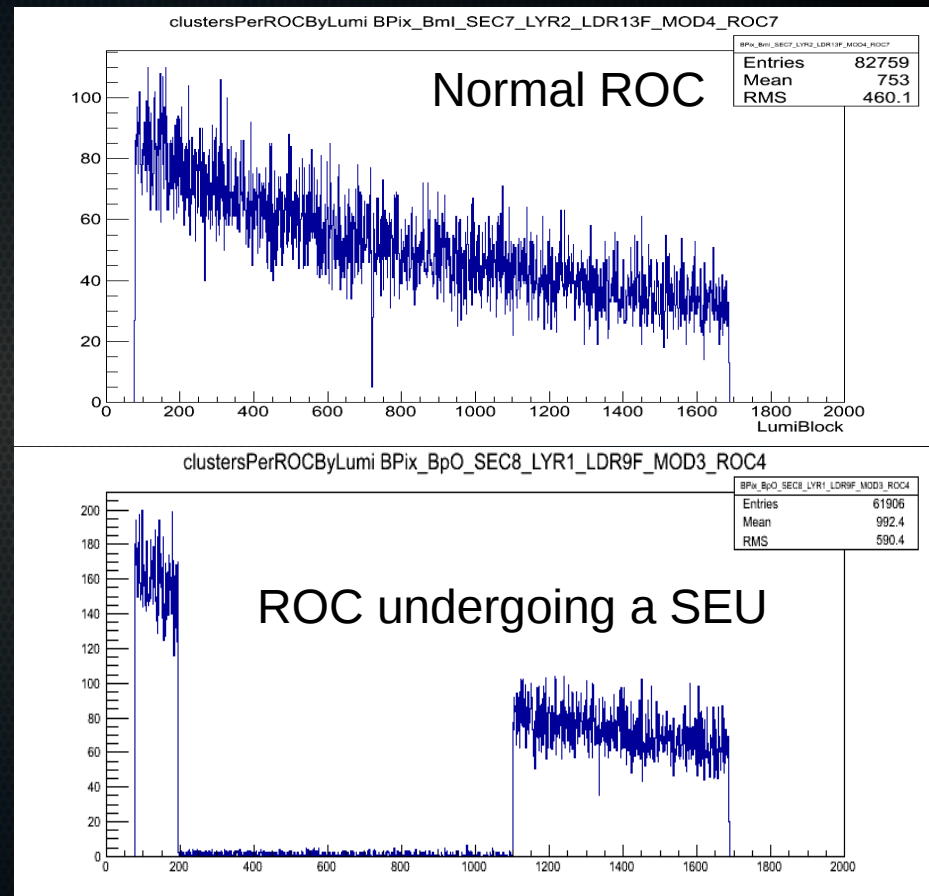
Cluster Multiplicity

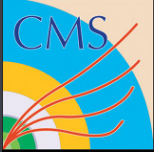




# Bare cluster occupancy

- Count of clusters in that ROC for each LumiBlock (= 23 s)
- Values depend on instantaneous Luminosity
  - Bad for setting a fixed threshold
  - Time dependence
- Need to evolve to a variable that factorizes out
  - Instantaneous luminosity
  - Pile Up

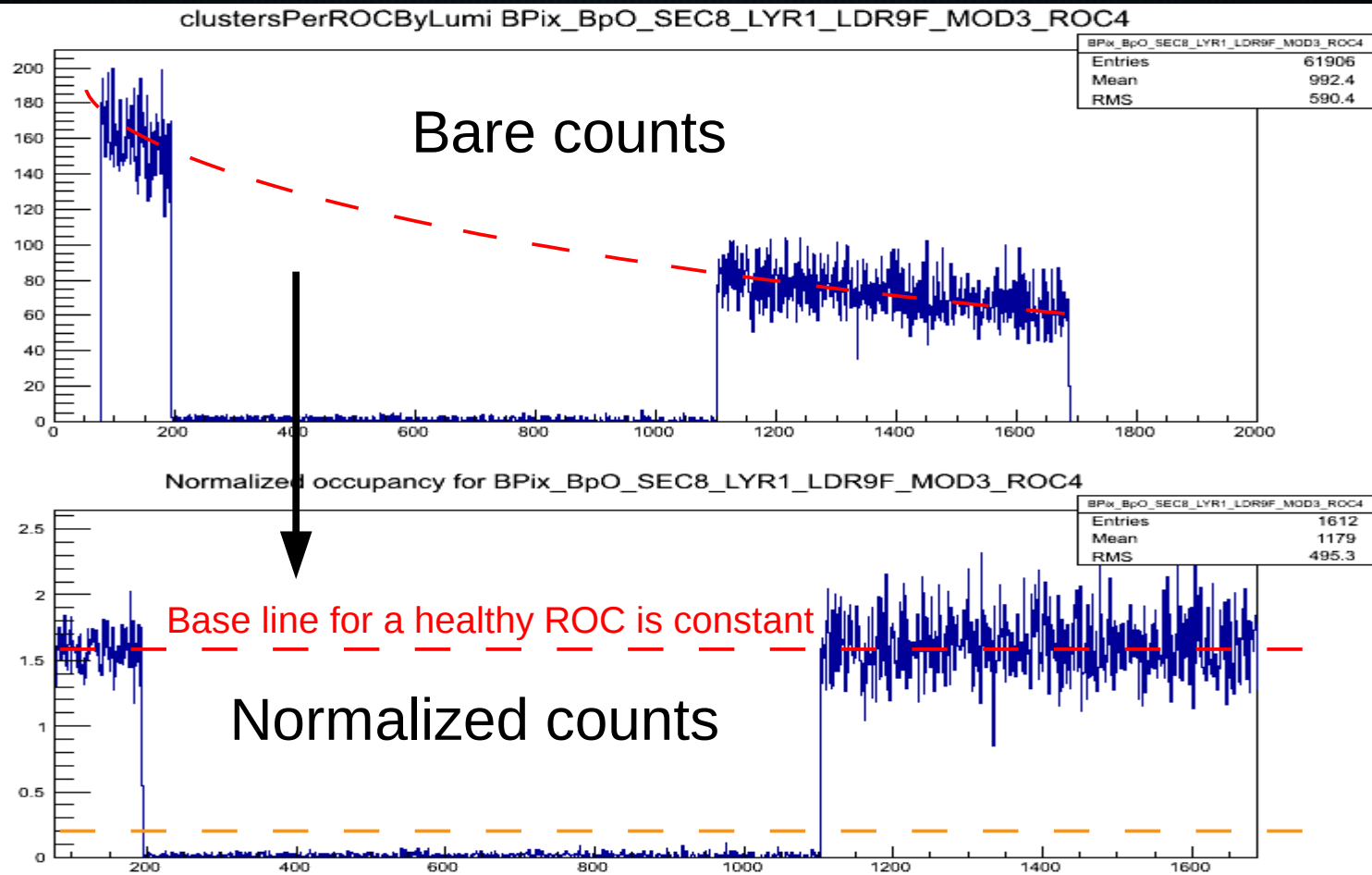




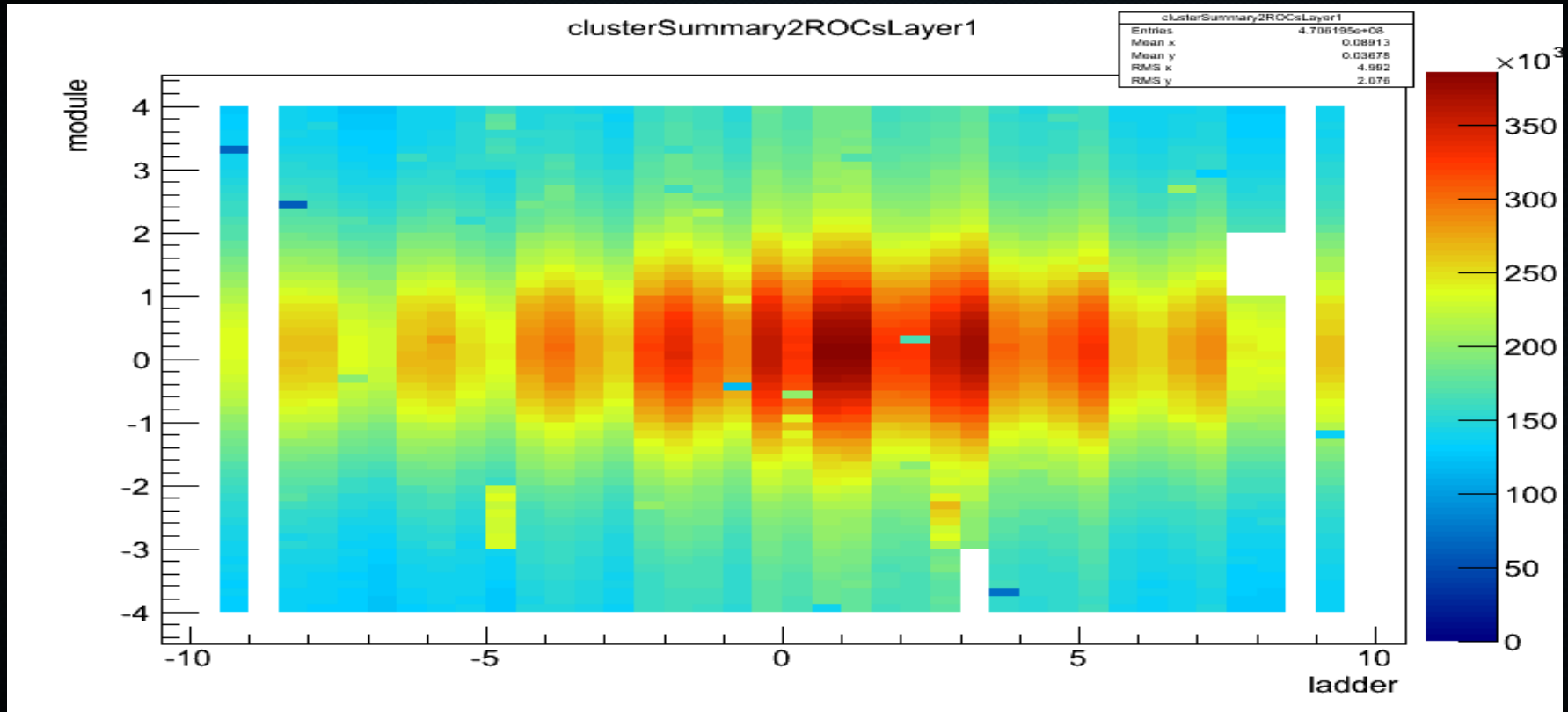
# Normalized cluster occupancy

- $O_{\text{normalized}} = (N_{\text{clusters in a ROC}} / N_{\text{clusters in all ROCs}}) * N_{\text{ROCs}}$
- Factorizes out the luminosity dependence
- Differences in  $\theta$  and layer expected
  - Acceptance changes
  - Particle flux changes
  - Beam spot not centered w.r.t. Barrel pixel
- Solution: calculate normalized occupancy in regions
  - Dividing detector along Z
  - Dividing detector in octants

# Normalized cluster occupancy

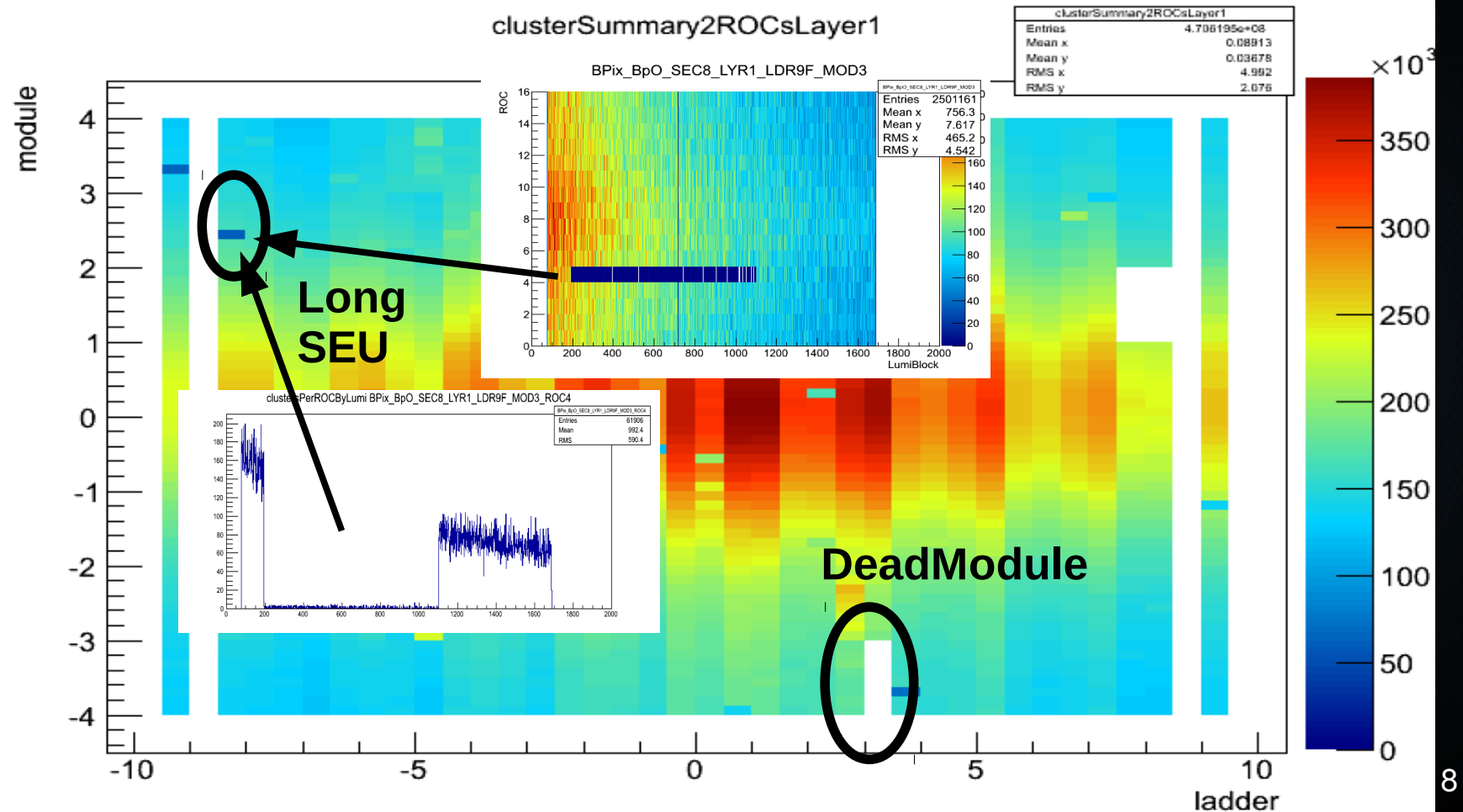


# Bare occupancy map (LYR1) Run 191226



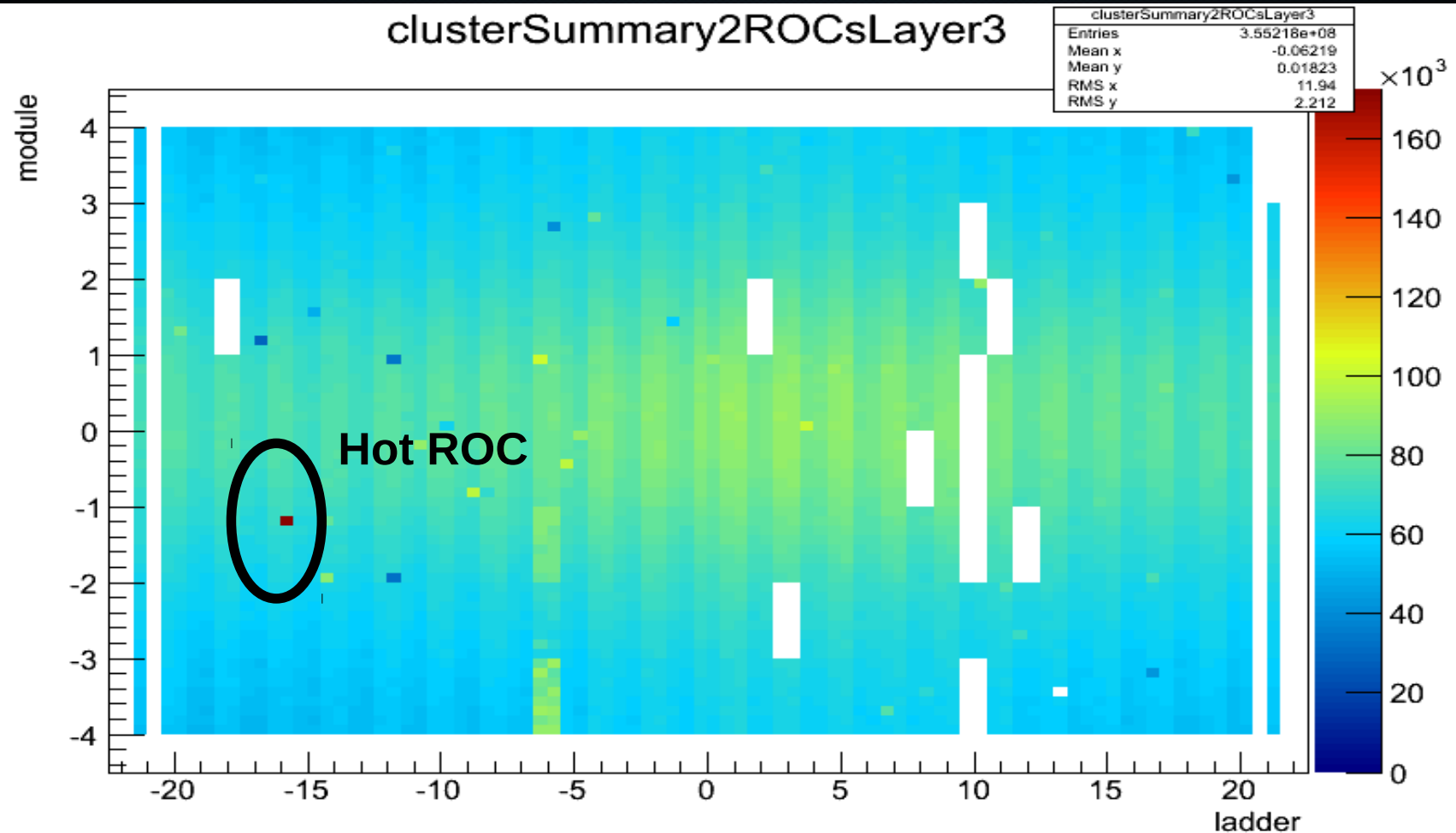
This summary map shows the cluster occupancy for each ROC, integrated over the whole run

# Bare occupancy map (LYR1) Run 191226

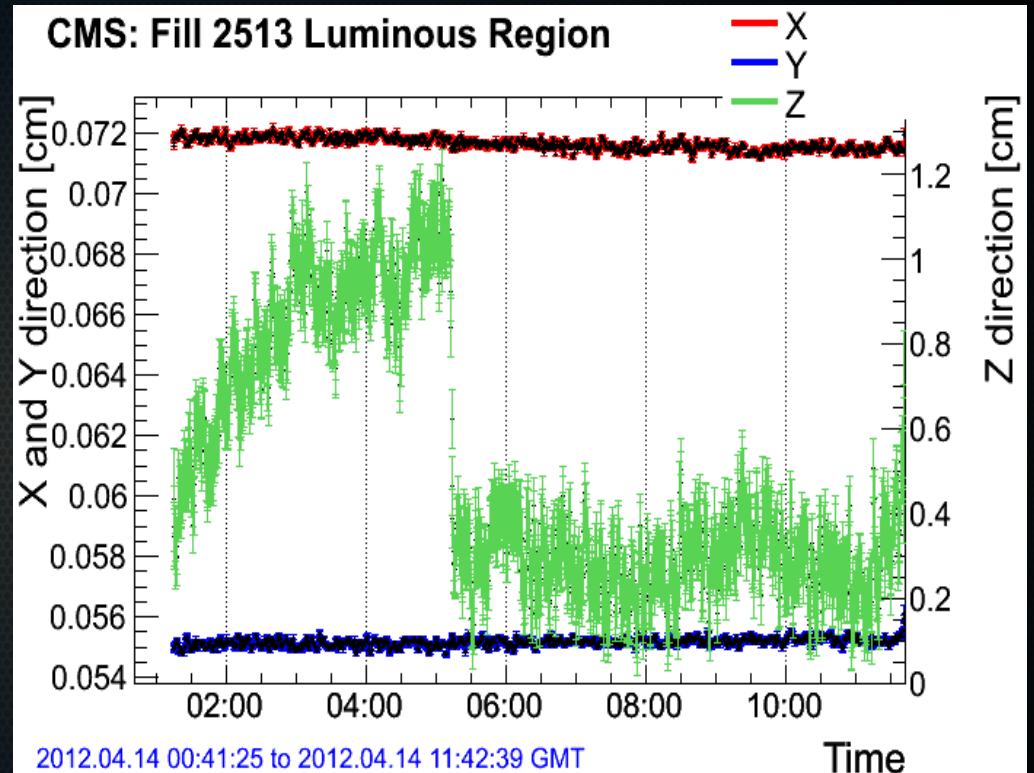
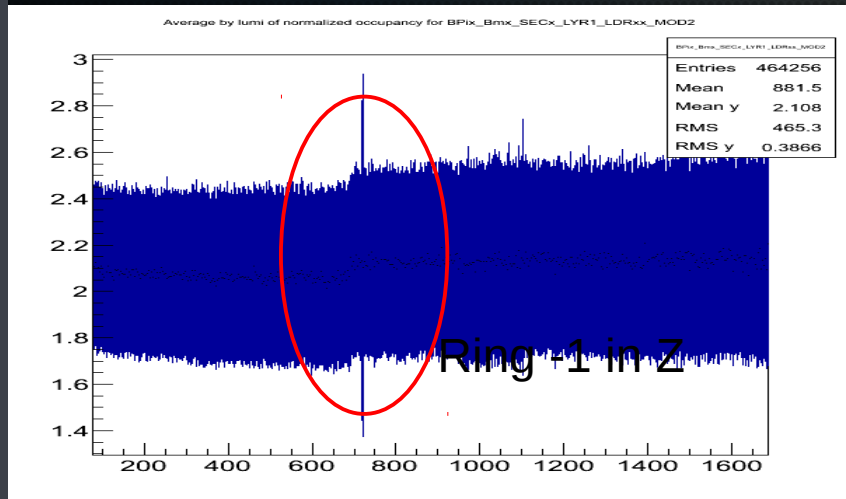
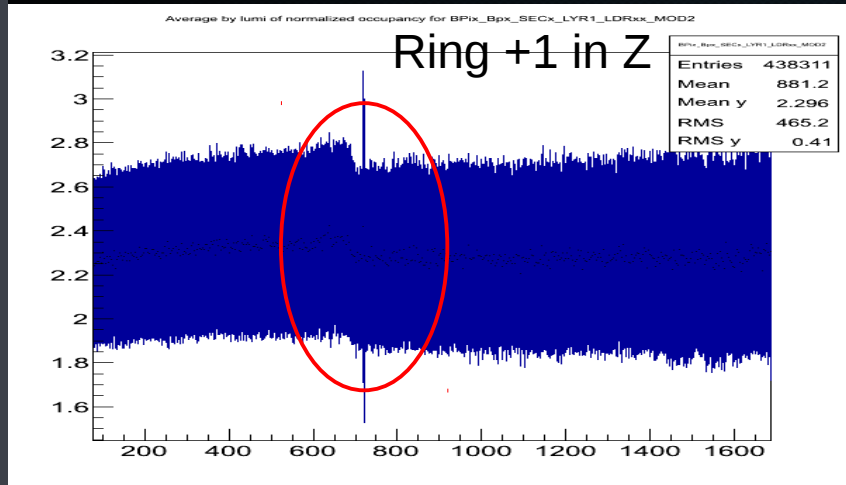




# Bare occupancy map (LYR3) Run 191226



# Cluster occupancy is quite sensitive...



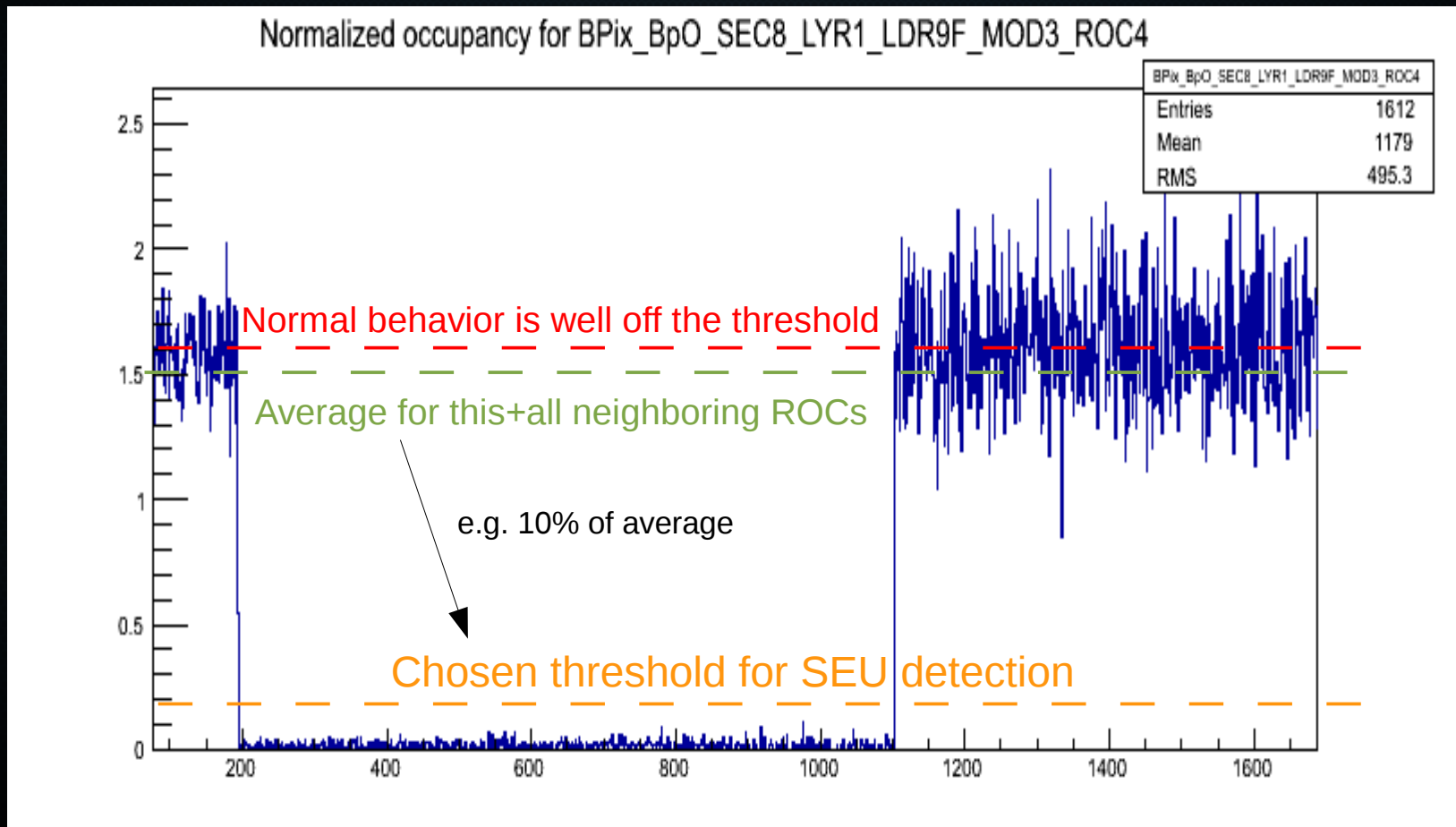
Sees the beam spot  
Moving in z by  $110 \mu\text{m}$



# Algorithm to find SEU

- For each LumiBlock
  - Compute the **average** of the normalized occupancy
  - Choose a fraction of this value (e.g. 10%) as **threshold**
  - Compare with the **normalized occupancy** of each ROC
- If the ROC is flat zero, put it in the list of dead ROCs
- If the ROC stays under the threshold for more than N (e.g. 6) consecutive LumiBlocks
  - Put in the list of SEU
  - Plot its evolution as a function of LumiBlock

# SEU seen from normalized occupancy

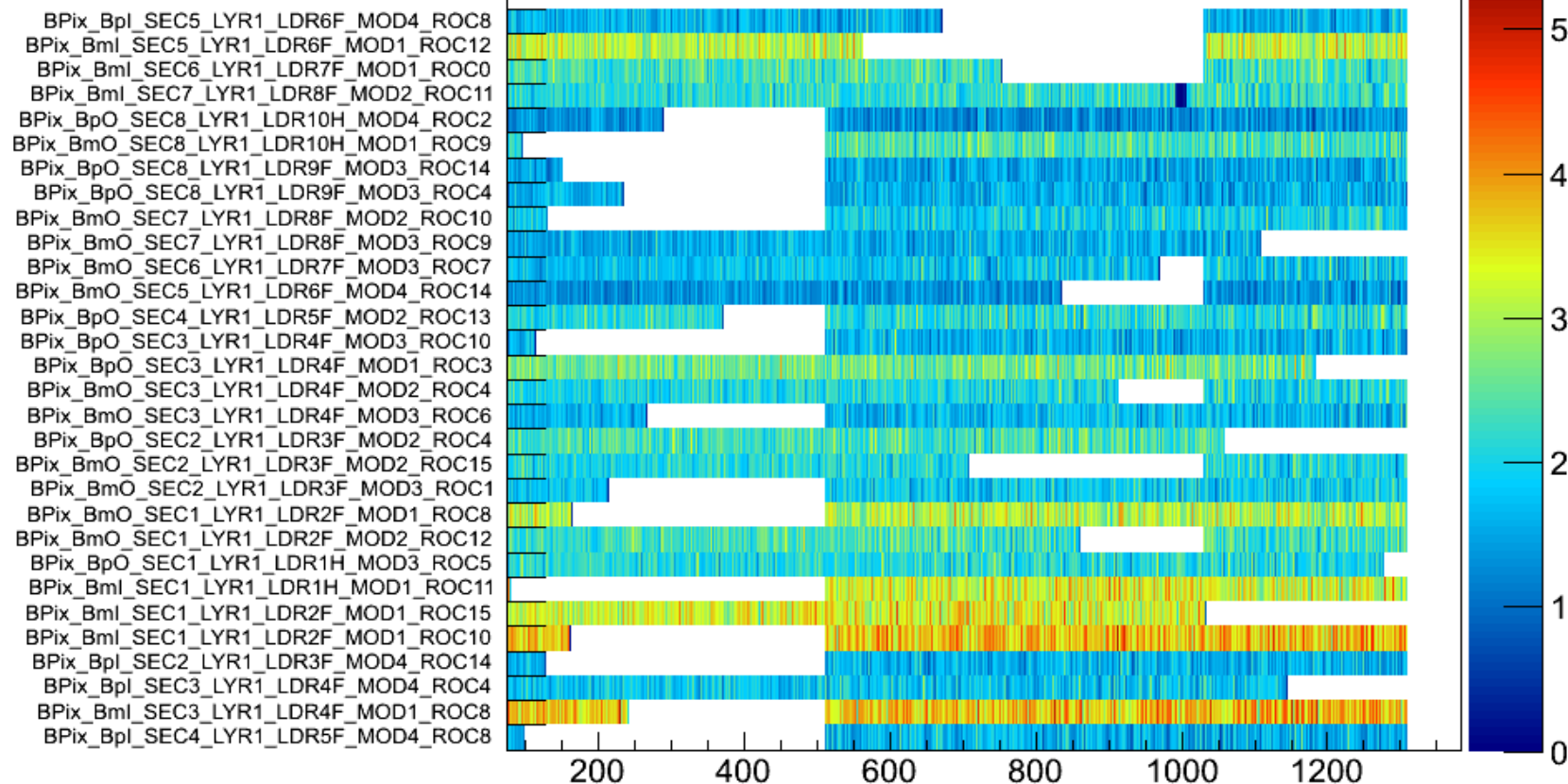




# Run 201191 – Layer1

SEUCandidates\_Layer1

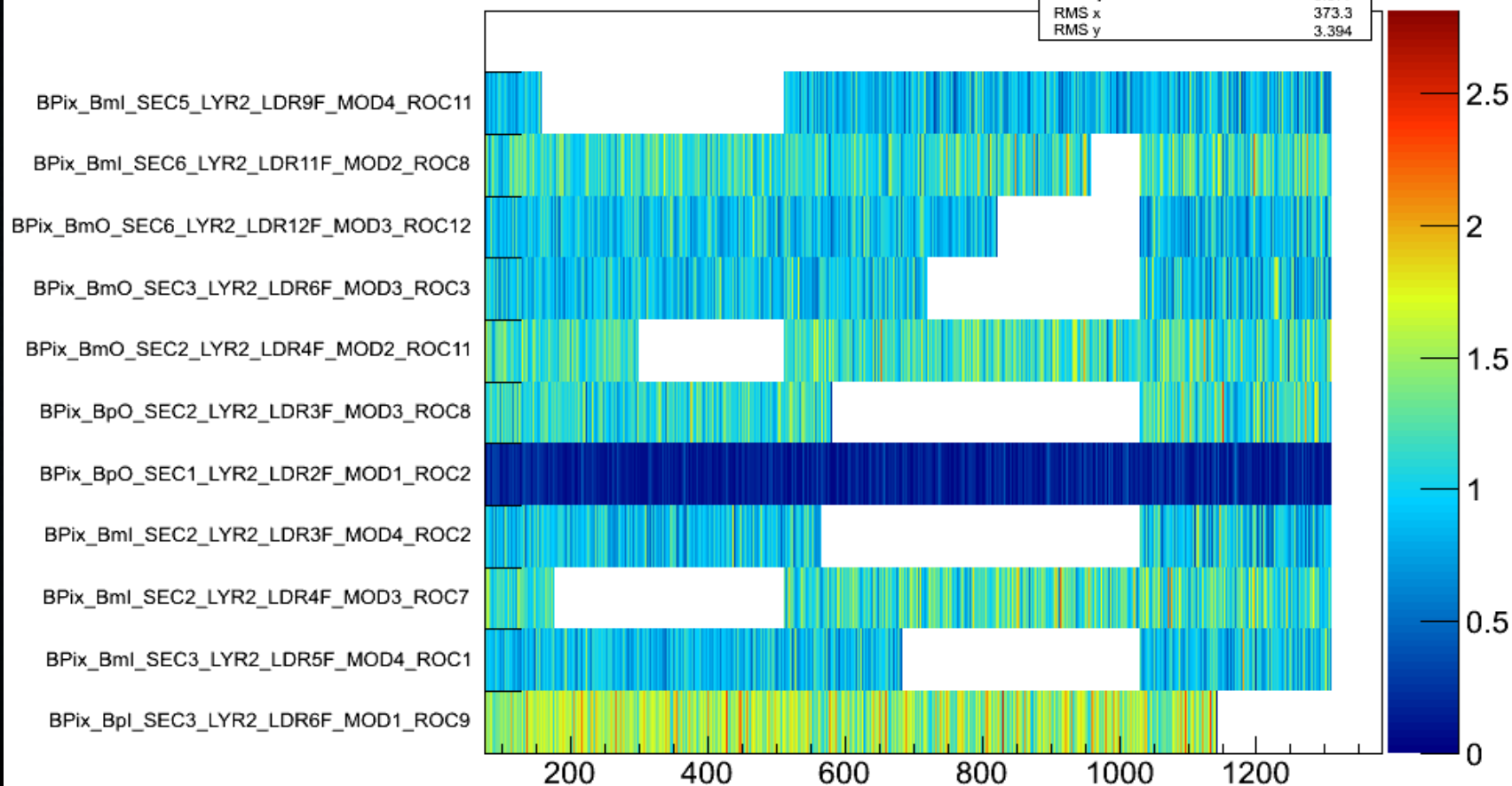
SEUCandidates_Layer1	
Entries	37050
Mean x	721.7
Mean y	14.03
RMS x	352.5
RMS y	8.669



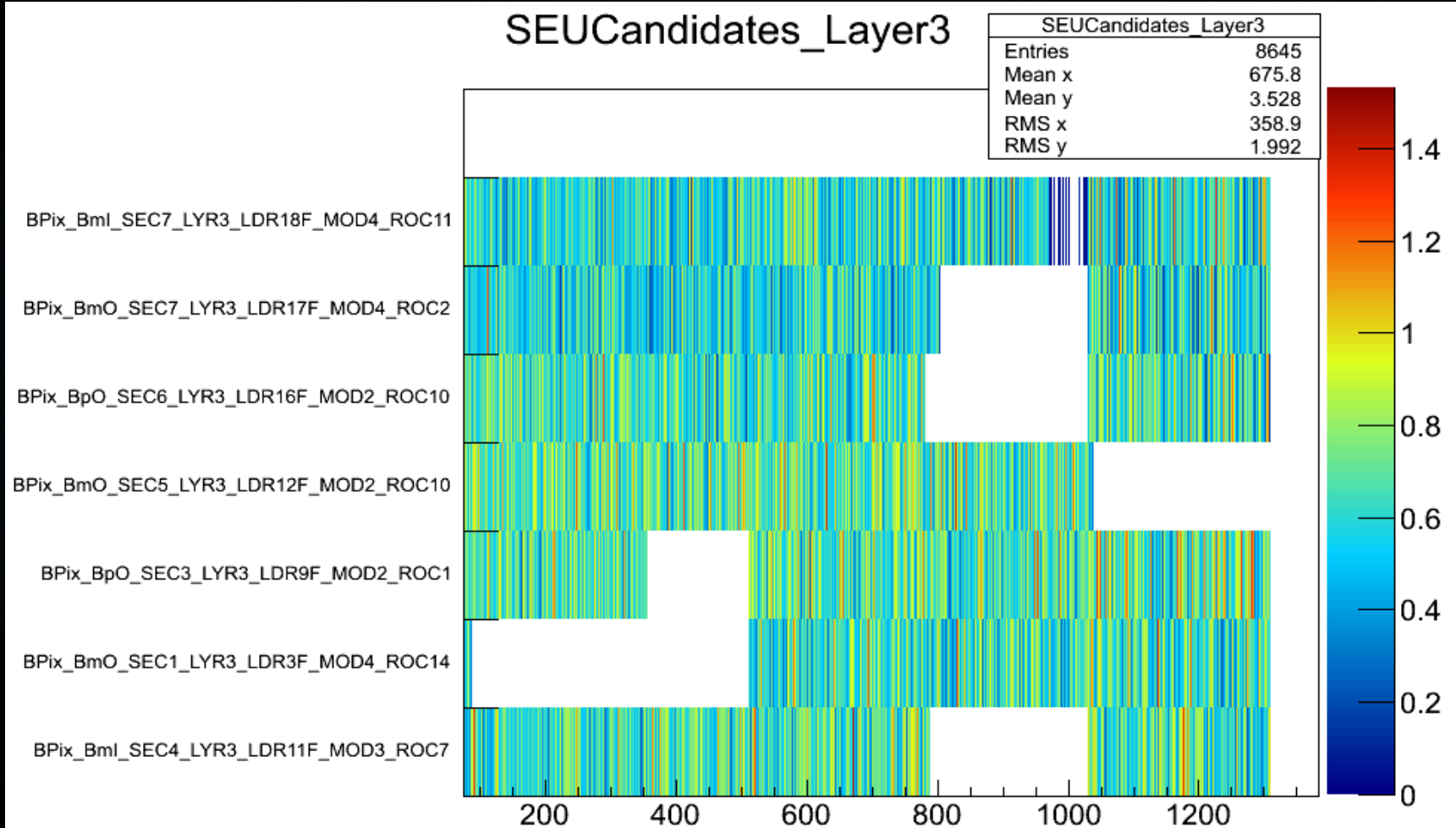
# Run 201191 – Layer2

SEUCandidates\_Layer2

SEUCandidates_Layer2	
Entries	13585
Mean x	683.3
Mean y	5.377
RMS x	373.3
RMS y	3.394



# Run 201191 – Layer3



# Summary & Outlook

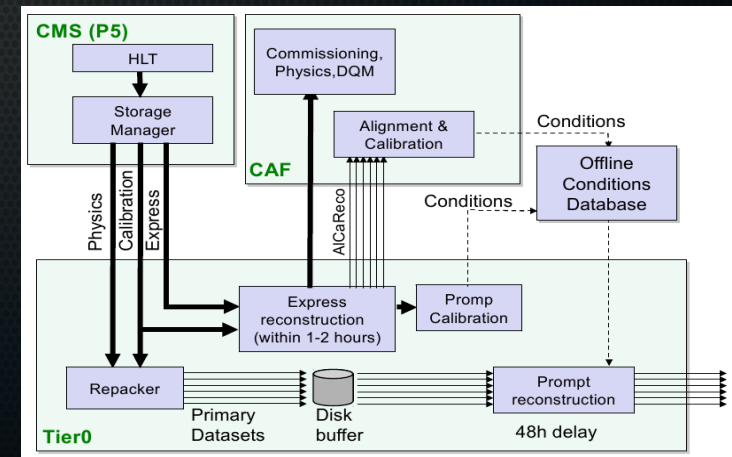
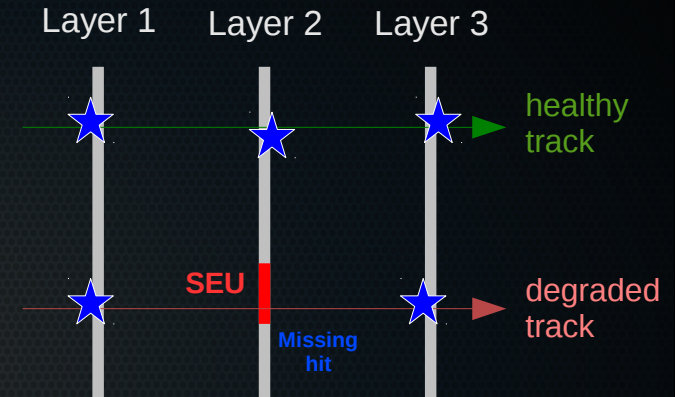
- Tested a proof-of-concept setup to find SEUs
  - SEUs are identified by the algorithm
- Next:
  - Re-write the code as a bundle of CMSSW modules (ongoing), aimed at running on ExpressStream
- Goal:
  - Identify automatically SEUs in the Pixel detector
  - Provide a list of dead ROCs and modules, to ease their book-keeping



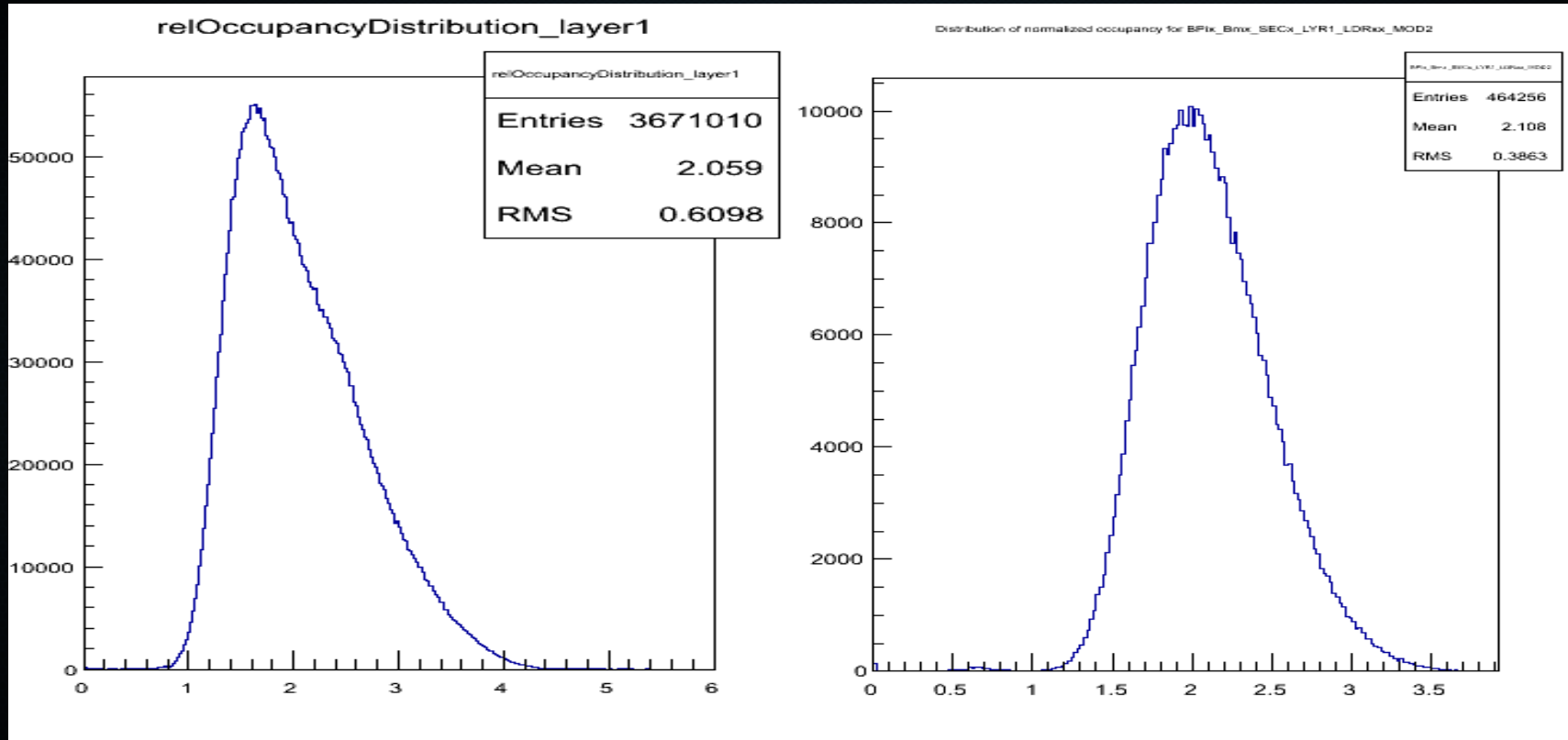
# BACKUP

## Tracking benefits from SEU detection

- Number of missing hits is a figure of merit for a track
  - A silent ROC pollutes good tracks with missing hits
- PromptReconstruction begins 48h after the end of run
- ExpressStream is available before for calibration, DQM, ...
- → use this data to find silent ROCs, and upload to DB
  - PromptReco: improved tracking



# Distribution of normalized occupancy

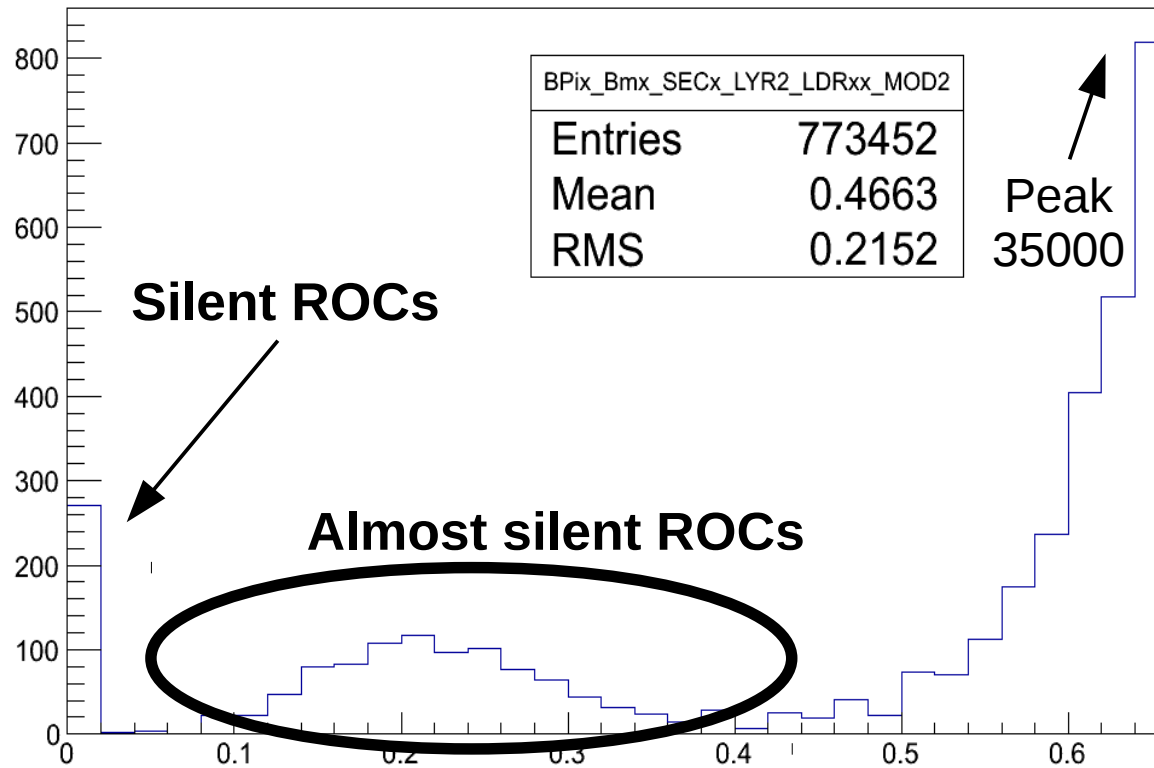


All Layer 1  
Integrated over run

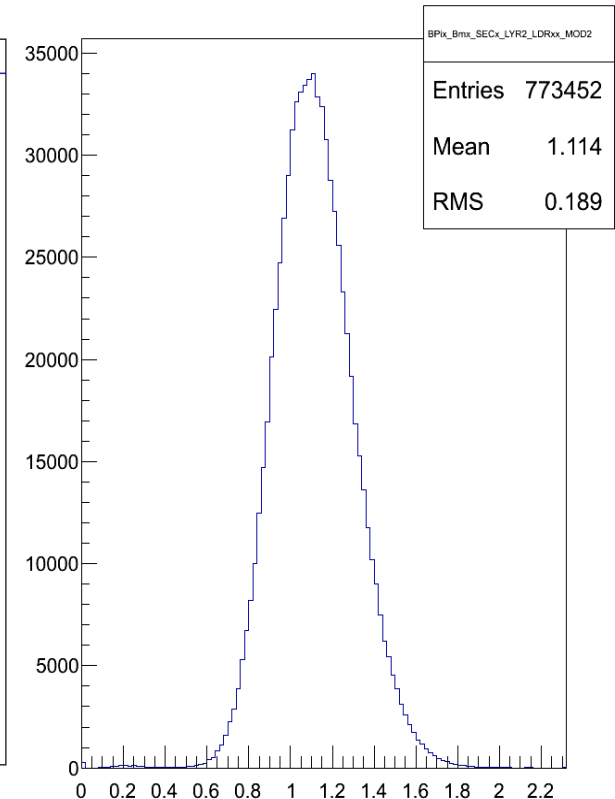
Bmx\_SECx\_LYR1\_LDRxx\_MOD2  
Integrated over run  
Specific layer and ring in z

# Low tail of the distribution for a single ring

Distribution of normalized occupancy for BPix\_Bmx\_SECx\_LYR2\_LDRxx\_MOD2



Distribution of normalized occupancy for BPix\_Bmx\_SECx\_LYR2\_LDRxx\_MOD2





# Implementation in CMSSW

# Foreseen Data Objects

- PixelROCIId
  - detId (uint32\_t) + roc number (uint32\_t)
  - Identifies uniquely a ROC
- PixelROCCounts
  - PixelROCIId + counts (uint32\_t)
  - Transports the counts across CMSSW modules
- PixelROCLumiBlockInterval
  - PixelROCIId + FirstSEULB + LastSEULB (both unsigned int)
  - Describes the interval in which a SEU was in effect

# Analisis workflow

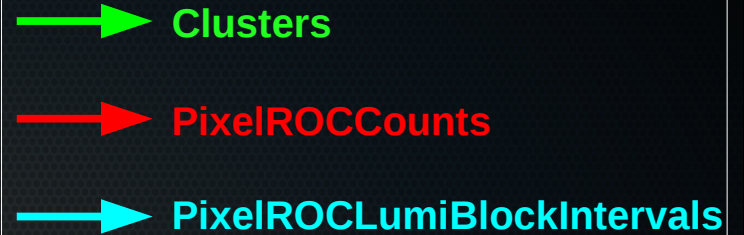
ExpressStream data



## SEUDetectorCounter (EDProducer)

- Counts clusters for each ROC and LB
- Stores the counts in the LumiBlock tree

Running on ExpressReco cluster



Persistent  
PixelROCCounts

SEUDetectorFlagger

SEUDetectorTimeFilter

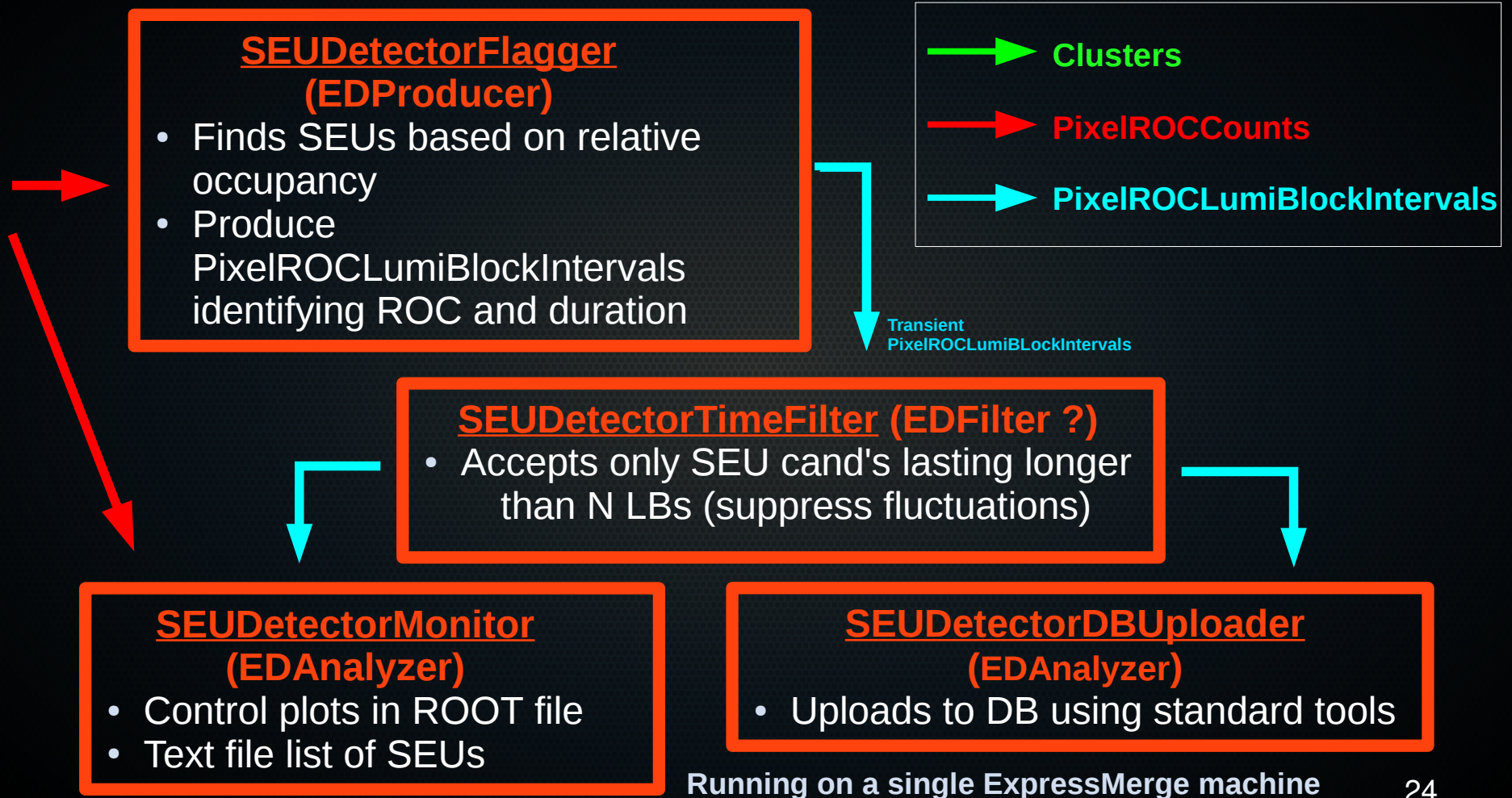
Details in the next page

SEUDetectorMonitor

SEUDetectorDBUploader



# Analisis workflow



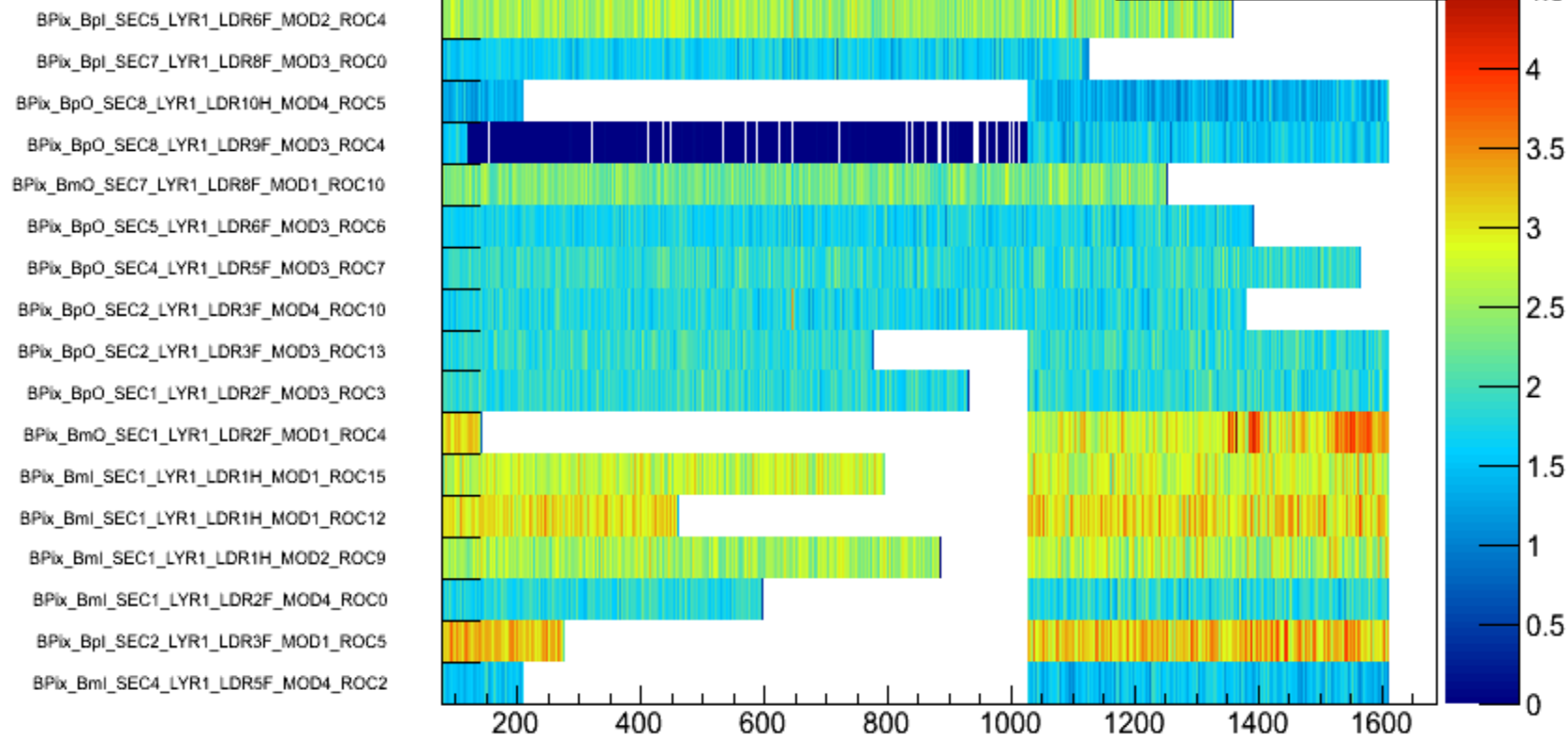


Some additional examples  
from different runs

# Run191226 Layer1

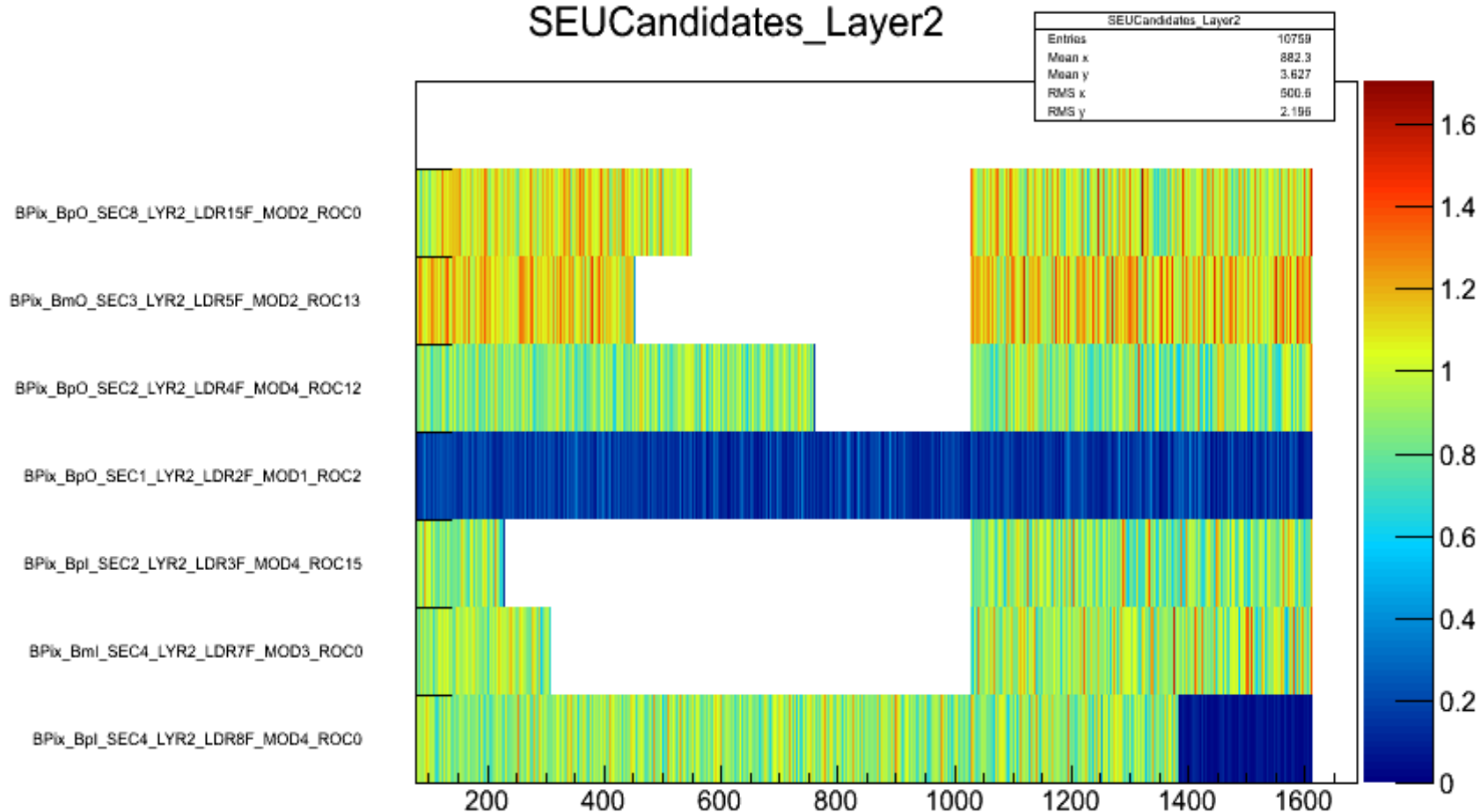
SEUCandidates\_Layer1

SEUCandidates_Layer1	
Entries	26129
Mean x	852.2
Mean y	8.211
RMS x	468.9
RMS y	4.68



# Run191226 Layer2

SEUCandidates\_Layer2

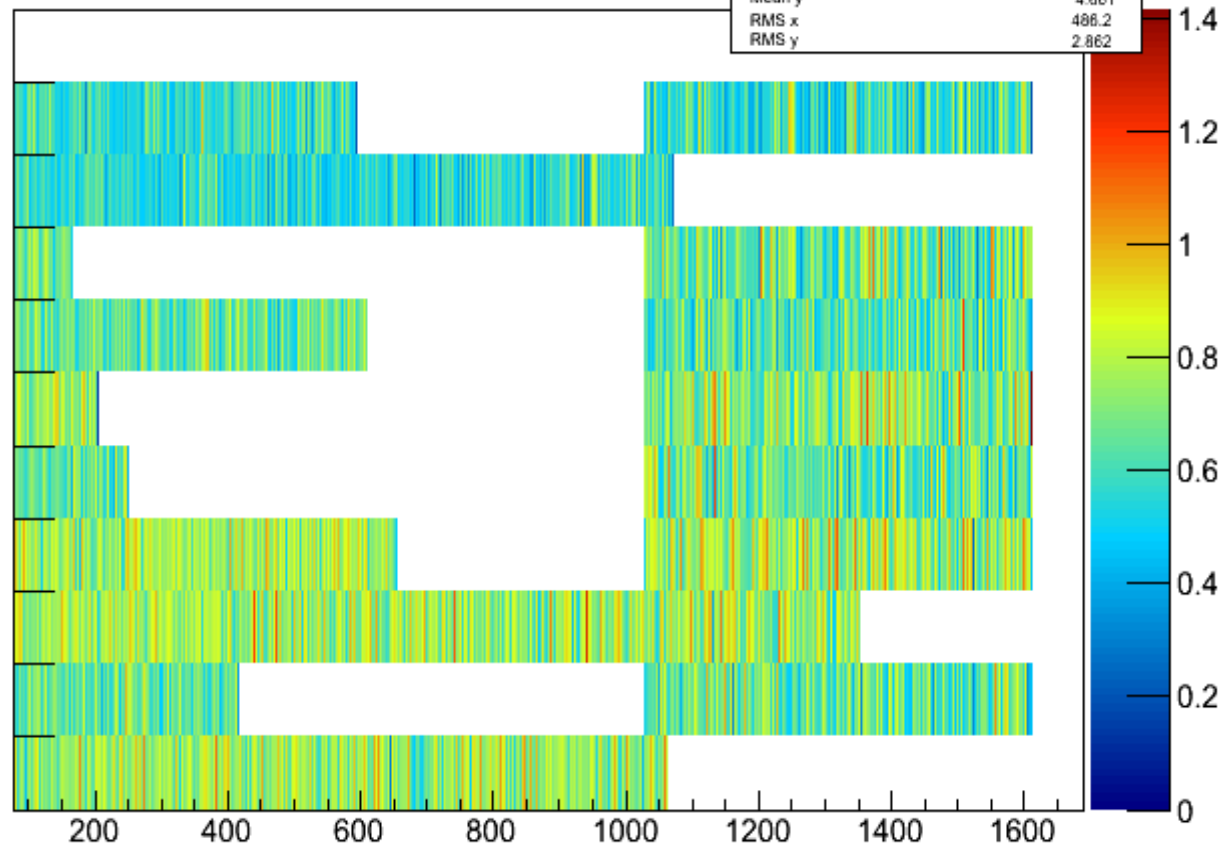


# Run191226 Layer3

SEUCandidates\_Layer3

SEUCandidates_Layer3	
Entries	15370
Mean x	839.1
Mean y	4.661
RMS x	486.2
RMS y	2.862

BPix\_BmI\_SEC7\_LYR3\_LDR18F\_MOD4\_ROC6  
BPix\_BpI\_SEC8\_LYR3\_LDR21F\_MOD4\_ROC10  
BPix\_BpO\_SEC7\_LYR3\_LDR18F\_MOD2\_ROC9  
BPix\_BpO\_SEC6\_LYR3\_LDR16F\_MOD2\_ROC12  
BPix\_BpO\_SEC5\_LYR3\_LDR13F\_MOD1\_ROC15  
BPix\_BmO\_SEC5\_LYR3\_LDR13F\_MOD2\_ROC0  
BPix\_BpO\_SEC4\_LYR3\_LDR11F\_MOD1\_ROC8  
BPix\_BmO\_SEC3\_LYR3\_LDR9F\_MOD1\_ROC14  
BPix\_BpO\_SEC3\_LYR3\_LDR7F\_MOD3\_ROC13  
BPix\_BpO\_SEC1\_LYR3\_LDR2F\_MOD2\_ROC4

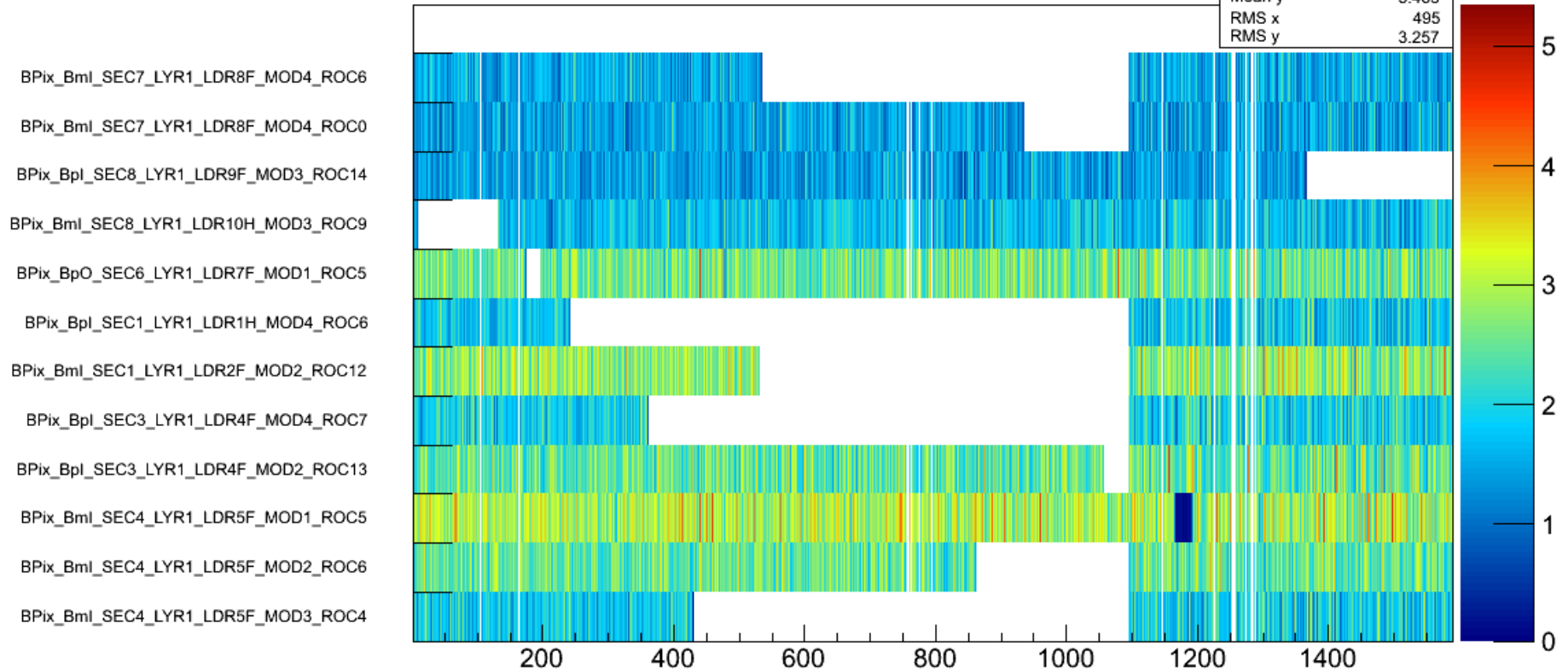




# Run 195950 - Layer1

SEUCandidates\_Layer1

SEUCandidates_Layer1	
Entries	17580
Mean x	799.8
Mean y	5.463
RMS x	495
RMS y	3.257



# Run 195950 - Layer2

SEUCandidates\_Layer2

SEUCandidates_Layer2	
Entries	14650
Mean x	776.6
Mean y	4.874
RMS x	484.8
RMS y	2.971

BPix\_Bpl\_SEC5\_LYR2\_LDR10F\_MOD4\_ROC10

BPix\_Bpl\_SEC6\_LYR2\_LDR11F\_MOD2\_ROC0

BPix\_Bml\_SEC7\_LYR2\_LDR14F\_MOD2\_ROC10

BPix\_BmO\_SEC4\_LYR2\_LDR8F\_MOD1\_ROC15

BPix\_BpO\_SEC1\_LYR2\_LDR2F\_MOD1\_ROC2

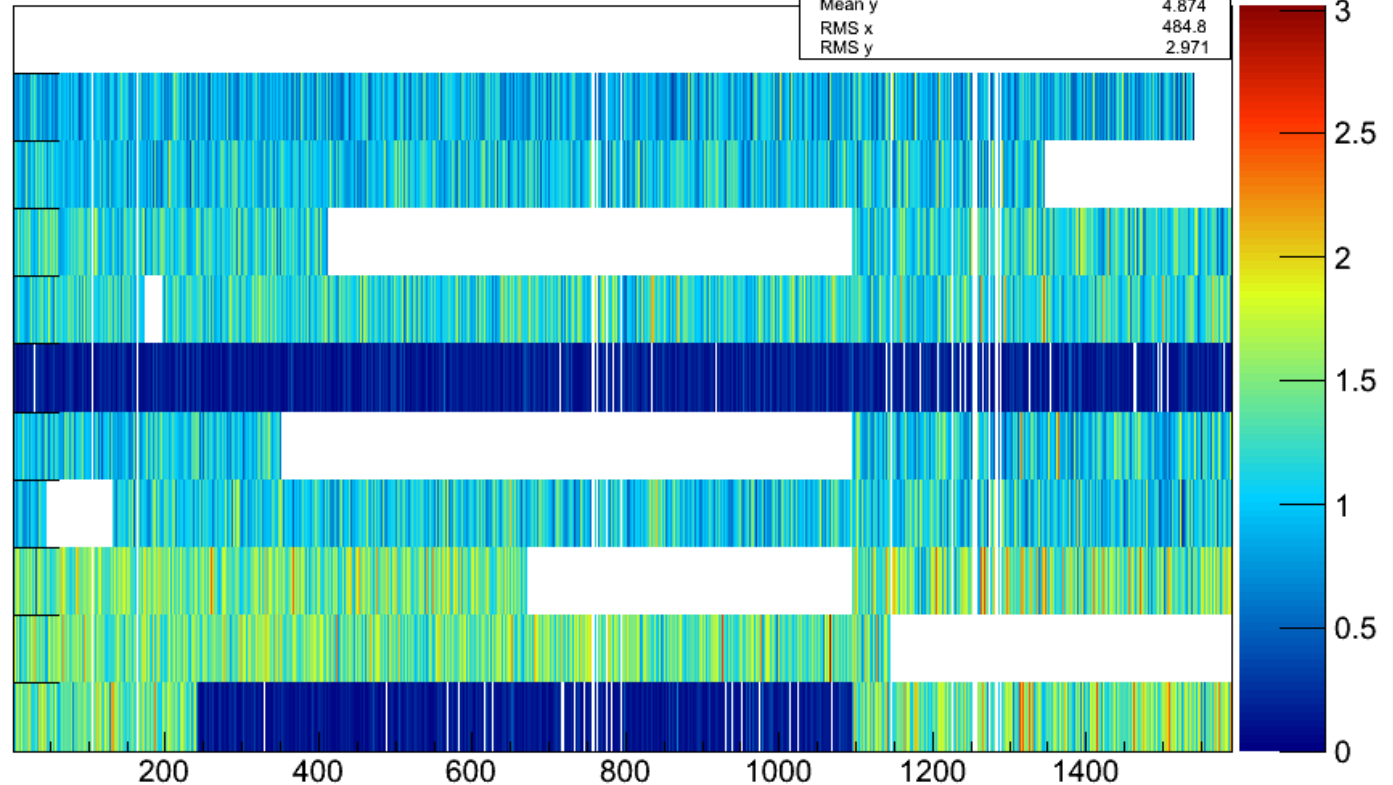
BPix\_BmO\_SEC1\_LYR2\_LDR2F\_MOD3\_ROC12

BPix\_BmO\_SEC1\_LYR2\_LDR2F\_MOD3\_ROC5

BPix\_Bpl\_SEC2\_LYR2\_LDR3F\_MOD1\_ROC7

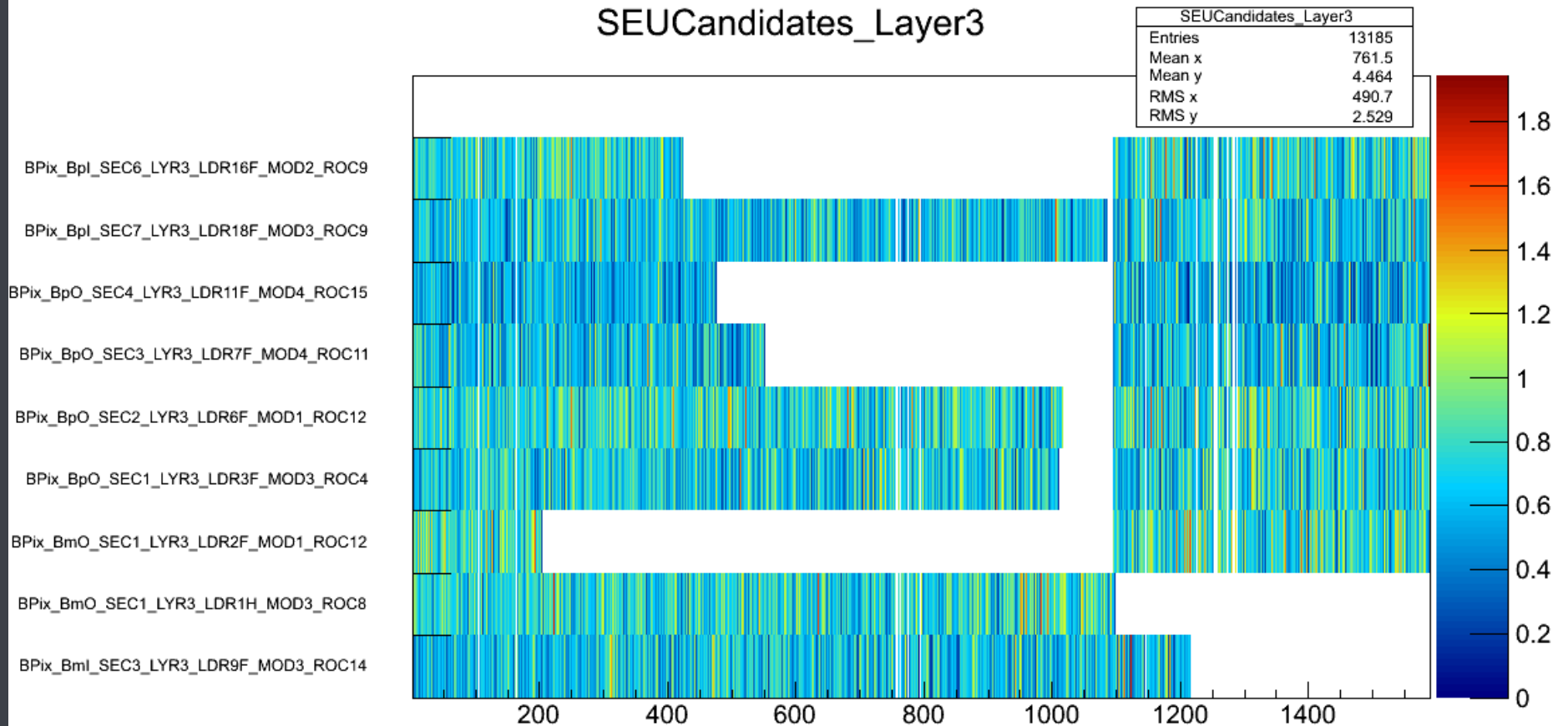
BPix\_Bpl\_SEC2\_LYR2\_LDR3F\_MOD1\_ROC5

BPix\_Bml\_SEC2\_LYR2\_LDR4F\_MOD1\_ROC15



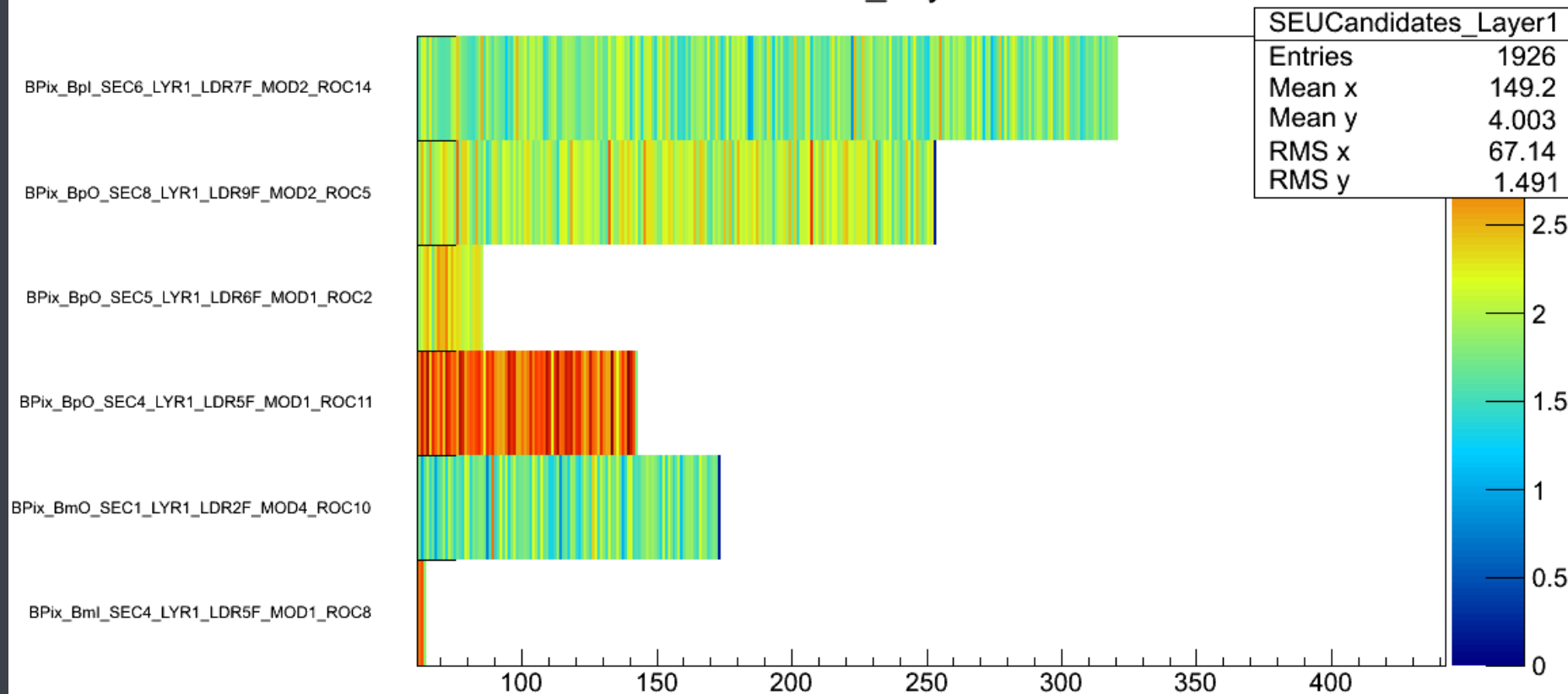
# Run 195950 – Layer3

SEUCandidates\_Layer3



# Run 196531 - Layer1

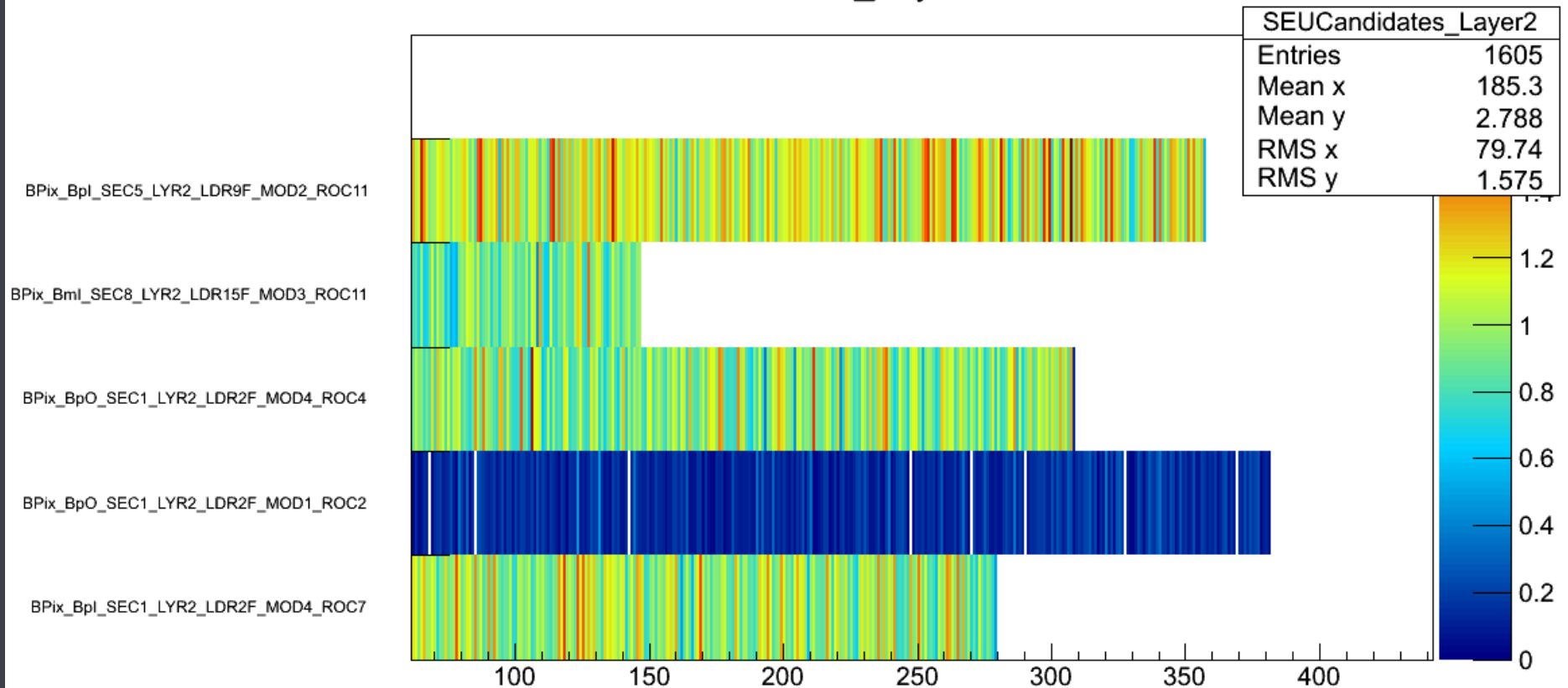
SEUCandidates\_Layer1





# Run 196531 – Layer2

SEUCandidates\_Layer2



# Run 196531 – Layer3

SEUCandidates\_Layer3

