ATO-614 dEdx Optimization

Marian Ivanov, Jens Wiechula,
Tuba Gündem, Marian Ivanov (Jr)

Links to the dashboard and the jupyter notebook:
https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4455805/simulation.html

https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4455806/truncation.html

https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4455807/interactiveToySimul.ipynb

Outlook

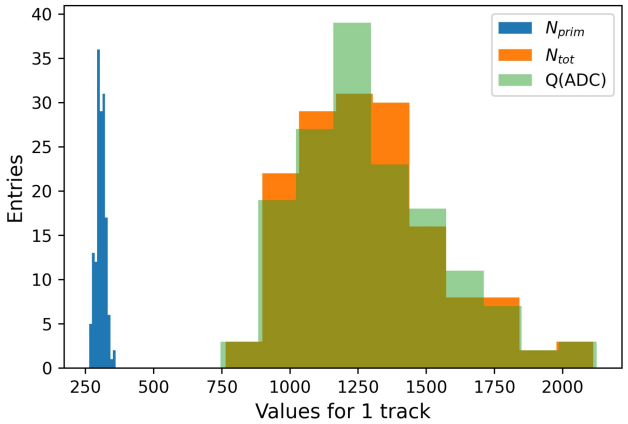
- Toy simulation
  - Simulating total charge (Q) with various dependencies
  - Investigating the simulation with RootInteractive
- Machine learning (ML) and analytical models
  - Using ML and analytical models to fit and predict the data
  - Functional composition for model validation in multi dimension
  - First results using simple Landau simulation will be presented
- Optimization
  - Obtaining optimal combined dEdx estimator for different pad regions of the TPC
- Data
  - Testing on real data

Microscopic Simulation

- Sequence of processes p0 → p1 → p2 → ... → pn

Primary ionization	Total ionization	GEM transparency	Gas gain fluctuations
N_{prim} : Number of primary electrons per cm - Poisson	N_{tot}: N_{prim} + number of secondary electrons (power law) with an effective range cut ($N_{SecSatur}$)	transGEM: 50% to 100% (random)	Q: total ionization + GEM transparency and gas gain fluctuations (exponential)

- Parameters of the processes to be calibrated
- Processed to be added
  - Correlations
    - common mode
    - ion tail
    - multiplicity
  - Calibration/Reconstruction



Microscopic Simulation

- Distributions:
  - N_{prim}, N_{tot}, Q
- Variables:
  - $dN_{prim}dx$ Mean number of primary electrons per cm
  - dN_{prim} Mean numbers of primary electrons over pad rows
  - $Sat_{on/off}$ ADC saturation at Q=1023 (10 bits ADC)
  - Region 0-IROC, 1-OROC1, 2-OROC2, 3-OROC3
  - Pad length 0.75, 1, 1.2, 1.5
  - transGEM 50% to 100%
- Statistics:
  - mean, median, std of
    - N_{prim}, N_{tot}, Q
- Data is simulated using ROOT::RDataFrame and investigated with an interactive dashboard
  - ROOT::RDataFrame → awkward → pandas → RootInteractive → Dashboard

ROOT::RDataFrame

- Creating an RDataFrame:

```
ROOT::RDataFrame df(nTracks);
auto rdf = df.Define("dNprimdx", "10 + gRandom->Rndm() * 500")
             .Define("SatOn", "gRandom->Rndm()<0.5")
             .Define("nSecSatur", "pow(10,2+gRandom->Rndm()*4)")
             .Define("TransGEM", "0.5+gRandom->Rndm()*0.5")
             .Define("region", "getRegionVector(152)")
             .Define("padLength", "getPadLengthVector(region)")
             .Define("Vector", "getVectorWithGasGain(dNprimdx,padLength,SatOn,nSecSatur,TransGEM)")
             .Define("nPrimVector", [](std::tuple<ROOT::VecOps::RVec<float>,ROOT::VecOps::RVec<float>,ROOT::VecOps::RVec<float>> vec) { return std::get<0>(vec); },{"Vector"})
             .Define("nTotVector", [](std::tuple<ROOT::VecOps::RVec<float>,ROOT::VecOps::RVec<float>,ROOT::VecOps::RVec<float>> vec) { return std::get<1>(vec); },{"Vector"})
             .Define("qVector", [](std::tuple<ROOT::VecOps::RVec<float>,ROOT::VecOps::RVec<float>,ROOT::VecOps::RVec<float>> vec) { return std::get<2>(vec); },{"Vector"})
             .Define("nPrimStd", "StdDev(nPrimVector)")
             .Define("nTotStd", "StdDev(nTotVector)")
             .Define("qStd", "StdDev(qVector)")
             .Define("nPrimMean", "Mean(nPrimVector)")
             .Define("nTotMean", "Mean(nTotVector)")
             .Define("qMean", "Mean(qVector)")
```

- Displaying some of the columns:

Row	nPrimVector	nTotVector	qVector	dNprimdx	SatOn	nSecSatur	TransGEM	region	...
0	37.3333f	242.667f	288.534f	34.799299	false	112.94509	0.80166458	0	...
	49.3333f	285.333f	314.786f					0	...
	33.3333f	84.0000f	86.0357f					0	...
	38.6667f	236.000f	243.420f					0	...
	22.6667f	69.3333f	82.4921f					0	...
	38.6667f	298.667f	310.031f					0	...
	34.6667f	97.3333f	96.9462f					0	...
	28.0000f	134.667f	123.905f					0	...
	41.3333f	208.000f	190.683f					0	...
	42.6667f	146.667f	152.671f					0	...
	...	...	...					...	...

Awkward - Pandas

- RDataFrame → Awkward array → Pandas

```
import awkward._v2 as ak
array = ak.from_rdataframe(
      rdf,
      columns=(
         "qVector",
         "nPrimVector",
         "nTotVector",
         "dNprimdx",
         "padLength",
         "region",
         "SatOn",
         "TransGEM",
         "nSecSatur",
         "nPrimMean",
         "nTotMean",
         "qMean",
         "qMedian",
         "nPrimStd",
         "nTotStd",
         "qStd",
             ),
   )
df =ak.to_dataframe(array)
```

- Pandas dataframe (df): Input for RootInteractive

RootInteractive Flow

- Input
  - Obtaining input data
  - Creating derived variables
    - On server - creating new column e.g:
      - df[C] = df[A] + df[B]
      - df[CRF] = rf.predict(df[[A,B]])
    - Function declaration evaluated on client in optional alias array
      - nTotStdRel = nTotStd/nTotMean
      - nTotStdRelExp = (1/sqrt(nPrimMean))*sqrt(1+sigmaNRel**2)
  - Acquiring RootInteractive properties from users to generate the html output
    - Mandatory
    - Optional

- Output (html)
  - Users can define as many
    - Figure Tabs
    - Figures
    - Widget Tabs
    - Widgets
  - Runtime changeable widgets and re-rendering the figures

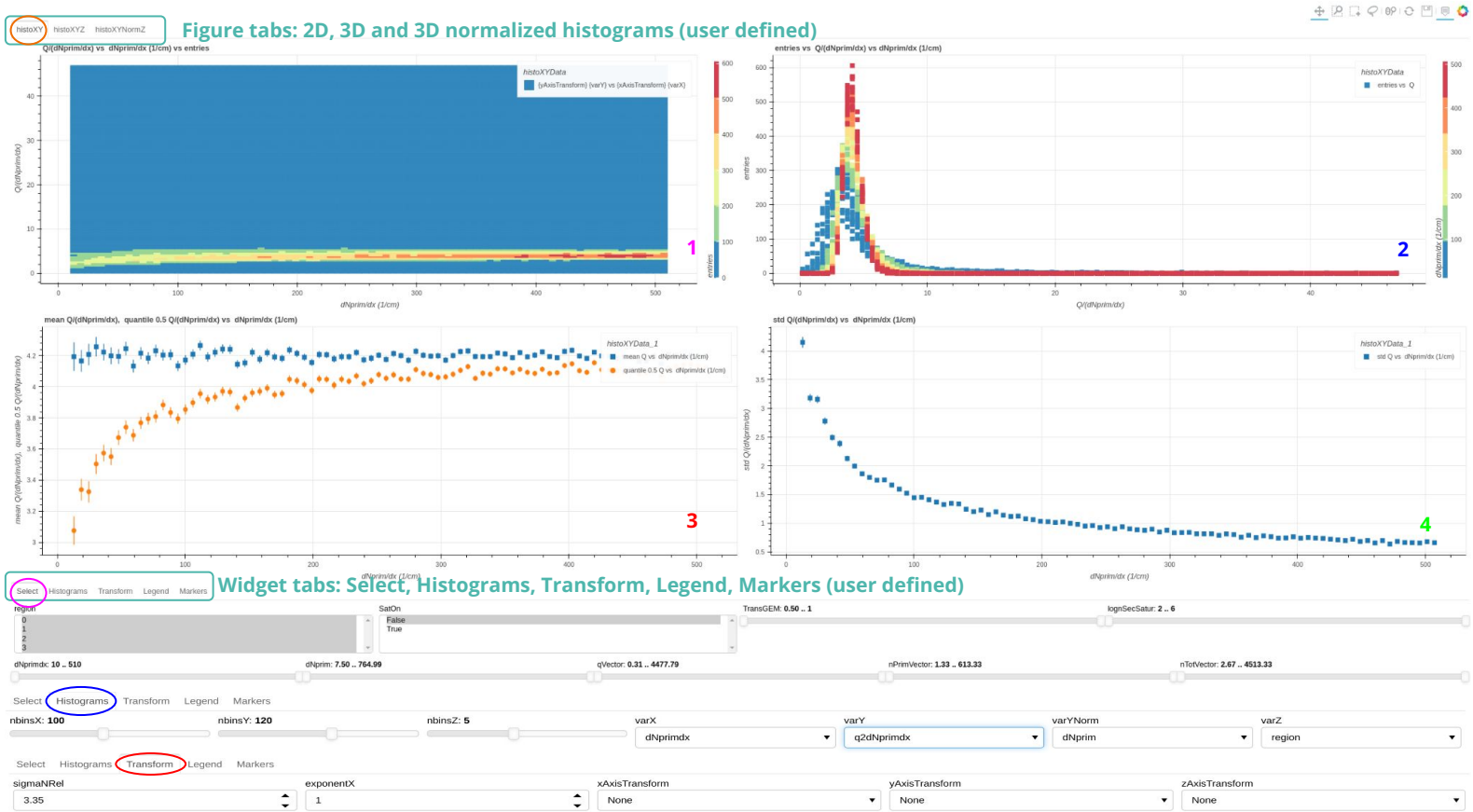
RootInteractive

- User defined RootInteractive properties are required to get the html output:
  - Alias array for calculated variable definition
    - [("q2NPrim", "qVector/nPrimVector"), ...]
  - Variable array
    - ["qVector", "nPrimVector", ...]
  - Parameter array
    - [{"name": "varX", "value":"dNprimdx", "options":variables}, ...]
  - Widget parameters
    - [["select", ["varX"], {"name": "varX"}], ...]
  - Widget layout dictionary
    - {"Histograms": [["varX", ...], {'sizing_mode': 'scale_width'}], ...}
  - Histogram array
    - [{"name": "histoXYData", "variables": ["varX","varY"], "nbins":["nbinsX","nbinsY"]}, ...]
  - Figure array
    - [["bin_center_1"], ["bin_count"], {"source":"histoXYData", "colorZvar": "bin_center_0"}], ...]
  - Figure layout dictionary
    - {"histoXY":[[0,1], [2,3], {"plot_height":220}], ...}

- Links to the dashboard and the jupyter notebook:
  - https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html
  - https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410582/interactiveToySimulationGG.ipynb

Dashboard

https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html



Example figure tab:

1- 3D histogram with x,y,entries

2- 3D histogram with entries,x,y

3- 2D histogram with x,mean and median of y

4- 2D histogram with x,std of y

Example widget tabs:

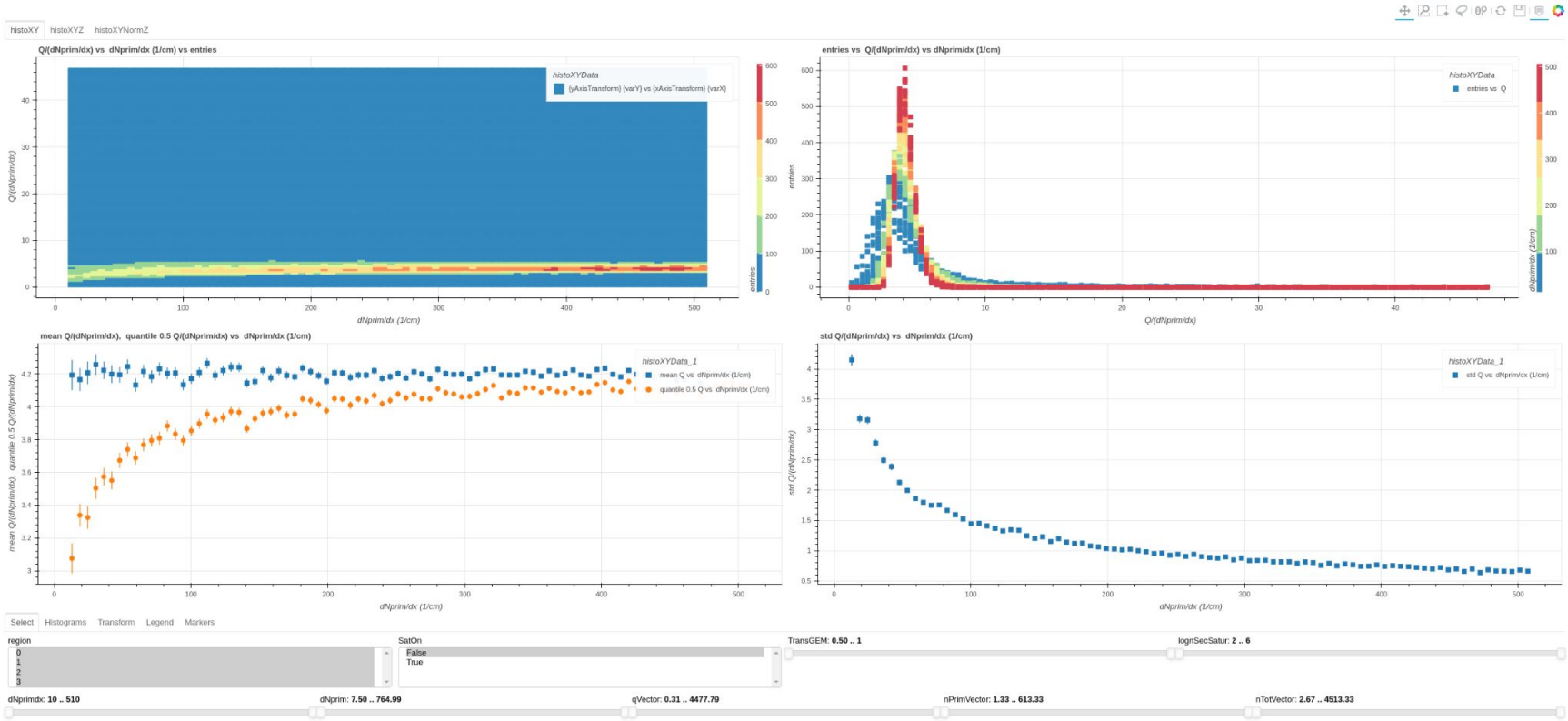
Selection: multi-selects, ranges

Histograms: number of bins for x,y,z, changeable axis values for x,y,z

Transform: changeable axis transformation

Observations

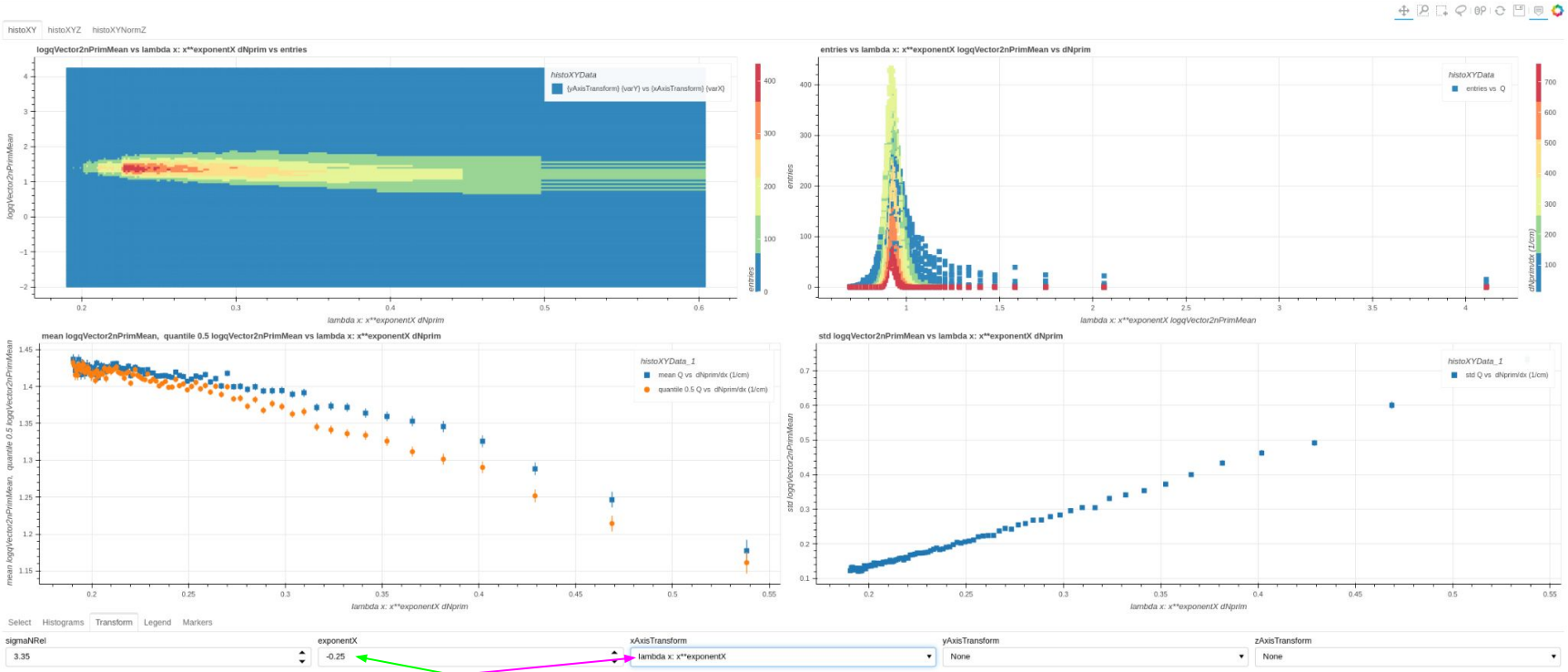
https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html



The mean of the Q/(dN$_{prim}$dx) as a function of dN$_{prim}$dx is independent of the dN$_{prim}$dx
The median of the Q/(dN$_{prim}$dx) as a function of dN$_{prim}$dx is dependent on the dN$_{prim}$dx

Observations

https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html



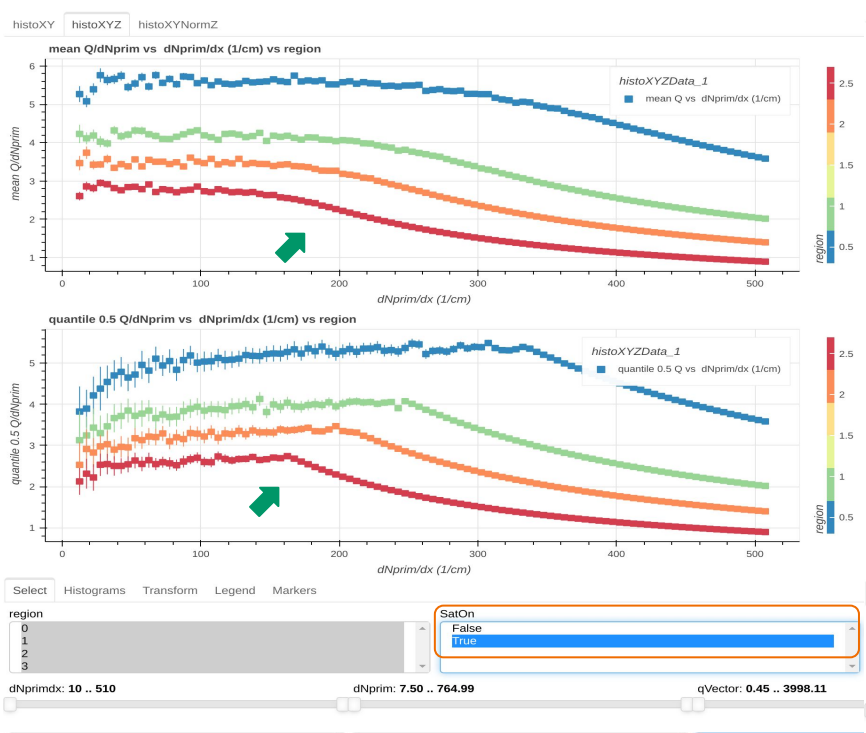
Axis transformation (x^exponentX) is used and exponentX can be determined interactively

The std of the log(Q/(<N_{prim}>)) depends almost linearly on the $dN_{prim}^{-0.25}$

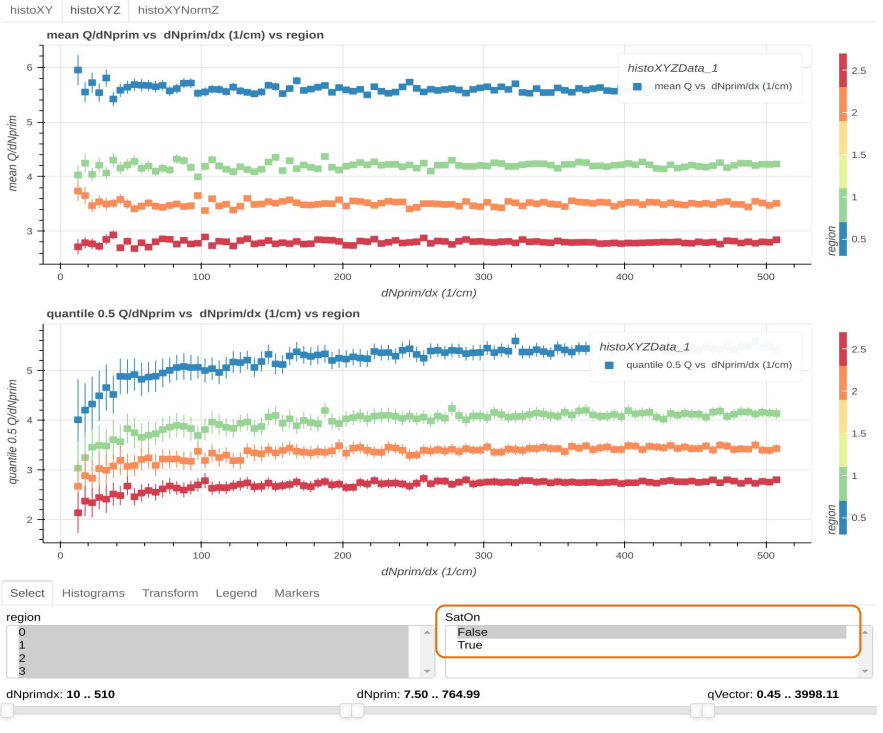
Observations

https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html

With ADC saturation cut



Without ADC saturation cut

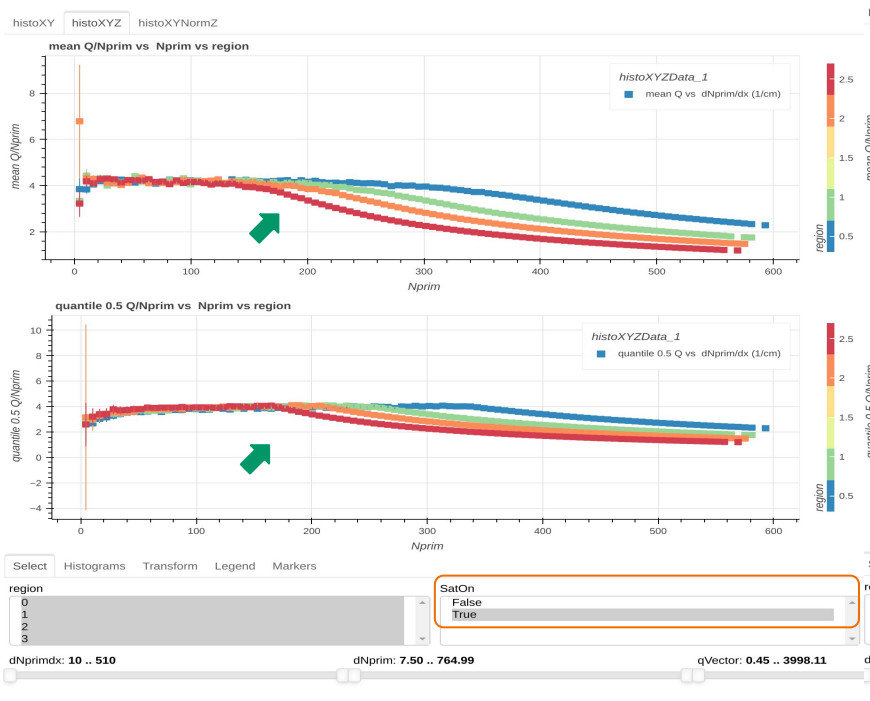


Q/dN$_{prim}$ as a function of dN$_{prim}$dx depends on the TPC pad region (color code)
The mean and median differs all the time especially at region determined by the ADC saturation cut

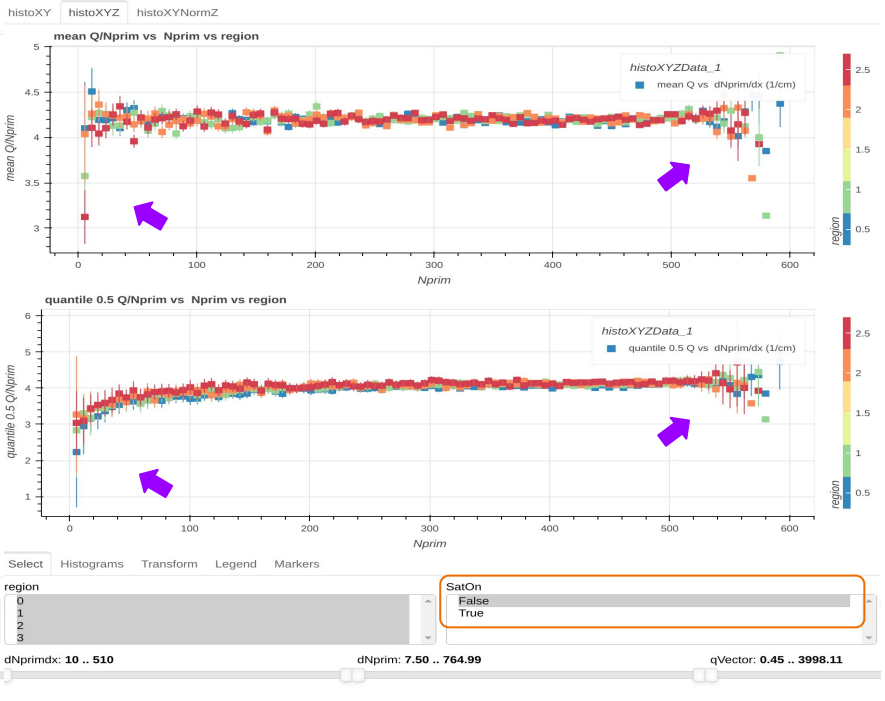
Observations

https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html

With ADC saturation cut



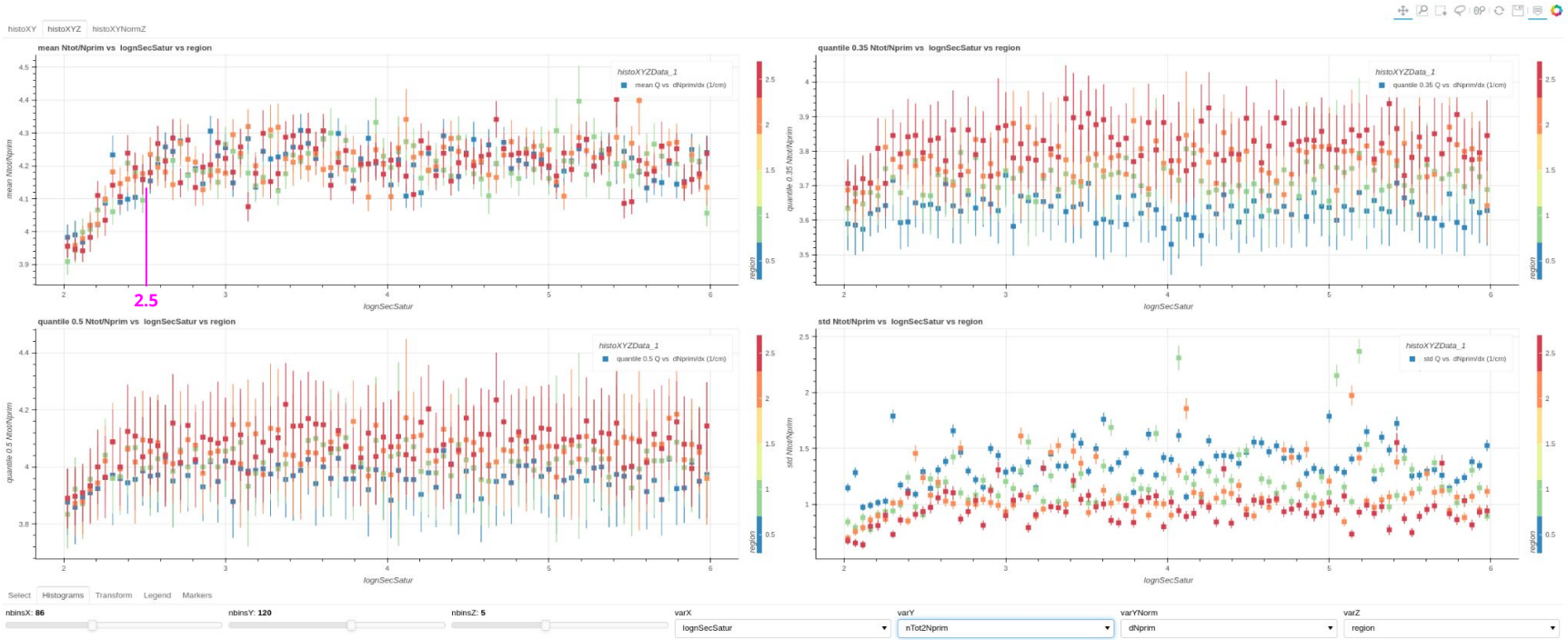
Without ADC saturation cut



Q/N_{prim} as a function of N_{prim} depends on the TPC pad region (color code) especially in ADC saturation cut region
The mean and median differs at regions determined by the ADC saturation cut and by the low and high Nprim values

Observations

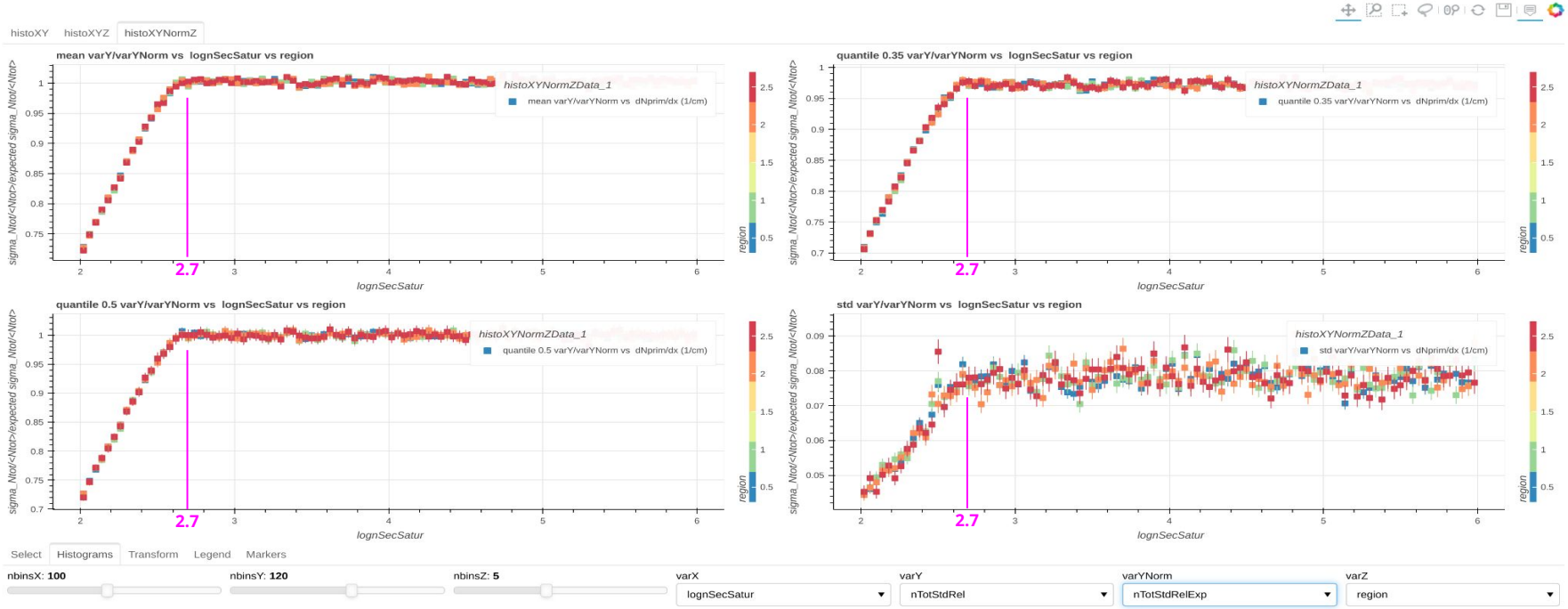
https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html



The mean of the N_{tot}/N_{prim} as a function of log10($N_{SecSatur}$) is sensitive to the electron range up to $10^{2.5}$ (electrons)
The median, quantile and std are not that sensitive

Observations

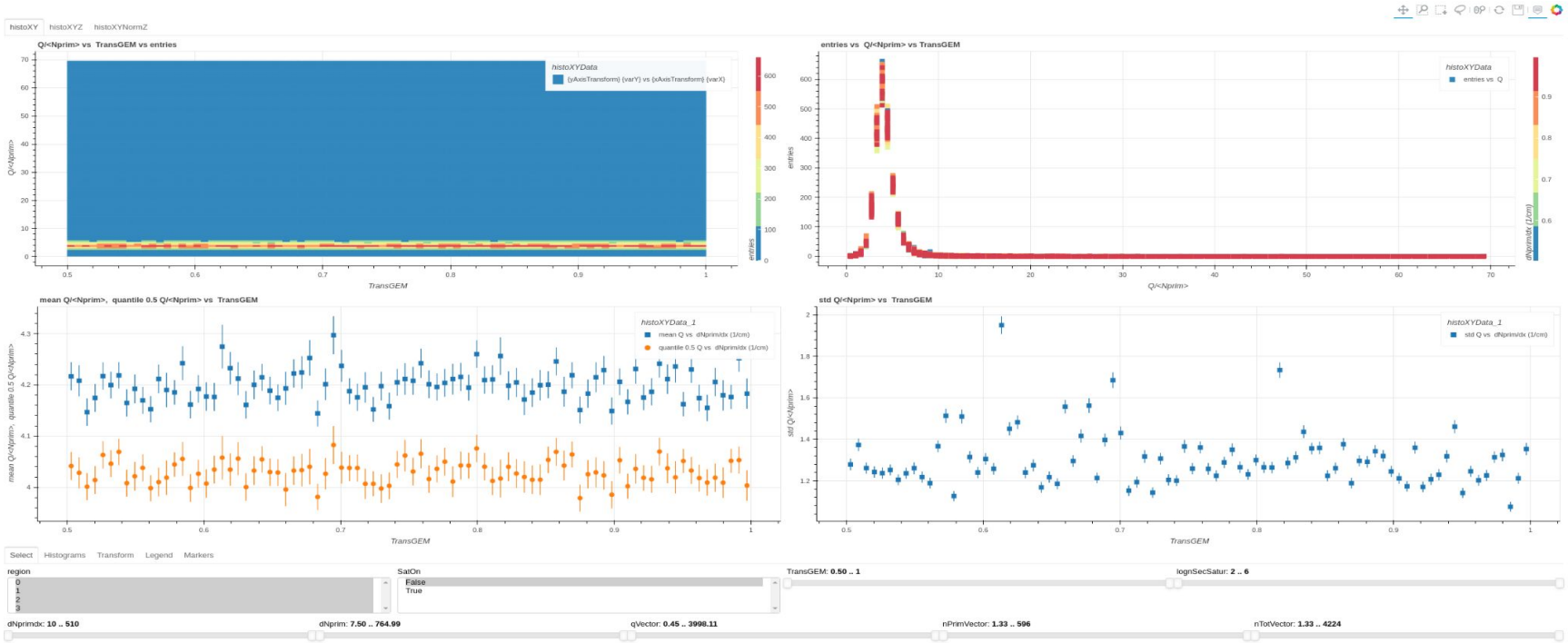
https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html



The mean, median, quantile and std of the $\sigma_{Ntot}/<N_{tot}>$ normalized to expected $\sigma_{Ntot}/<N_{tot}>$ as a function of log10($N_{SecSatur}$) is sensitive to the electron range up to $10^{2.7}$ (electrons)

Observations

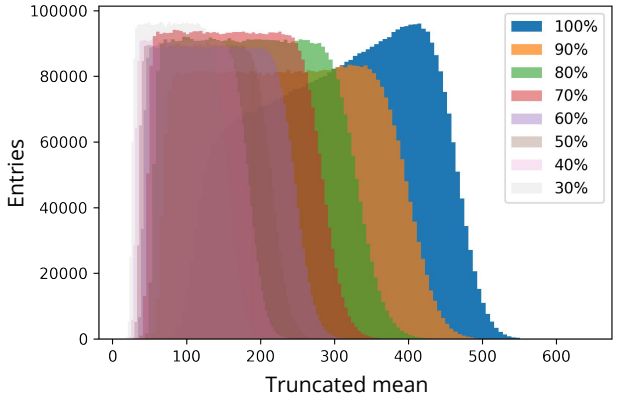
https://indico.cern.ch/event/1221198/contributions/5156214/attachments/2558723/4410043/dEdxWithGasGain.html



The mean, median and std of the Q/<N_{prim}> as functions of GEM transparency are not sensitive to GEM transparency

Simple Landau Simulation - First Results

- 5000000 tracks with Q values are simulated
  - Using the Landau distribution
- Statistics:
  - mean, median, rms of Q
  - mean, median, rms of log(Q)
- dEdx:
  - truncation of Q with 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%



ML - Random Forest

- Relative resolution results from different estimators using the Landau toy simulation
  - Q mean 0.1031
  - Q median 0.0848
  - Q std 0.2393
  - log(Q) mean 0.0808
  - log(Q) median 0.0848
  - log(Q) std 0.4351
  - Truncation
    - 40% 0.0842
    - 50% 0.0783
    - 60% 0.0758
    - 70% 0.0756
    - 80% 0.0786
    - 90% 0.0884
  - Q mean, median, std 0.0777
  - log(Q) mean, median, std 0.0773
  - Truncation 50-60-70 % 0.0751
  - Truncation 50-60-70-80 % 0.0749
  - Truncation 50-60-70-80-90 % 0.0749
  - Truncation 40-50-60-70-80-90 % 0.0749

logarithmic Q transformation gives relative resolution that is comparable with the truncated mean

optimal truncation between 60%-70%

minor improvement of performance using set of truncation

Thank you
for your attention