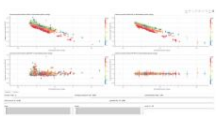
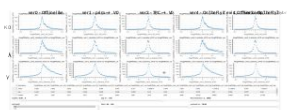
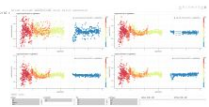
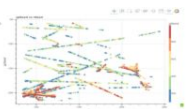
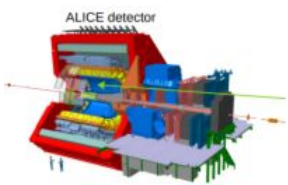
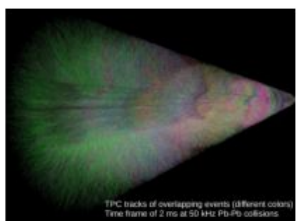
UNIVERZITA KOMENSKÉHO V BRATISLAVE

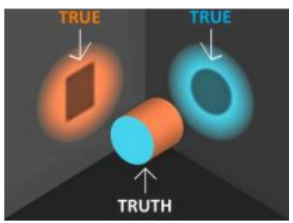
GSI

RootInteractive expert tool for multidimensional statistical analysis, machine learning and analytical model validation.

Marian Ivanov (GSI Darmstadt), Marian Ivanov jr (UK Bratislava)



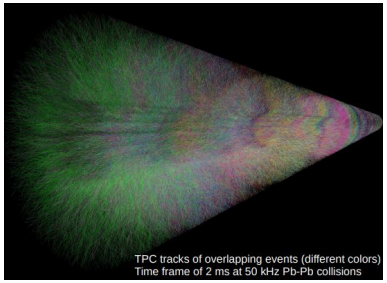


https://github.com/miranov25/RootInteractive https://arxiv.org/abs/2403.19330 https://github.com/miranov25/pyhep2024-rootinteractive

Alice Run 3 - goals and challenges

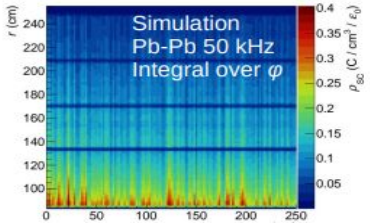
Record large pp and Pb-Pb minimum bias sample

- Continuous readout at 50 kHz Pb-Pb collisions and 500kHz-1Mhz pp collisions
- Unknown collision time
- Events overlapping in TPC → substantial higher occupancy (~5 PbPb collisions, 100 pp collisions)



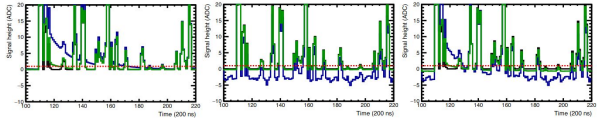
Tracking challenge: space charge in TPC detector distorting trajectories

- Non-uniform space-charge distorting E field
- Large space point distortions O(5 cm) and Distortion fluctuations O(5 %) ~ 0.2 cm
- To be calibrated to σ ~100 μm with space granularity O(10^6) in space O(1-5 ms) in time



PID challenge: Significant baseline bias and fluctuation

- Online digital signal processing to recover baseline (in FPGA)
- To be corrected below internal noise level

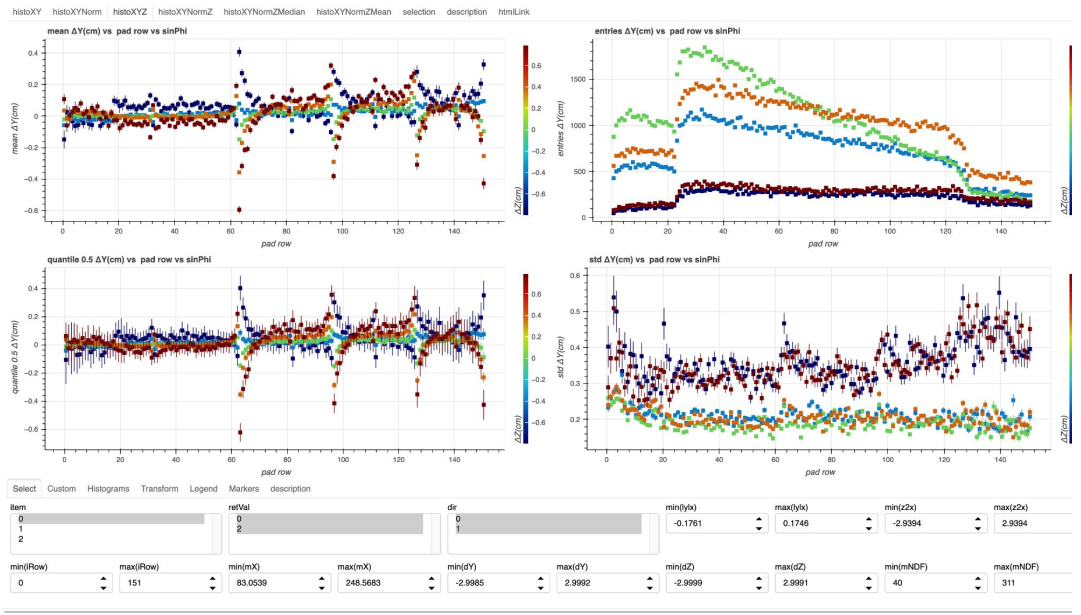


High interaction rates, pile-ups, and distortions demand advanced data analysis methods. Requires experts and highly customizable tools to navigate and interpret data in multidimensional parametric spaces.

RootInteractive project

Seeing is believing

Querying/Iterative Interacting/predicting is understanding



Reconstruction/distortion monitoring example - 10^7 points x 50 attributes (space points,track, MC predictions)

- https://github.com/miranov25/RootInteractive#readme

Multi-Dimensional Interactive Data Analysis and Aggregation:

- **Machine Learning, Fitting, and Histogramming:** Comprehensive data processing techniques implemented on servers, such as Jupyter notebooks and Python scripts.
- **Client-Side Data Aggregation:** Supports extensive tabular datasets ranging from 10^6 to 10^7 rows, 10 to 300 columns, and up to 10^8 data entries (rows x columns) directly within a browser environment.

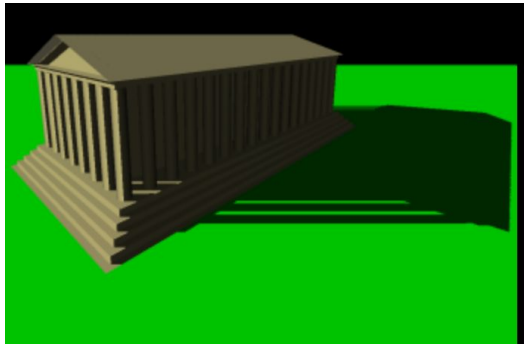
RootInteractive - current ALICE expert projects

- Run3 alignment & space point distortion calibration
- Run3 digital signal processing parameter optimization
- Run2, Run3 track reconstruction optimization, validation
- MC/data mapping & TPC data volume studies
- Run2, (Run3) expert differential QA/QC , performance parameterization, performance web pages
- Expert data representative sampling/skimming
- PID calibration/validation and dEdx optimization

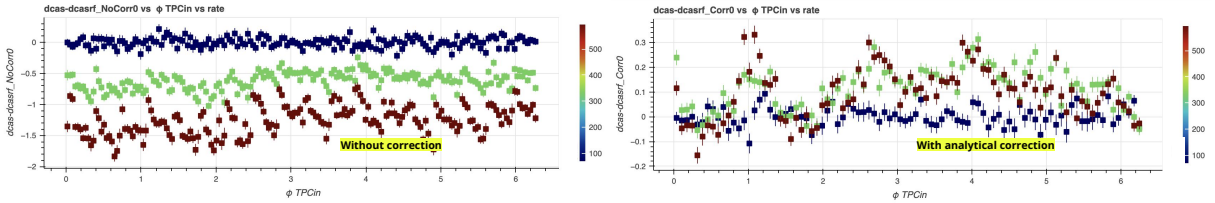
- Run3 (4D) reconstruction development - trackCombinator - V0, Cascade, Kink, cosmic finder
- Fast simulation - fastMCkalman for detector and reconstruction optimization (Run3,Alice3)
- High dEdx,spallation tracking (collaboration with DUNE experiment)
- Magnetic monopole reconstruction
- Particle production as function of event properties

- Particle production - MC generators parameter scan

Multi-dimensional analysis vs shadow projections



Track DCA bias due space charge distortion contribution before and after correction
Reference-ML prediction(s) at low rate without Space charge distortion

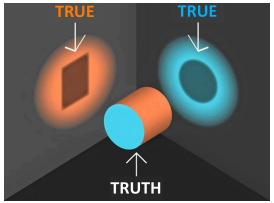


$$\sigma_{\vec{A}\ominus\vec{A}_{ref}} \leq \sigma_{\vec{A}}(+)\sigma_{\vec{A}_{ref}}$$

Object and reference objects (models/reference models, MC/Data,Data/ref. data), should be compared optimally in the full relevant multi-dimensional space.

- Shadow projection → Assumptions, imagination and rhetorical art in describing data needed
- Comparison statements to be based on invariants or on normalized data - e.g. the difference between the object and the reference object
  - After projection impossible
- In many typical cases variance $\sigma_{A\text{-}Aref}$ is very often smaller by orders of magnitude
  - For example, the rms value of the difference between ionic currents and scaled average values can be used as an alarm criterion. We cannot use the ion current itself
- Differential approach - possibility to decompose and understand the data e.g. distortion due space charge, and alignment in figure above

RootInteractive - a general purpose tool for multi-dimensional statistical analysis



Challenge: Oversimplification at the analysis level can lead to complex or incorrect explanations.

Objective: Equip users with tools to address multidimensional challenges, streamlining the data analysis process:

- **Fit and Visualize N-dimensional Functions:** Incorporate uncertainties and biases directly within the visualization.
- **Validate Assumptions:** Numerically evaluate approximations and compare models differentially to ensure robust conclusions.
- **Functional Composition:** Facilitate the integration of non-parametric and parametric functions, enhancing error propagation analysis.
- **Rapid Feedback:** Enable very fast feedback—from seconds instead of weeks—to foster interactive expert discussions from day one.
- **Optimization:** Conduct multidimensional parametric optimization efficiently.
- **Configurable Visualization:** Manage both unbinned and binned data with ease, utilizing interactive multidimensional histogramming and projection to extract derived aggregate information on both server (Python/C++) and client (JavaScript) sides.
- **Standalone Client Application:** Deliver analysis capabilities in a standalone HTML document, eliminating the need for additional software installation, perfect for sharing and collaboration.

A detailed differential understanding of the detector system, MC and reconstruction/calibration performance is a prerequisite for the successful application of Machine learning in physical analysis

Consideration: symmetries, alarms and invariants

Aggregation/projections of normalized data e.g. (data-model), (MC-Data), (data-symmetry) in multiple dimensions :

- RMS spread is much smaller
- Alarms/Outlier tagging with statistical significance - e.g. (data-model) > N σ , or likelihood
- **Invariance/symmetries**
  - in-variance in time (using e.g. reference/average run), in-variance in space (e.g. rotation, mirror symmetry)
  - B field symmetry
  - data - non parametric/parametric analytical model
  - smoothness resp. local smoothness

In RootInteractive supported mostly comparing data with reference “symmetric regression” and “template support” automatic comparison to reference data

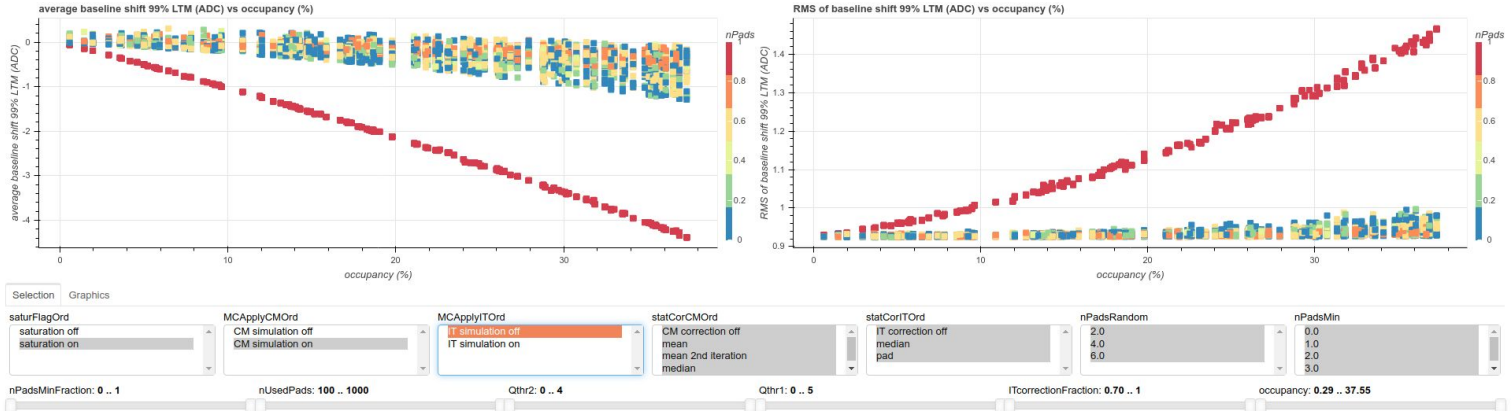
Multidimensional parameter optimization example - ALICE digital signal processing

Digital signal processing (13 parameters in example) needed for particle identification and data volume optimization. **O(200000) parameter settings simulated/generated on server**

- parameters: effects (On/Off), algorithm (different version), parameters of individual algorithms

Simulation and visualization/aggregation (groupby+RootInteractive) done by bachelor student, fully solving optimization problems of DSP

- Dashboard to answer “all questions”, FEEDBACK time for follow up questions O(seconds)
- Standalone dashboards, others could reproduce result based on the instruction in presentation, movie instruction
- **Interactive expert use-case discussion within ONE meeting. DSP understand and solved. Project DONE.**



Presentation, notebook, interactive dashboard and movie in RootInteractive tutorial:

- https://indico.cern.ch/event/1135398/contributions/4764024/subcontributions/370740/attachments/2402507/4114272/CMITSimulGEMTPC_RootInteractiveTutorial10032022.pdf
- https://gitlab.cern.ch/alice-tpc-offline/alice-tpc-notes/-/blob/master/JIRA/ATO-559/parameterScan.ipynb
- https://indico.cern.ch/event/1073383/contributions/4588170/attachments/2334149/3986420/simulScan_02112021.html
- https://indico.cern.ch/event/1135398/contributions/4764024/subcontributions/370740/attachments/2402507/4109039/CMITSimulationsGEMTPC.mp4

Machine learning in RootInteractive - differential validation of MC/data and ML models

Using external models:

- E.g comparing the U-Net for the distortion correction with simple data driven Machine learning using Random Forest
- Parameter optimization in respect to different cost functions

RootInternactive extensions wrappers to scikit-learn and xgboost

- Fast approximation of functions and local PDFs

Interactive validation in RootInteractive on client O(10^6-10^7) points

- Unbinned predict
- Aggregated information for further postaggragation
  - Local mean, median, STD - unbinned predict
  - Local kernel regression parameters -aggregated information on the mesh
    - Usually statistical properties of predict- value, resp. Mash of 1D histograms
- Generalized kernel linear regression on client (ND groupby+rolling+kernel)
- Predict on client (wasm+ONNX) in queue

Machine learning - derived variables - RF regression - per channel QA example

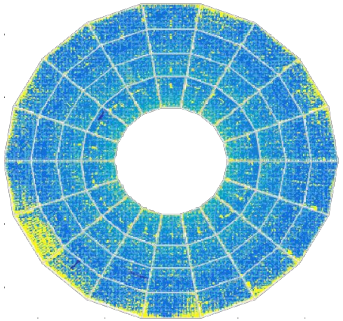
```
statDictionary={"mean":None,"median":None, "std":None}

varListG=["lx","ly","GainMap","A_Side"]
varListLocal=["lx","ly","GainMap","roc"]
vars=[
   "NClusters_Clusters_Mean",'NClusters_Digits_Mean',
   'QMax_Clusters_Mean', 'QMax_Digits_Mean',
   'IDC0_Mean','SAC0_Mean'
]
statOut=miErrPDF.predictStat(dfK0[variableX],statDictionary)
```

Per channel QA and example derived QA variables for NClusters_Clusters:

- **NClusters_Clusters_Mean**
- NClusters_Clusters_MeanRF0,
- NClusters_Clusters_MeanRF0,
- NClusters_Clusters_MeanRFL,
- NClusters_Clusters_MeanRFL_Med
- NClusters_Clusters_MeanRFL_Std

TPC noise map



Defining models:

- varying parameter of models, input variables and local statistics

Global (varListG) and **local regression (varListLocal)** extracting for basic calibration and QA properties of ALICE TPC calibration and QA variables

- global **φ symmetric** model, local model **without φ symmetry**
- **Automatic alarms - data "out of range "|data-prediction|<nσ" without "reason" (other calibration, masking known problems)**

Robust local statistics - median and local std estimator for the outlier tagging and PDF description

RootInteractive/Multi-Interactive project preparation and presentation

Expert data preparation:

- Agreement on data to collect and aggregate
- Data sources
- Variables to import - asking questions
- Symmetries, invariances and possible alarms
- Pre-aggregation
- Data sampling
- Machine learning models
- Underlying Analytical models if exist
- Re-iteration

Data presentation:

- **Agenda: presentation, notebook, dashboard+ (optional)movie**
- Goal
- **Data preparation explained**
- **Variables description**
- Observation highlights with snapshot from dashboard
- **Domain experts, participants in the meeting should be able to participate in decisions, resp. be able to interact with dashboard data based on description in presentation**

The data is presented in a multidimensional way. The aim is to answer all questions within one meeting/session. If the information is not sufficient, new data sources to be agreed on.

RootInteractive interactive dashboard declarations

User defined RootInteractive properties are required to get the html output (explained in next slides)

- **Alias array for derived variable/function definition - e.g defining status bitmask**
  - aliasArray=[("IDC0_OK","(0x2*(abs(IDC0_MeanRF0_LRatio)<sigmaRFCut0))|(0x4*(abs(IDC0_MeanRFL_LRatio)<sigmaRFCutL))"),..]
- **Variable array**
- **Parameter array - to control parameterized functions, selection and variable selection for ND histograms**
- **Widget description array**
- **Widget layout dictionary**
- **Histogram array**
- **Figure array**
- **Figure layout dictionary**

Usually started from a template configuration e.g.:

aliasArray, variables, parameterArray, widgetParams, widgetLayoutDesc, histoArray, figureArray, figureLayoutDesc = getDefaultVarsDiff()

Functions on client - derived variables and functional composition

Predefined parametric javascript function

```
# here we can define derived variables -  to define some invariances eg abs(XX_Mean/XXXMedain)<
aliasArray=[
#    ("","dNprimdx*padLength"),     # ionization over pad
    ("Unit","1+roc*0"),
    ("phi","arctan2(gy,gx)"),
    ("QMax_Clusters_OK","(0x1*(NClusters_Clusters_Mean>minEntries))|(0x2*(abs(QMax_Clusters_MeanRF0_LRatio)<sigmaRFCut0))|(0x4*(abs(QMax_Clusters_MeanRFL_LRatio)<sigmaRFCutL))"),
    ("QMax_Digits_OK","(0x1*(NClusters_Digits_Mean>minEntries))|(0x2*(abs(QMax_Digits_MeanRF0_LRatio)<sigmaRFCut0))|(0x4*(abs(QMax_Digits_MeanRFL_LRatio)<sigmaRFCutL))"),
    ("SAC0_OK","(0x2*(abs(SAC0_MeanRF0_LRatio)<sigmaRFCut0))|(0x4*(abs(SAC0_MeanRFL_LRatio)<sigmaRFCutL))"),
    ("IDC0_OK","(0x2*(abs(IDC0_MeanRF0_LRatio)<sigmaRFCut0))|(0x4*(abs(IDC0_MeanRFL_LRatio)<sigmaRFCutL))"),
    #("IDC0_OK","1+(abs(IDC0_RMS/IDC0_Mean)<0.5)"),
    ("IDC0_MeanOK","0x1*(IDC0_RMS<5) |0x2*(IDC0_MeanLxCut)")
]
```

Anonymous function (used for example in ND histograms as weights or variable)

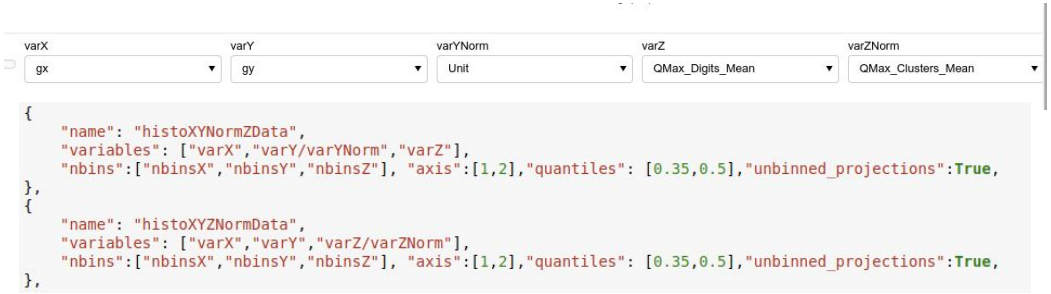
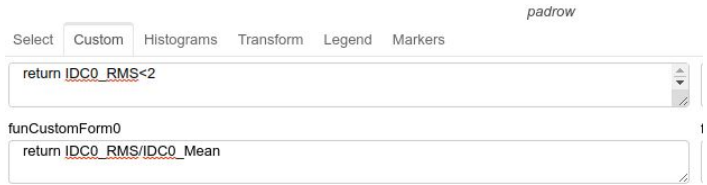


Figure axis transformation

Custom javascript function (javascript function as a text)

```
# defining custom java script function to query  (used later in varaible list)
aliasArray+=[{
        "name": "funCustom0",
        "variables": [i for i in variables if  "ustom" not in i ],
        "func":"funCustomForm0",
    },
    {
        "name": "funCustom1",
        "variables": [i for i in variables if   "ustom" not in i],
        "func":"funCustomForm1",
    },
    {
        "name": "funCustom2",
        "variables": [i for i in variables if   "ustom" not in i],
        "func":"funcCustomForm2",
    },
]
```



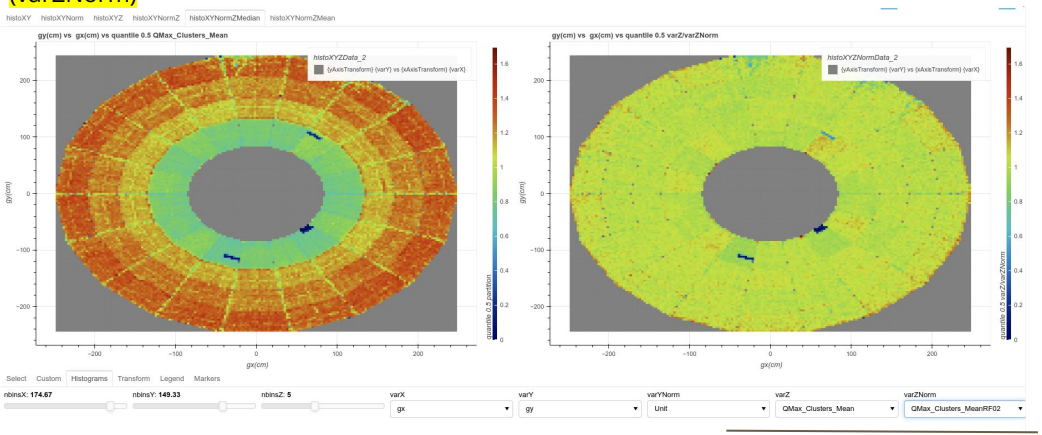
Many different ways to define derived variables and functional composition.
Dependency trees to resolve functional and data source dependencies.

Histogram declaration - calibration QA browser

Set of the 2D, 3D (ND) histograms declared ()

```
histoArray=[
    {
        "name": "histoXYData",
        "variables": ["varX","varY"],
        "nbins":["nbinsX","nbinsY"], "axis":[1],"quantiles": [0.35,0.5],"unbinned_projections":True,
    },
    {
        "name": "histoXYNormData",
        "variables": ["varX","varY/varYNorm"],
        "nbins":["nbinsX","nbinsY"], "axis":[1],"quantiles": [0.35,0.5],"unbinned_projections":True,
    },
    {
        "name": "histoXYZData",
        "variables": ["varX","varY","varZ"],
        "nbins":["nbinsX","nbinsY","nbinsZ"], "axis":[1,2],"quantiles": [0.35,0.5],"unbinned_projections":True,
    },
    {
        "name": "histoXYNormZData",
        "variables": ["varX","varY/varYNorm","varZ"],
        "nbins":["nbinsX","nbinsY","nbinsZ"], "axis":[1,2],"quantiles": [0.35,0.5],"unbinned_projections":True,
    },
    {
        "name": "histoXYZNormData",
        "variables": ["varX","varY","varZ/varZNorm"],
        "nbins":["nbinsX","nbinsY","nbinsZ"], "axis":[1,2],"quantiles": [0.35,0.5],"unbinned_projections":True,
    },
]
```

ALICE TPC QA example mean charge: left - raw values(varZ) , right-normalized to “expectation” (varZNorm)



Anonymous function (used for example in ND histograms as weights or variables)

Parameterized histograms:

- Variables and weights could be any variable from data source (column, derived functions, anonymous function)
  - In the QA/calibration browser variables defined by user selecting (varX,varY, varZ)
  - Binning controlled by parameters (nbinsX, …)
- Derived aggregated data exported as new data source
  - Declaring quantiles and projections
  - Projection could be binned (fast) and unbinned

Customizable N-dimensional histograms and projection. Example:

- X,y median profile of cluster charge map (left) and normalized to phi symmetric RF prediction

RootInteractive dashboard declarations - configurable templates (comparison to reference)

https://arxiv.org/abs/2403.19330

View the Interactive Demo https://github.com/miranov25/pyhep2024-rootinteractive

```
aliasArray, variables, parameterArray, widgetParams,
widgetLayoutDesc, histoArray, figureArray, figureLayoutDesc =
getDefaultVarsDiff(variableList, defaultVariables, weightList,
diffFunctionList)
...
 getDefaultVarsRefWeights(variables=variables)
```

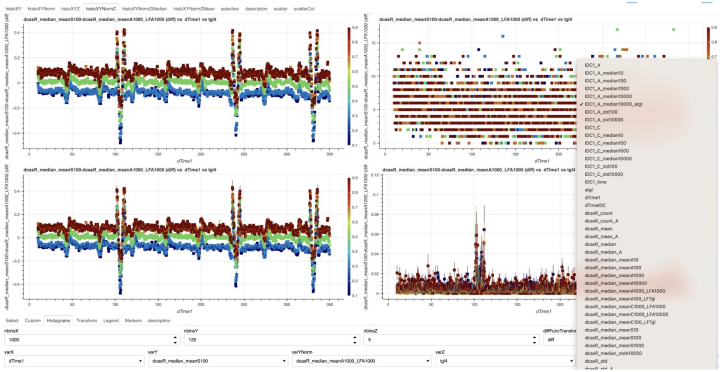


Figure 2. A snapshot of a real use-case l interactive dashboard within ALICE, generated using getDefaultVars. This dashboard is used for comparing ALICE TPC [18] currents (IDC) and TPC tracking bias (<DCA>) time series. The layout includes functions for 1D, 2D, and 3D aggregation, facilitating comparisons between aggregated data and various iterations of machine learning predictions. Additionally, it provides local aggregated statistics. You can find a partial list of the input variables for interactive aggregation on the right side of the snapshot (expanding select for the varZ selection)

Simplification of using interface using set of predefined parameterizable templates to define standard layouts, extending only user defined widget control.
Templates focussed mostly on comparison of data and reference data, resp comparison of their distributions for user defined selection

Ongoing development

WebAssembly Interface - Development Plans

Introducing new functions, transformations, and data sources utilizing WebAssembly:

- **Fast Fourier Transform (FFT):** Leveraging Wasm for efficient FFT computations.
- **Convolution and Deconvolution:**
- **Numpy-like Interface:** Support for binned data operations.
- **Functional Interface:** Facilitates operations with unbinned kernel functions.
- **ONNX Interface:**
- Enables user-defined functional compositions using ML models. This includes calculating derivatives of predictions based on parameter deltas (e.g., discrepancies between model calculations and actual data).
- **Optimization:** Transition of legacy JavaScript numerical codes to Wasm based on performance benchmarks.

These developments are aimed at enhancing computational efficiency and expanding the functionality of our client-side data analysis tools, which currently rely solely on JavaScript.

https://webassembly.org/

RootInteractive - RDataFrame interface - Domain Specific Language (experimental)

```
def makeDefine(name, code, df, cppLibDictionary=None, verbose=3, flag=0x1):
    """
    define new column in RDataFrame using "Python like" syntaxe
    :param name:            - name of the new column
    :param code:            - source code string
    :param df:              - data frame to add new implementation and to define input variable list
    :param cppLibDictionary - dictionary to store generated code and dependencies
    :param verbose:         - verbosity bitmask
    :param flag             - 0x1-makeDefine / do nothing if column exist, 0x2- force bit to redefine if exist 0x4
- test only
    :return:                  new rdf with new column  and append code information to the cppLibDictionary
    """
```

DSL example

```
In [6]: rdf = makeDefine("array2D0_0", "array2D0[0,:]", rdf, cppLib, 3);
INFO:root:   Data type: float, ('f', 32)
INFO:root:array2D0_0
array2D0[0,:]
INFO:root:Implementation:
 ROOT::VecOps::RVec<float> array2D0_0(ROOT::VecOps::RVec<ROOT::VecOps::RVec<float> > &array2D0){
    ROOT::VecOps::RVec<float> result(array2D0[0].size() - 0);
    for(size_t i=0; i<array2D0[0].size() - 0; i++){
        result[i] = array2D0[0][0+i*1];
    }

    return result;
}
```

ROOT RDataFrame: A modern and advanced interface for analyzing data stored in TTree, CSV, and other formats using C++ or Python.

- In our **RDataFrame_array.py**, we offer an interface for creating **C++ template functions**, **C++ libraries**, and **RDataFrame columns** using a domain-specific language reminiscent of the traditional **tree→ Draw** query language and **tree→ SetAlias**.
- We enhance C++ functionality with Python syntax for **array slicing and projection**.
- Internally, **Python ast**, **cppyy.ll**, and **RDataFrame type information** are employed to parse Python-like functional syntax.

Goal: To Support

CTF (native reconstruction data) & AO2D Data

- **Detector and combined tracks, collisions, V0s, cascades, indices**
- Indices remapping and ambiguous match resolution

Calibration & QA Data:

- Time and spatially granular materialized views e.g.:
  - Current Estimators for TPC, TOF, V0, T0, DCA
  - Analog currents, pressure, temperature

Prototype C++ use case (AO2D skimming analysis) - Domain Specific Language (experimental)

```
from RootInteractive.Tools.RDataFrame.RDataFrame_Array import *
logging.getLogger('').setLevel(logging.DEBUG)
rdfLib = {}
dfRI = ROOT.RDF.AsRNode(ROOT.df)
dfRI = makeDefine('qPt', 'tracks[:].getQ2Pt()', dfRI, rdfLib)
dfRI = makeDefine('tgl', 'tracks[:].getTgl()', dfRI, rdfLib)
dfRI = makeDefine('tpcNcls', 'tracksExtra[:].fTPCNClsFindable', dfRI, rdfLib)
dfRI = makeDefine('nCrossed', 'tracksExtra[:].fTPCNClsFindable-tracksExtra[:].fTPCNClsFindableMinusCrossedRows',
dfRI, rdfLib)
dfRI = makeDefine('normChi2TPC', 'sqrt(tracksExtra[:].fTPCChi2NCl)', dfRI, rdfLib)
dfRI = makeDefine('normChi2ITS', 'sqrt(tracksExtra[:].fITSChi2NCl)', dfRI, rdfLib)
dfRI = makeDefine('fITSClusterMap', 'tracksExtra[:].fITSClusterMap', dfRI, rdfLib)
dfRI = makeDefine('dEdx', 'tracksExtra[:].fTPCSignal', dfRI, rdfLib)
dfRI = makeDefine('fFlags', 'tracksExtra[:].fFlags', dfRI, rdfLib)
dfRI = makeDefine('fITSChi2NCl', 'tracksExtra[:].fITSChi2NCl', dfRI, rdfLib)
dfRI = makeDefine('fTPCChi2NCl', 'tracksExtra[:].fTPCChi2NCl', dfRI, rdfLib)
dfRI = makeDefine('fTOFChi2', 'tracksExtra[:].fTOFChi2', dfRI, rdfLib)
dfRI = makeDefine('fLength', 'tracksExtra[:].fLength', dfRI, rdfLib)
dfRI = makeDefine('tgl2', 'tracks[:].getTgl()', dfRI, rdfLib)
makeLibrary(rdfLib, "ao2dRDFtest.C", includes="")
ROOT.df = ROOT.RDF.AsRNode(dfRI)
```

```
(venv3) Singularity> wc ao2dRDFtest.C
 158  306 5229 ao2dRDFtest.C
```

```
df=ROOT::RDF::AsRNode(makeRDF0(tree))
.L ao2dRDFtest.C+g
df=getDFAll(df)
df.Snapshot("snapshot2","snapshot2.roo
t",{"qPt","tgl","fITSChi2NCl","fTPCChi
2NCl","fTOFChi2","posTPCM","posTOFM","
infoDCA","tpcNcls","normChi2ITS","norm
Chi2TPC","nCrossed"});
```

makeDefine to generate C++ template code macro using Tools.RDataFrame.RDataFrame_Array

Generated code later used as compiled macro in debug mode

19

RootInteractive - conclusion

Widespread Application: RootInteractive is extensively used in numerous ALICE projects for multidimensional analysis, proving to be a crucial expert tool for various use cases. This includes distortion calibration, reconstruction optimization, dE/dx enhancement, and developing new reconstruction algorithms like track combinator.

Current Use Cases: The tool is predominantly employed for tasks related to detector calibration, simulation, Quality Assurance (QA), and global reconstruction activities for RUN3 and RUN2 (as a reference) and looking forward towards Alice 3.

Future development: There are plans to expand its machine learning functionalities by integrating functional compositions using ML (ONNX) models on the client side. Domain specific language adaptation to simplify expert physics analysis, icing performance parametarizations and skimmed data.

Domain-Specific Language Adaptation: Enhancements to simplify domain-specific language for expert physics analysis, including performance parametrizations and handling of skimmed/sampled data.

Upcoming Projects: A pilot project for N-dimensional physical analysis using sampled/skimmed data is underway

Backup

Generalized linear (kernel) regression in RootInteractive - client side

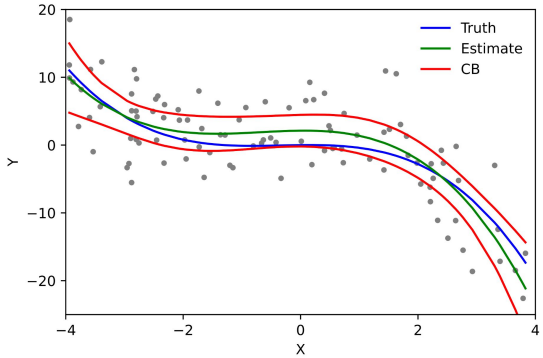
Example, declaring generalized linear kernel regression

```
regressionArray=[
        {“name”:“regre1”, “varX”:[“x1”,”x2”...,”xn”], “varY”:”y1”, “weights”:”w”}
        {“name”:“regreAgg1”, “varX”:[“x1”,”x2”...,”xn”], “varY”:”y1”, “weights”:”w”
        “varAgg”:[“xagg1”,”xagg2”...,”xaggn”],”nbinsAgg”:[...],”rollingAgg”:[...]}
]
```

- **Scikit-learn like user interface**
  - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
  - Using fit and predict
  - Regression predict new data source can be used as an alias function
- Pol0 group-by regression, mean, median,quantiles, RMS
- **Pandas groupby + ND-rolling/sliding kernel + Linear regression**
  - Interface as in the C++ code in original ND pipeline
  - Using fit and predict on the grid
  - Prediction of values and derived variables (using local fit parameters, e.g local derivatives)
  - Predict is new data source
  - **Work in progress**

- https://en.wikipedia.org/wiki/Linear_regression
- https://en.wikipedia.org/wiki/Kernel_regression

Linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables



Example of a cubic polynomial regression, which is a type of linear regression. Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(*y* | *x*) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression.

Data preparation - RDataFrame <-> awkward (new interface)

Defining RDataFrame

```
ROOT::RDataFrame df(nTracks);
auto rdf = df.Define("qVector", "getQVector(160)")
             .Define("logqVector", "ROOT::VecOps::log(qVector)")
             .Define("qStd", "StdDev(qVector)")
             .Define("qMean", "Mean(qVector)")
             .Define("qlStd", "StdDev(logqVector)")
             .Define("qlMean", "Mean(logqVector)")
             .Define("qMedian", "TMath::Median(qVector.size(), qVector.data())")
             .Define("qlMedian", "TMath::Median(qVector.size(), logqVector.data())")
             .Define("qTrunc", "truncate(qVector);")
             .Define("logqTrunc", "ROOT::VecOps::log(qTrunc);");
```

Loading awkward array

```
In [7]:
%%time
array = ak.from_rdataframe(
        rdf,
        columns=(
            "logqTrunc",
            "logqVector",
            "qMean",
            "qMedian",
            "qStd",
            "qTrunc",
#            "qVector",
            "qlMean",
            "qlMedian",
            "qlStd",
        ),
    )
CPU times: user 1min 44s, sys: 884 ms, total: 1min 45s
Wall time: 10.2 s
```

dEdx optimization example

- Defining the data and derived function (C++) with native data representation
- loading the data → awkward array
- Execution scaling with number of cores (32 used in example)
- ML training/prediction → RDataFrame ()

Significant performance increase with parallel "RDataFrame ↔ awkward" in respect to previously used direct Tree queries interface. Used extensively, e.g. in fastMCKalman (distortion simulation/correction) and in trackCombinator (V0,cascade,cosmic,loop finder) prototyping use case studies