R-Data Framework IN HEP

Mohammed Faraj

The CODATA-RDA Research Data Science Advanced Workshops

ICTP 25th-29th July 2022

Introduction

- In LHC Run3, the amount of data is expected to increase drastically, two times the data collected during Run2.
- Clusters and Machines that we use to analyse our data/MC contain multi-core CPUs/GPUs, which can be used to analysis of millions of events in an efficient way.
- Therefore, now more than ever, HEP comrades need robust, performant analysis software to take full advantage of the underlying hardware and obtain the results more efficiently.

Our Wishlist

Fast

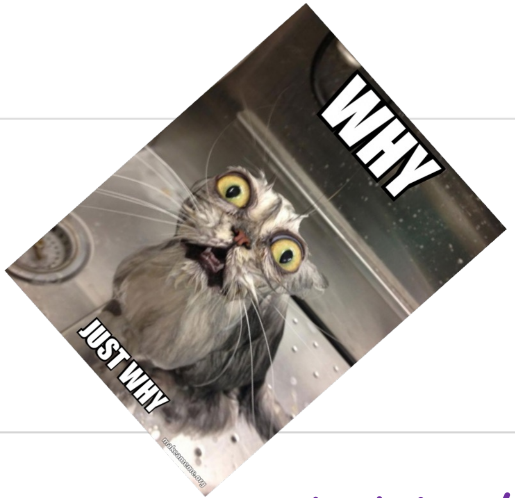
tidy

Support C++ and Python

Flexible

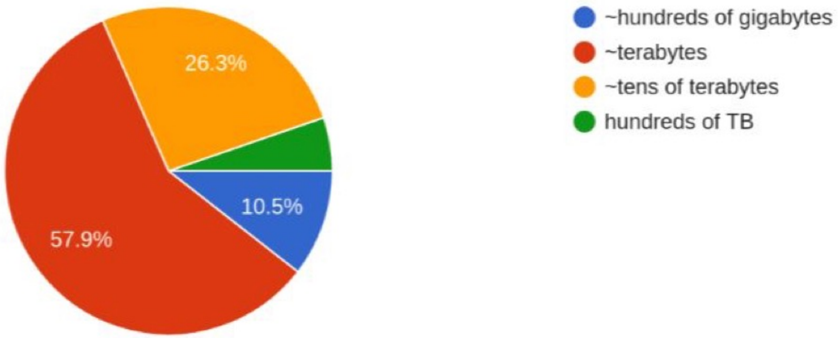
reusable

Why?

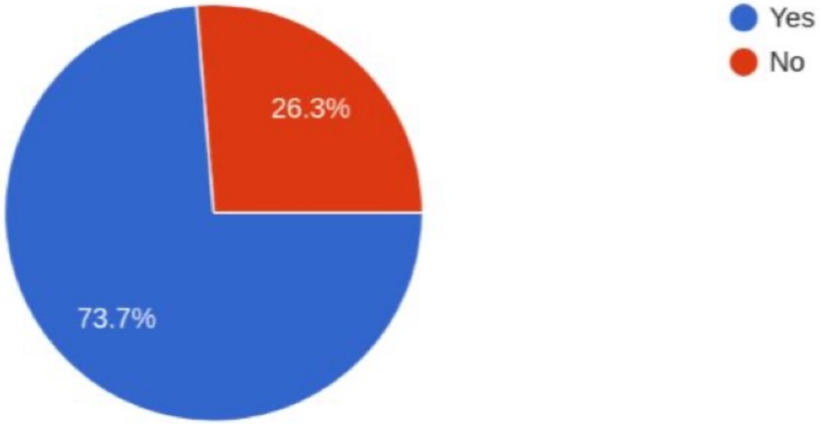


- Even in the simplest case, e.g., studying/producing Control Plots, we have:
  - About 100 Histograms
  - 130 variations of each hist
- Complexity increases based on the analysis, production mode (2D or 3D hists) and the number of regions you defined.
- Don't forget! the final aim is to analyse the entire dataset, Run2 ($140 fb^{-1}$) and Run3 ($450\ fb^{-1}$)

- NTuple size a huge problem



Is the size an issue for your analysis



Answer and Solution!

The complexity and Data size are increasing; ROOT team came up with: The New RDataFrame.

Simple

Powerful tool to analyse data using C++ and python

Parallelisable

Multithread supports, Multi-core Machines

declarative

Manage dependencies among objects without showing control-flow

Clear workflow

Graph style approach, optimized event loop

optimised

User writes analysis, ROOT takes care of optimization

RDataFrame HEP

What is RDataFrame?

RDataFrame is ROOT's high-level interface for efficient data analysis. With RDataFrame, it is possible to read, select, modify and write ROOT data, as well as easily produce histograms, cut-flow reports and other results.

Analysis Language
~50% C++
~50% Python

Storage
Local Disk
Fast-access Network storage
EOS or other not-so-fast backends

PlatForm

Laptop or PC

Many-core Machine

computing cluster + Submission

ROOT::RDataFrame addresses all use cases with a single High-Level programming model

RDataFrame overview

RDataFrame reference guide

RDataFrame tutorials

- Three simple setups for a user.
- Build a DataFrame object from a source.
- Transformations in the input
  - Define: New columns from existing columns
  - Filter: Create ranges of events by applying cuts
- Apply actions on the transformed data, e.g. Histograms

Design Principles

Elements of **declarative programming**
"user says what, ROOT chooses how"

High level interfaces provide less typing, increased readability, abstraction of complex operations
...and allow **transparent optimisations**, e.g. multi-thread parallelisation, lazy evaluation and caching

Elements of **functional programming**
pure functions & higher-order functions

Users code in terms of **small reusable components**

Less side-effects and less shared state increase **thread-safety and code correctness** traditional way

```python
import ROOT

fIn = ROOT.TFile.Open("file.root")
tree = fIn.tree

for event in tree:
    if len(event.muons)<1: continue
    if not event.MET>20: continue

    for muon in event.muons:
        if muon.pt > 25 and  abs(muon.eta)<2.4 \
        and muon.dz<0.1 and muon_dxy<0.01 \
        and muon.relIso<0.5:

         selmuon_pt = muon_pt
```

loop over events

loop over muons

parallelisation difficult - let alone in python

introducing dataframes

```python
import ROOT

ROOT.ROOT.EnableImplicitMT()

RDF = ROOT.ROOT.RDataFrame
d = RDF(treeName, inputFile)

d = d.Filter("nMuon>1 && MET>20”)\
     .Define(“SelMuon_pt”
     ,“Muon_pt[Muon_pt>25 \
     & abs(Muon_eta)<2.4 \
     & Muon_dz<0.1 & Muon_dxy<0.01 \
     & Muon_relIso<0.5]”)
```

transparently parallelised

Be prepared for massive changes

RDataFrame Example I

```
ROOT::RDataFrame df(dataset);
```

Run on This (ROOT, CSV, NanoAOD..etc) dataset

```
auto df2=df.Filter("x > 0")
           .Define("r2","x*x+y*y");
```

Only Accept events with x larger than 0, then define a new variable, r2, which is equal to x*x+y*y

```
auto rHist = df2.Histo1D("r2");
```

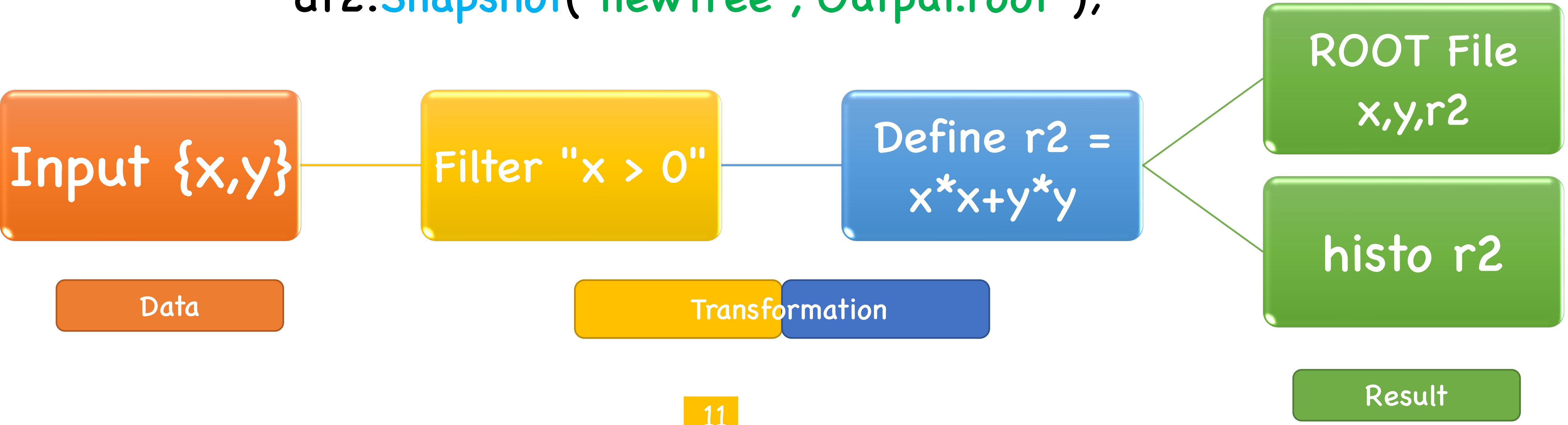
Plot r2 for events that are passed the cut

```
df2.Snapshot("newTree","Output.root");
```

Write the skimmed data and r2 to a new ROOT file

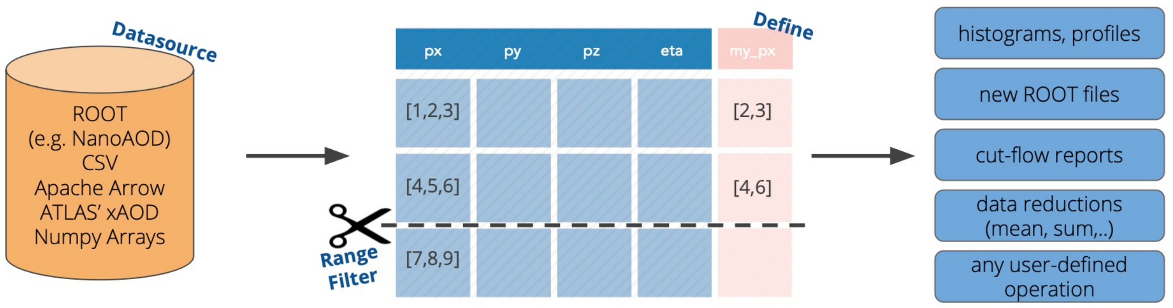
In term of Graphs

```
ROOT::RDataFrame df(dataset);
auto df2=df.Filter("x > 0")
                  .Define("r2","x*x+y*y");
auto rHist = df2.Histo1D("r2");
df2.Snapshot("newTree","Output.root");
```



RDataFrame Example II

```
RDataFrame("tree", "f.root").Define("pt", "muon_pt[muon_eta > 0]").Histo1D("pt")->DrawClone();
```




```
RVecD selectPt(RVecD &pt, RVecD &eta) { return pt[eta > 0]; }

auto h = RDataFrame("tree", "f.root")
                .Define("pt", selectPt, {"muon_pt", "muon_eta"})
                .Histo1D<RVecD>("pt");
h->Draw();
```

[Rvec reference guide](#)

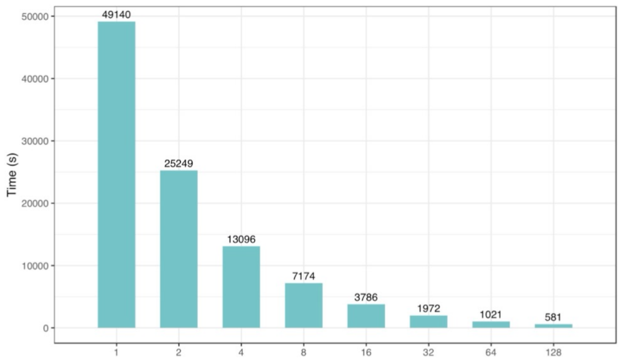
RDataFrame Example III

```
auto leptons = [] (std::vector<float> el_pt,std::vector<float> el_eta,std::vector<float> el_phi,std::vector<float> el_e,
std::vector<float> mu_pt,std::vector<float> mu_eta,std::vector<float> mu_phi,std::vector<float> mu_e)
```

```
std::vector<TLorentzVector> leptons;
leptons.clear();
TLorentzVector v1(0,0,0,0);
TLorentzVector v2(0,0,0,0);
if(mu_pt.size() == 2) {
v1.SetPtEtaPhiE(mu_pt.at(0),mu_eta.at(0),mu_phi.at(0),mu
_e.at(0));
v2.SetPtEtaPhiE(mu_pt.at(1),mu_eta.at(1),mu_phi.at(1),mu
_e.at(1));
leptons.push_back(v1);
leptons.push_back(v2);
}
leptons.push_back(v1);
leptons.push_back(v2);

return leptons;
```

```
frame = frame.Define("leptons", leptons,
{"el_pt","el_eta","el_phi","el_e","mu_pt","mu_eta
","mu_phi","mu_e"});
```



RDataFrame real-world use cases

Distributed analysis with RDataFrame in TOTEM (Avati et al., 2019)

4.7 TB of ROOT data processed in ~10 minutes on a cluster with 128 cores

Dark Matter sensitivity study (Pani and Polesello, 2018)

ATLAS: xAOD-DataSource

ALICE: Apache Arrow support contributed by G. Eulisse, currently investigating RDF-based analysis workflow for Run 3

FCC plans to develop workflows based on RDF (C. Helsens, "General status and plans", FCC week 2019, slide 20)

Other Examples and frameworks

W mass analysis Framework

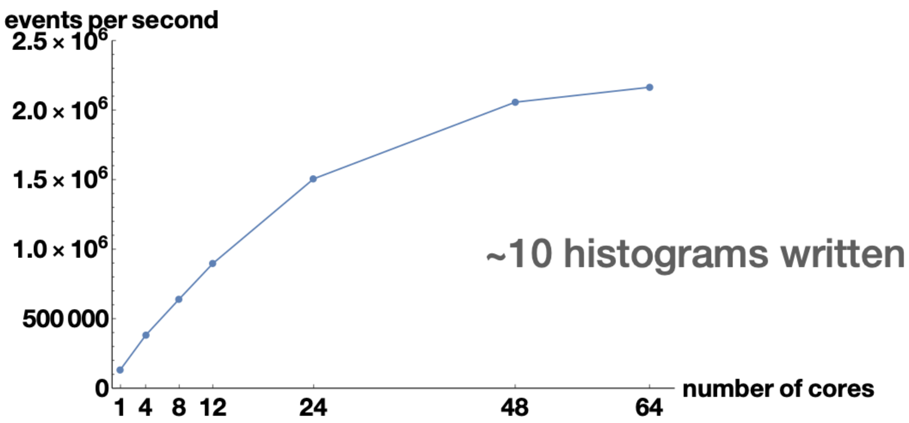
LoopSUSYFrame ATLAS analysis tool

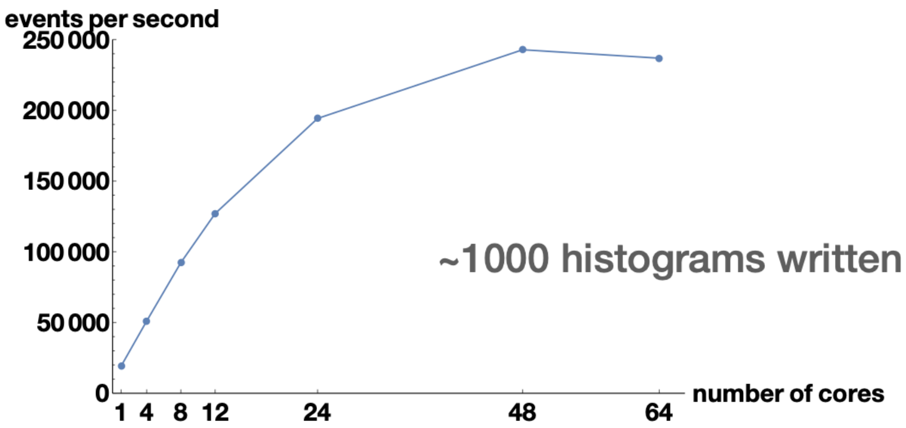
Bamboo Analysis Framework

Timber analysis Framework

Kit's CROWN

Turnaround of a few hours for O(100) plots "thousands of Hists" of CMS Run2 data on a batch system





To conclude

- Several analyses have developed their analysis framework based on the RDataFrame.
- Encouraging performance results are obtained.
- The four main experiments at CERN have started using the RDF.
- A flexible framework.

References

- ROOT::RDataFrame, Enrico Guiraud for the ROOT Team, 2019
- Readable and efficient HEP data analysis with bamboo, Pieter David
- Using the new RDataFrame in a physics study, Elisabetta Manca
- High-throughput analysis with Modern ROOT, ROOT team 2020
- Wmass analysis framework, Suvankar et al.
- Recent advancements in high-performance analysis and statistical modelling with ROOT, Enrico Guiraud, 2021
- Optimized access to ATLAS analyses within the ROOT Framework, Umesh Worlikar