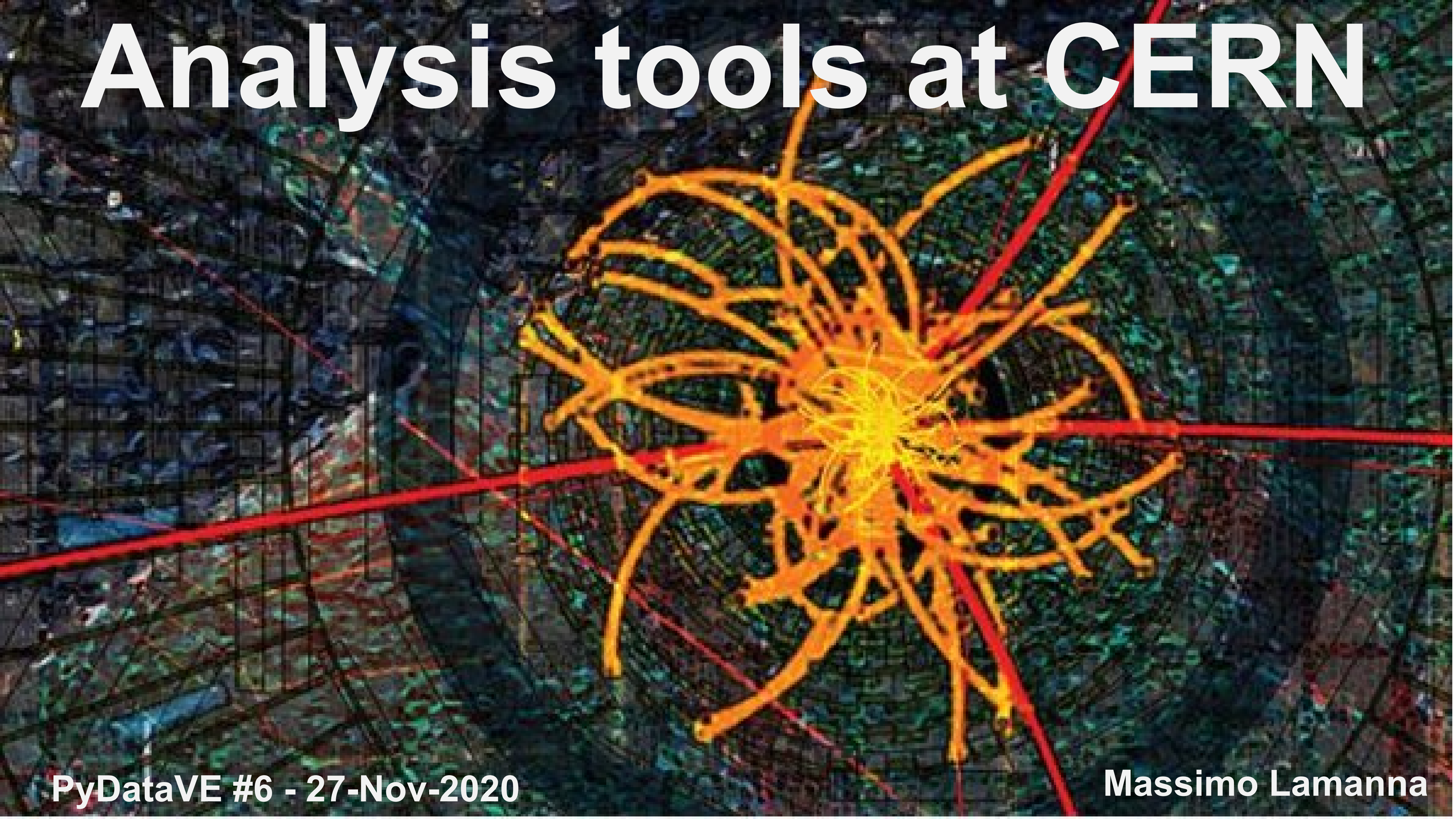
Analysis tools at CERN



PyDataVE #6 - 27-Nov-2020

Massimo Lamanna

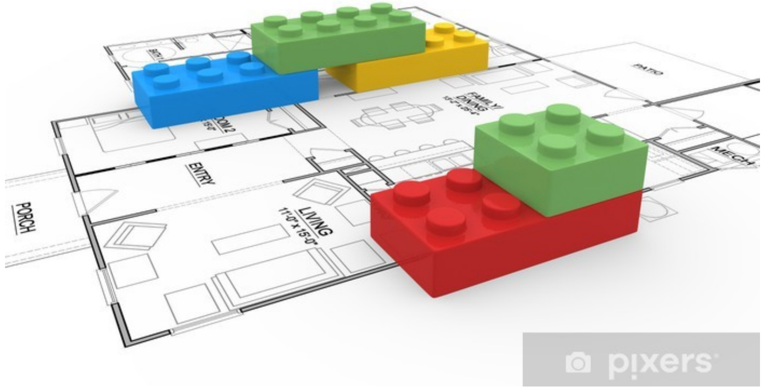
Goals of the presentation

- Informally share some info about the 'traditional' (yet bleeding edge) tools for data analysis in High-Energy Physics (HEP)
  - Until recently, full unique self-contained ecosystem
  - Now feeling the presence of matplotlib, pandas, ...
- The real answer (for me) would be:
  - why all these analysis tools (not only CERN...) are *so* complicated :) ?
  - no direct answer (maybe you can help :)

History and present

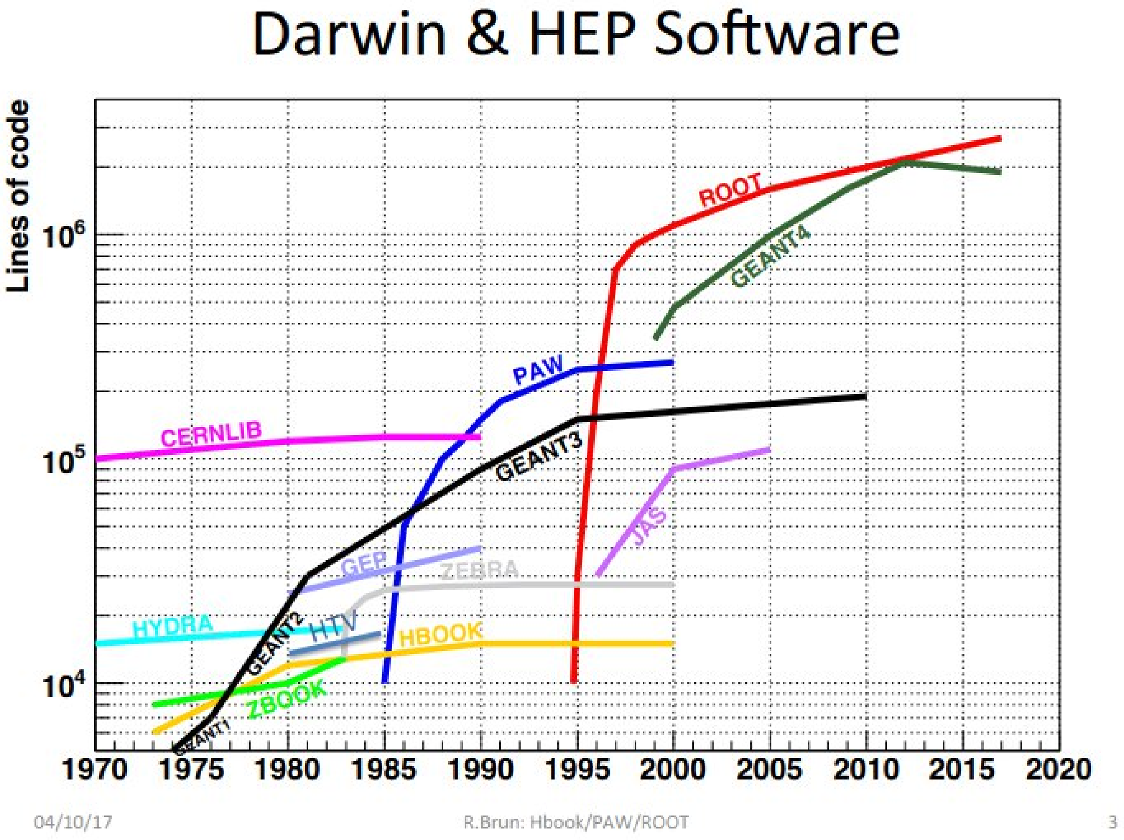
- HBOOK (1970-1980)
- PAW (1990)
- ROOT (2000 onward)

- Basic element: Histograms!
  - Batch/mainframe world (submit and job and print the output)
  - Fortran world (empower the language with memory management)
  - Doc: few-page printouts :)



R. Brun seminar

- 40 Years of Large Scale Data Analysis in HEP - the HBOOK, PAW and ROOT Story



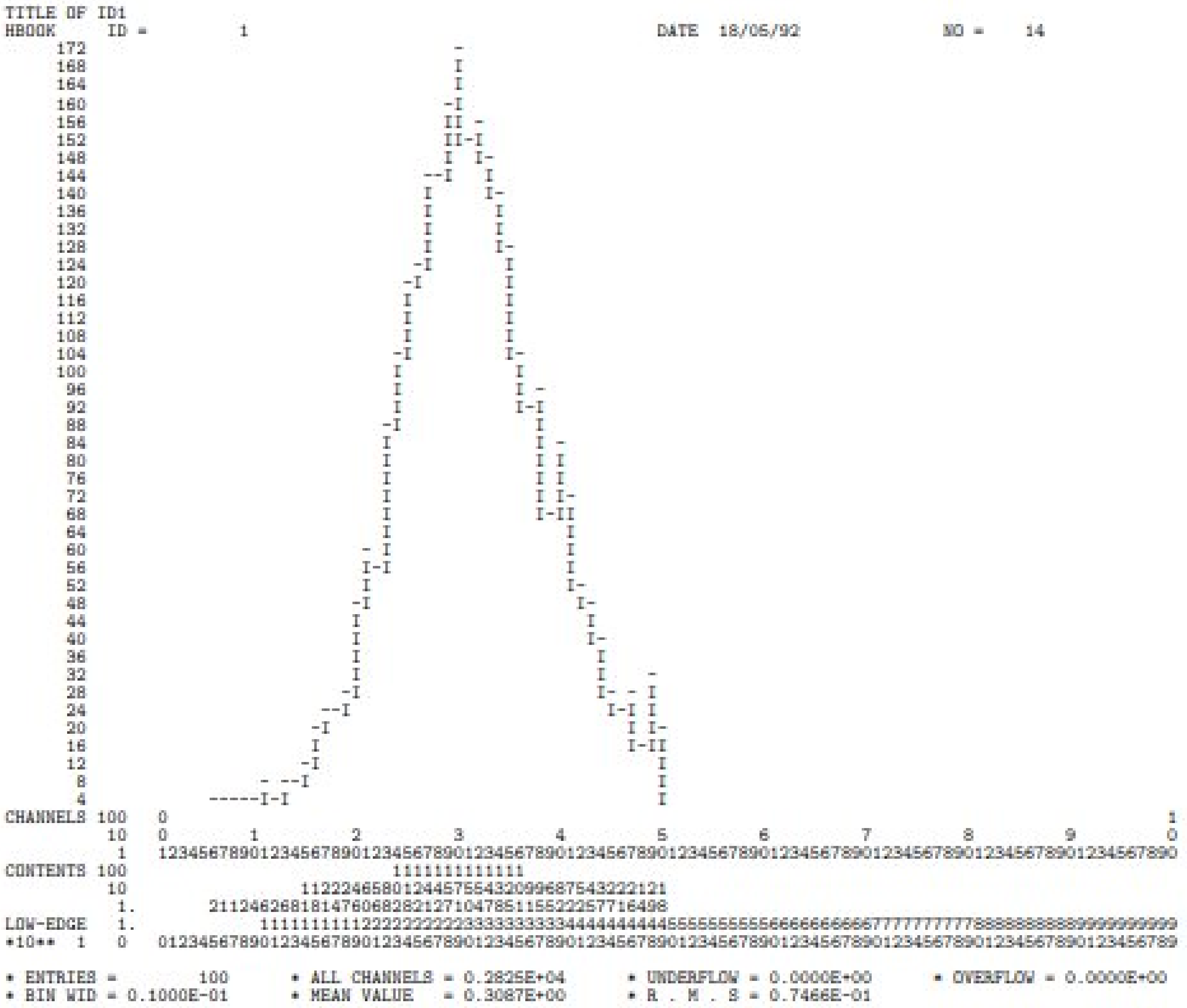
The main authors of PAW appear on the photo above (left to right):

Rene Brun – project coordinator
Pietro Zanarini – author of the user interface module, KUIP
Olivier Couet (standing) – author of the graphics module, HIGZ
Carlo Vandoni – author of the array manipulation module, SIGMA

- https://indico.cern.ch/event/667648/attachments/1526850/2425425/cern17.pdf

HBOOK world

- Object with properties and methods
- Frequency table:
  - 100 values measured
  - Mean value about 0.31 with a RMS of 0.07
  - No value recorded higher than 1 and lower than 0 (over/underflow)
  - ...
- Actually there is an **algebra** of these objects...



`CALL HBOOK1 (ID,CHTITL,NX,XMI,XMA,VMX)`

Action: Books a one-dimensional histogram.

Input parameter description:

ID	histogram identifier, integer non zero
CHTITL	histogram title (character variable or constant up to 80 char
NX	number of channels
XMI	lower edge of first channel
XMA	upper edge of last channel
VMX	upper limit of single channel content (see below). VMX=0. means 1 word per channel (no packing).

This is the would-be constructor...

`CALL HFILL (ID,X,Y,WEIGHT)`

Action: Fills a 1-dimensional or a 2-dimensional histogram. The channel which contains the va
and for two-dimensionals the cell that contains the point (X,Y), gets its contents increased by WE
All booked projections, slices, bands, are filled as well.

Input parameter description:

Update (add a value)

ID	histogram identifier
X	value of the abscissa
Y	value of the ordinate
WEIGHT	event weight (positive or negative)

`CALL HOPERA (ID1,CHOPER,ID2,ID3,C1,C2)`

Action: Fills an histogram I3 with values such that, logically (operands are the bin contents)

```
ID3 = C1 * ID1 (OPERATION) C2 * ID2
```

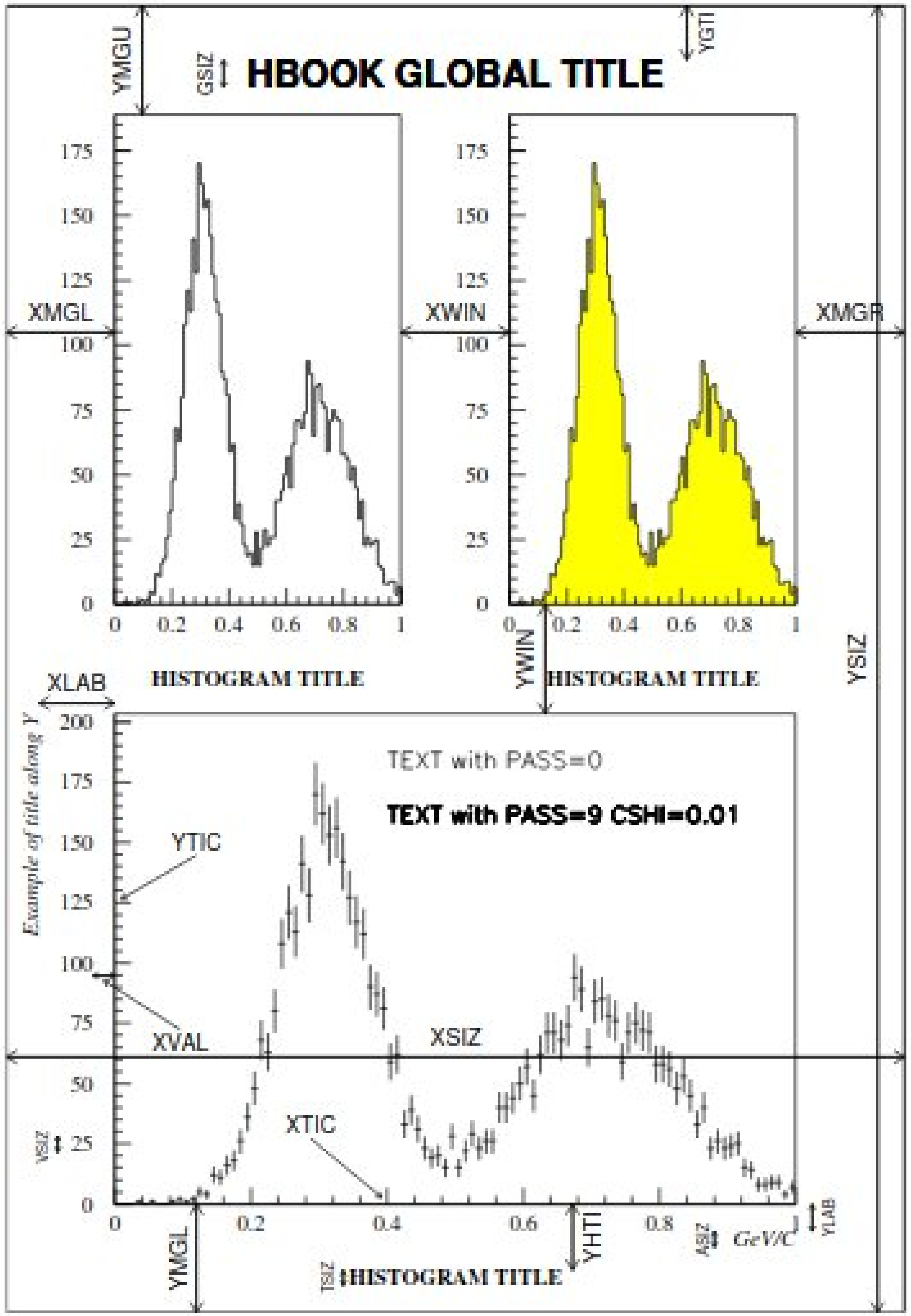
Input parameters:

ID1,ID2	Operand histogram identifiers.
CHOPER	Character variable specifying the kind of operation to be performed (+,-,*,/); 'B' compute binomial errors; 'E' compute error bars on the resulting histograms correctly, assuming that the input his tograms ID1 and ID2 are independent. For instance /BE will generate binomial errors for the division of ID1 by ID2.
ID3	Identifier of the histogram containing the result after the operation.
C1,C2	Multiplicative constants.

histo3 = c1 * histo1 + c2 * histo2

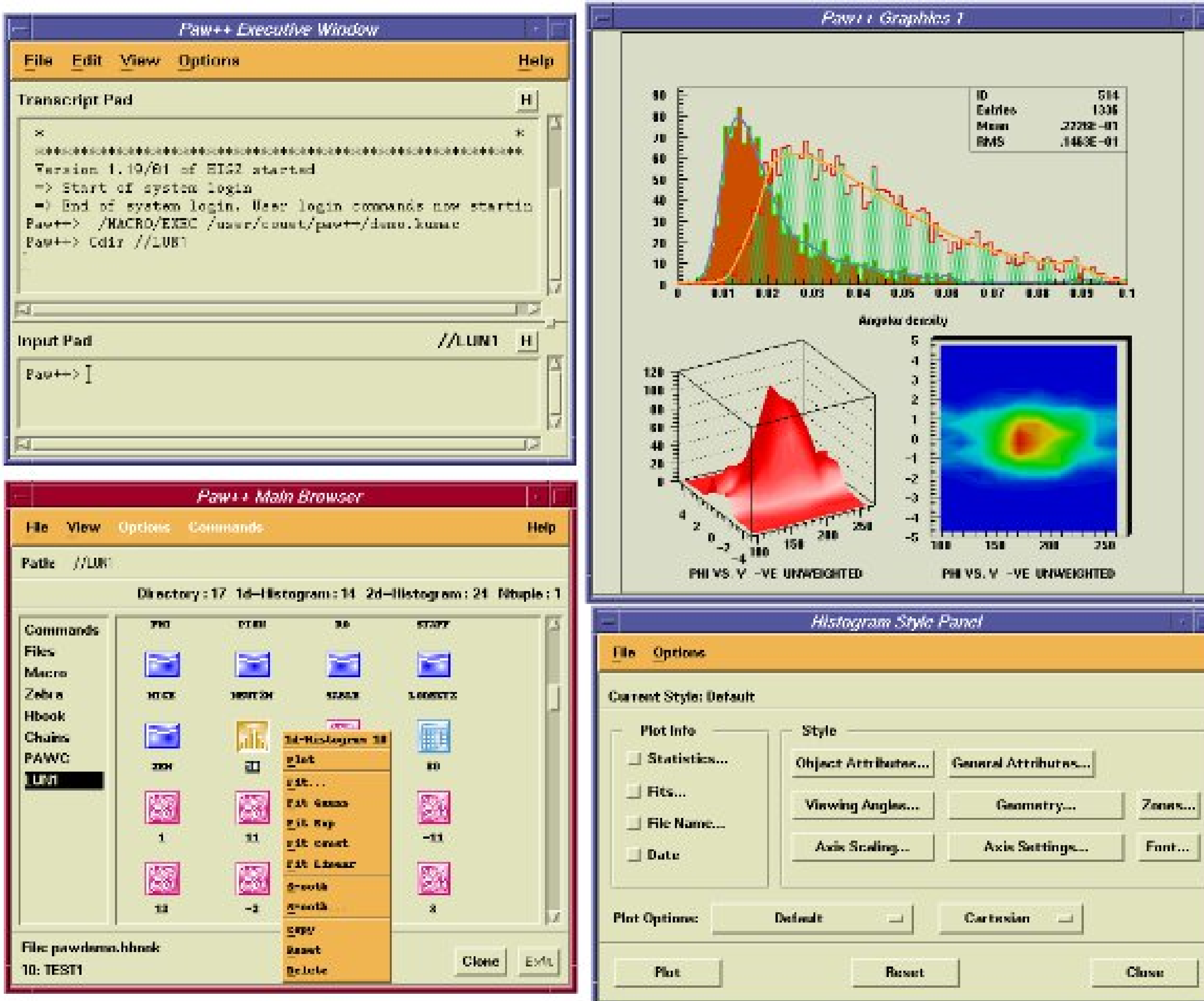
PAW (circa 1990)

- HBOOK --> PAW
  - https://en.wikipedia.org/wiki/Physics_Analysis_Workstation
  - ~500-page manual !
- Mainframe --> Workstation
  - Graphics capabilities
  - Edit in one window, see the results in another
- Integration
  - HBOOK
    - Minimisation and fitting
    - ntuples to store data (memory and disk - think to binary CSV)
  - HPLOT (graphics)
  - SIGMA (manipulate 1d arrays)
  - COMIS (command line and FORTRAN interpreter). First exploration of interactivity
  - ...



Examples of the SIGMA processor (1)

```
    zone 2 2
❷   APPLICATION SIGMA
       X=ARRAY(200,0#2*PI)
       sinus=sin(x)
       sinx=sin(x)/x
❷   EXIT
    gra 200 x sinus
    set dmod 2
    gra 200 x sinx 1
    set dmod 0
❶   SIGMA x=array(300,0#8)
    sigma g=cosh(x)+sin(1/(.1+x*x))
    gra 300 x g
    sigma x=array(300,0#3)
❸   GRAPH 300 x $SIGMA(cosh(x)+sin(1/(.1+X*X)))
    sigma x=array(300,0#1)
❹   GRAPH 300 x $RSIGMA(cosh(x)+sin(1/(.1+X*X)))
```



ROOT
Data Analysis Framework

About | Install | Get Started | Forum & Help | Manual | Blog Posts | Contribute

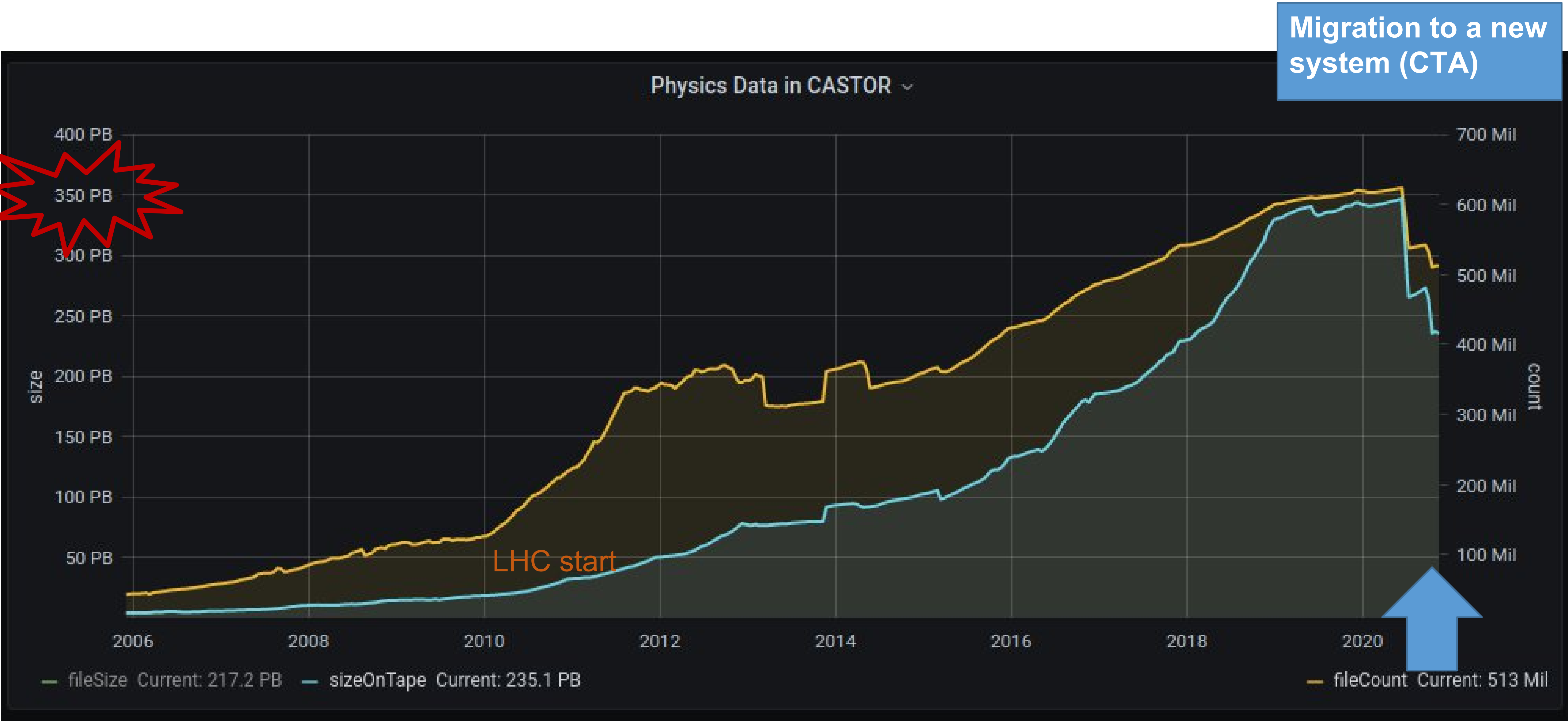
ROOT: analyzing petabytes of data, scientifically.

An open-source data analysis framework used by high energy physics and others.

Learn more | Install v6.22/02

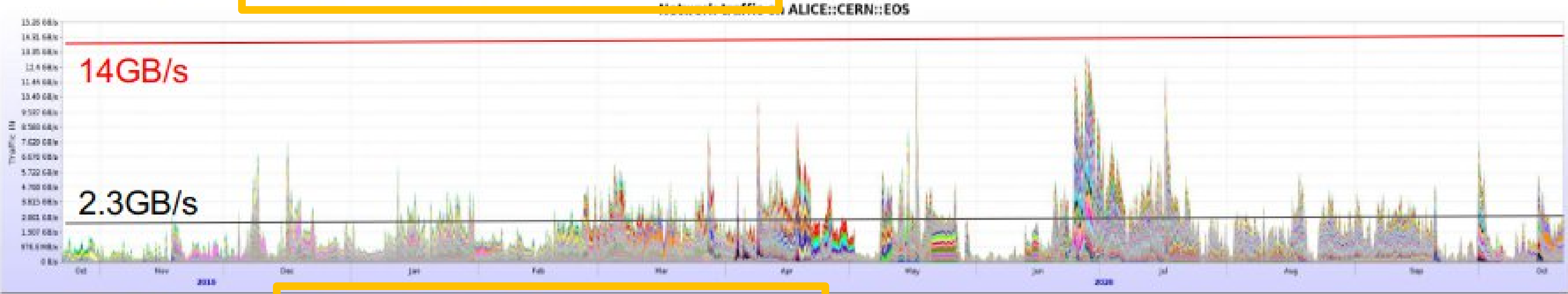
Get Started | **Reference** | **Forum & Help** | **Gallery**

CASTOR: the CERN tape archive

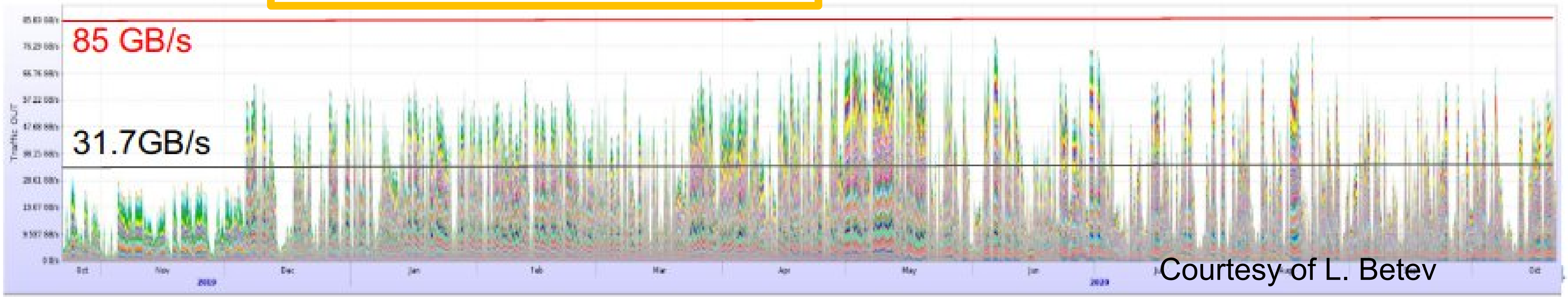


ALICE::CERN::EOS in the last 365 days

- Traffic in 72.6PB @2.3GB/s average

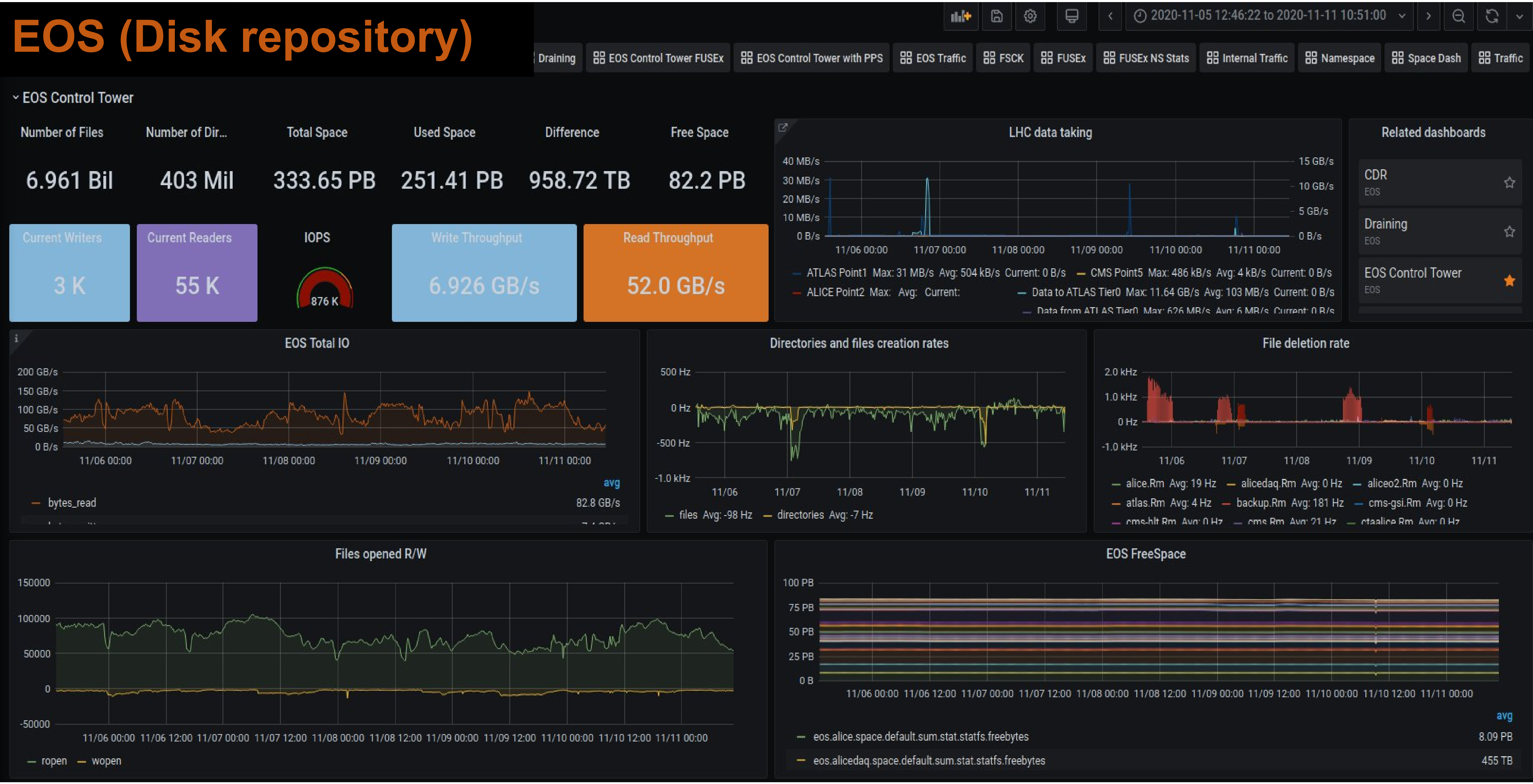


- Traffic out 1EB @31.7GB/sec average



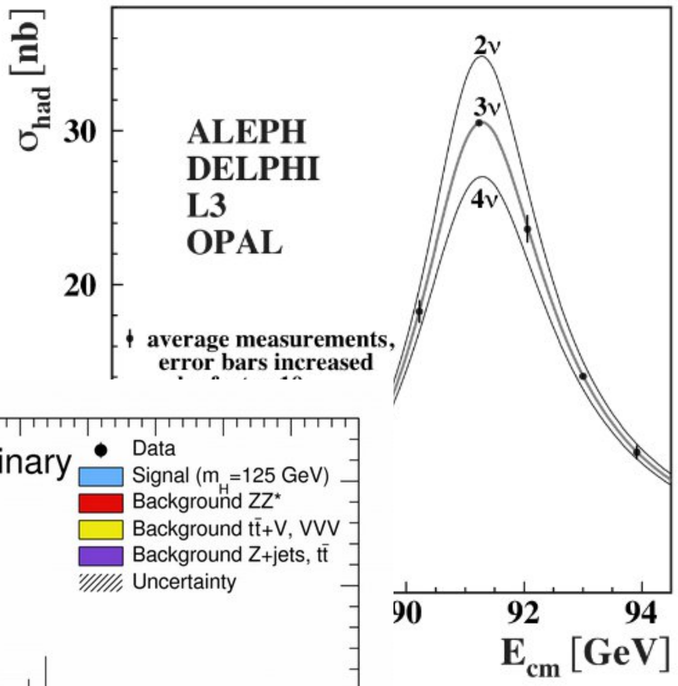
Courtesy of L. Betev

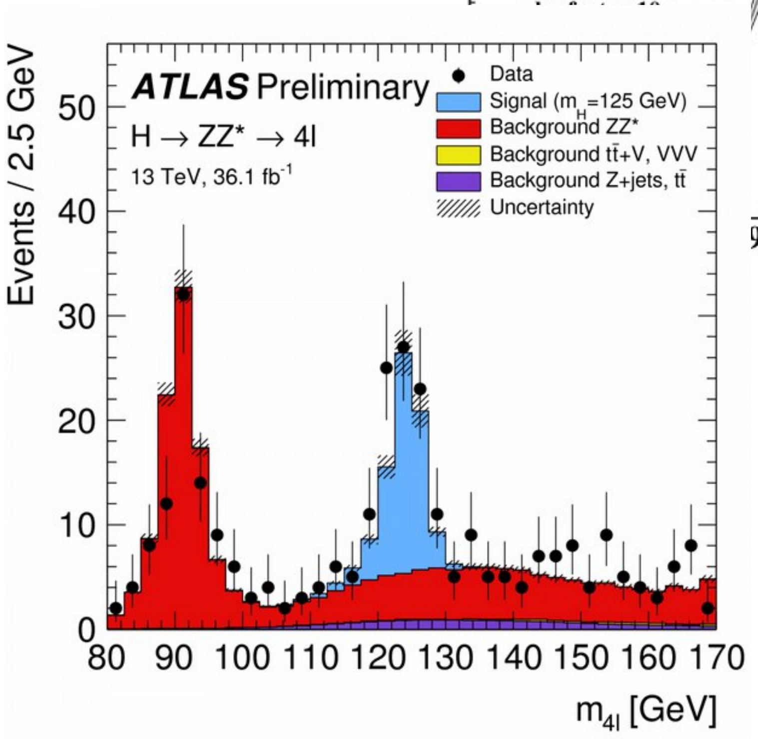
EOS (Disk repository)



ROOT

- PAW successor
  - Same architect as in PAW etc... (R. Brun et al.)
- FORTRAN --> C++
  - Also as language to steer the system (interpreted C++)
- Emphasis on data structure (I/O, disk storage)
- <u>Language </u>to discuss and share results





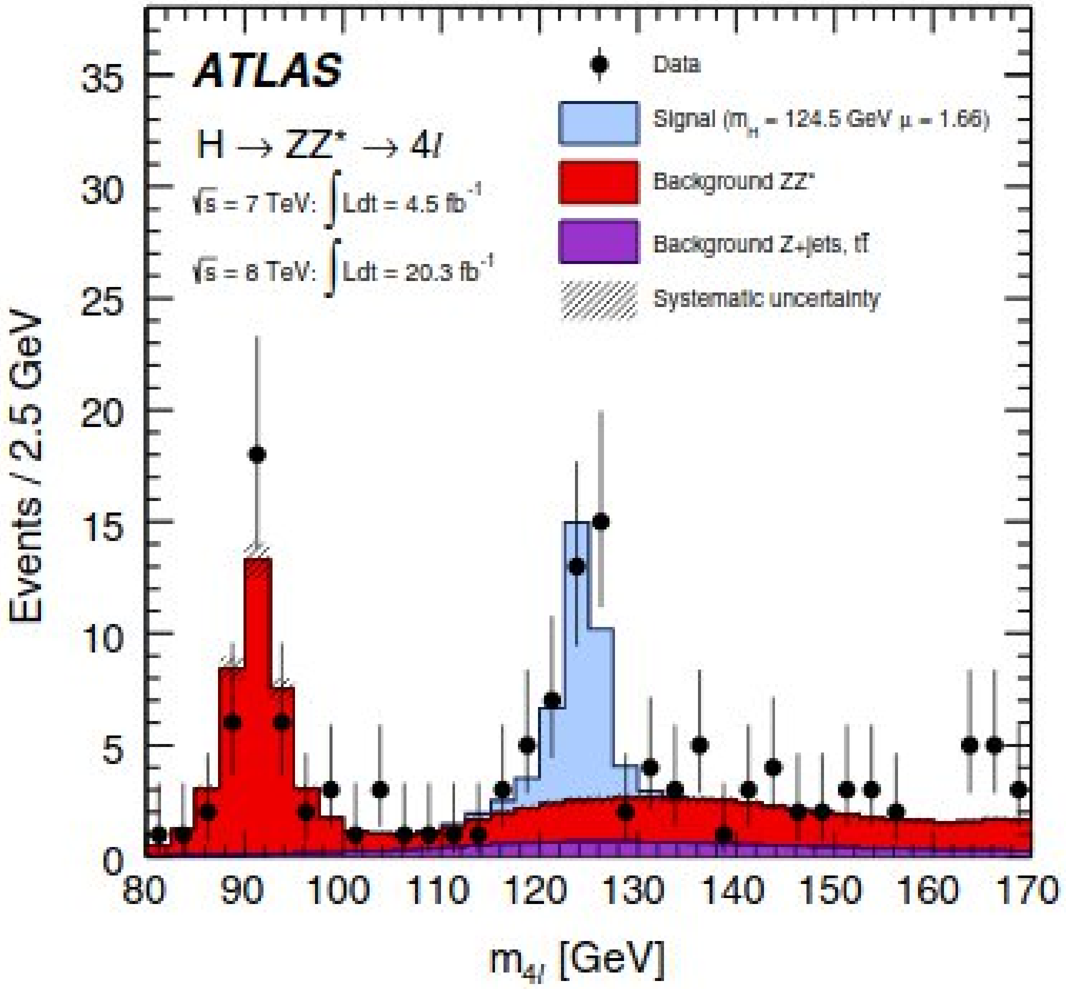
R. Brun, F. Rademakers,ROOT—an object oriented data analysis framework, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 389, 81 (1997)

Some highlights

- C++ --> Python
  - Python as front-end language
  - The engine chews C++ (performance critical)
- If you know the history, you can find traces of *good ol' fortran*:
  - Predefined fitting of a gaussian? The keyword is 'GAUS' (4 chars as reserved in HBOOK)

```
g3 = TF1( 'g3', 'gaus', 110, 121 )
```

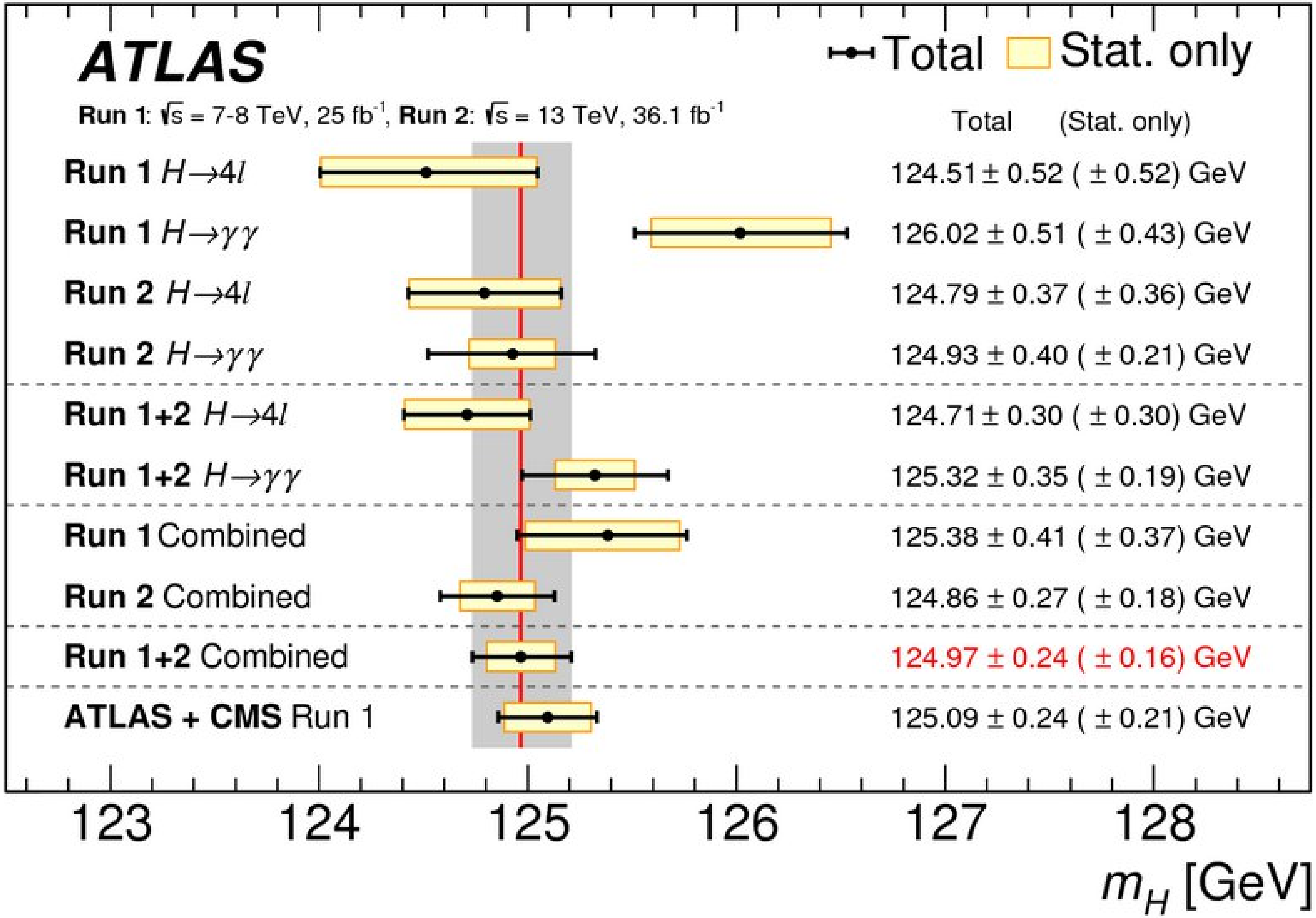
based on an analysis optimized to measure the signal strength [18]. The expected statistical uncertainty for the 2D fit with the observed μ value of 1.66 is 0.49 GeV, close to the observed statistical uncertainty. With the improved



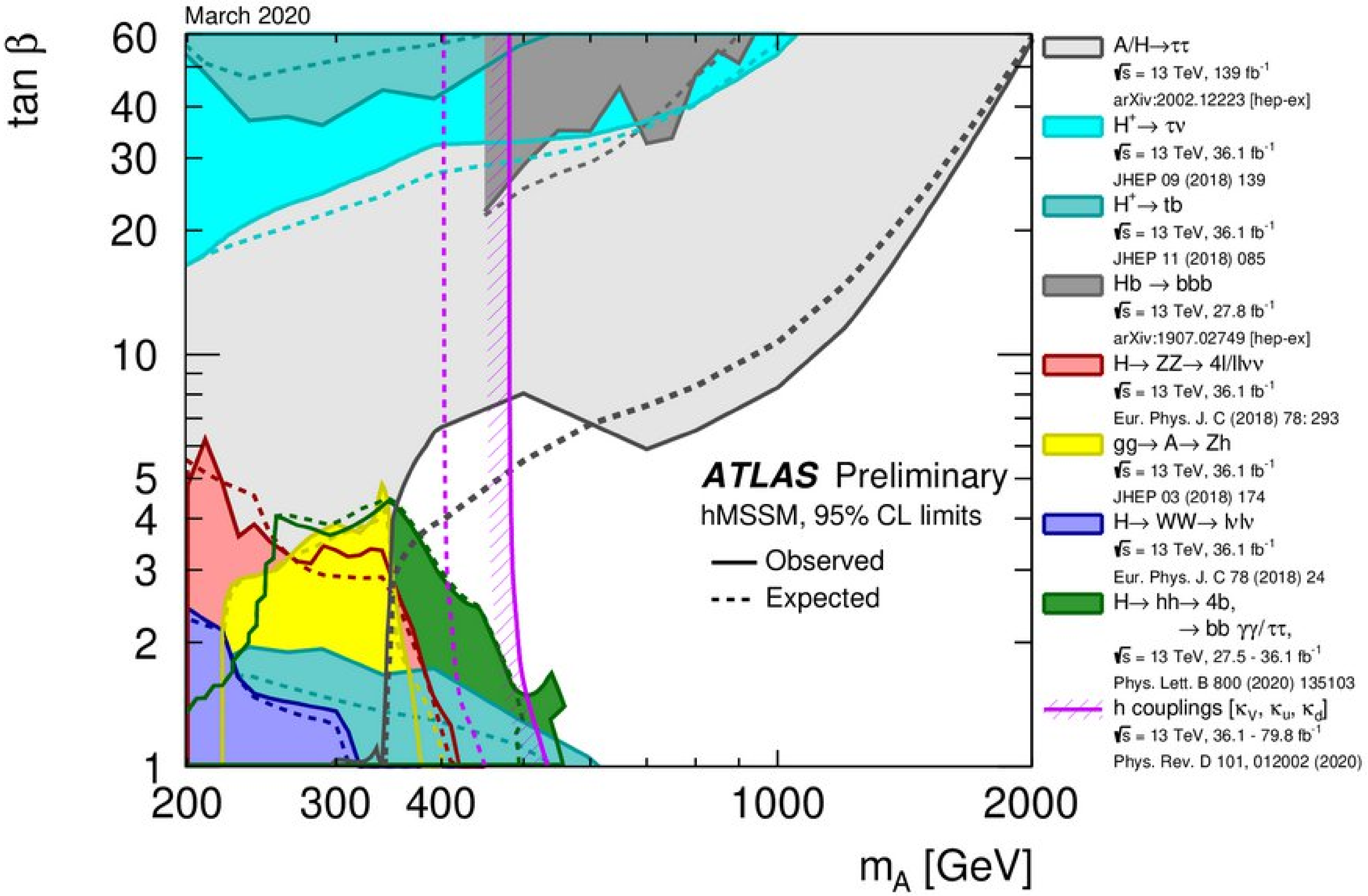
(a)

uncertainties on the electron and muon energy scales, the mass uncertainty given above is predominantly statistical with a nearly negligible contribution from systematic uncertainties. The mass measurement performed with the 1D model gives $m_H = 124.63 \pm 0.54$ GeV, consistent with the 2D result where the expected difference has a root mean square (RMS) of 250 MeV estimated from Monte Carlo pseudo-experiments. These measurements can be compared to the previously reported result [15] of $124.3^{+0.6}_{-0.5}(\mathrm{stat})^{+0.5}_{-0.3}(\mathrm{syst})$ GeV, which was obtained using the 1D model. The difference between the measured values arises primarily from the changes to the channels with electrons—the new calibration and resolution model, the introduction of the combined track momentum and cluster energy fit, and the improved identification, as well as the recovery of noncollinear FSR photons, which affects all channels. In the 120–130 GeV mass window, there are four new events and one missing event as compared to Ref. [15]. Finally, as a third cross-check, the measured mass obtained with the per-event-error method is within 60 MeV of the value found with the 2D method.

Figure 7 shows the scan of the profile likelihood, $-2\ln\Lambda(m_H)$, for the 2D model as a function of the mass of the Higgs boson for the four final states, as well as for all



ATLAS
Total
Stat. only
Run 1: √s = 7-8 TeV, 25 fb-1, Run 2: √s = 13 TeV, 36.1 fb-1
Total (Stat. only)
Run 1 H→4l
124.51 ± 0.52 (± 0.52) GeV
Run 1 H→γγ
126.02 ± 0.51 (± 0.43) GeV
Run 2 H→4l
124.79 ± 0.37 (± 0.36) GeV
Run 2 H→γγ
124.93 ± 0.40 (± 0.21) GeV
Run 1+2 H→4l
124.71 ± 0.30 (± 0.30) GeV
Run 1+2 H→γγ
125.32 ± 0.35 (± 0.19) GeV
Run 1 Combined
125.38 ± 0.41 (± 0.37) GeV
Run 2 Combined
124.86 ± 0.27 (± 0.18) GeV
Run 1+2 Combined
124.97 ± 0.24 (± 0.16) GeV
ATLAS + CMS Run 1
125.09 ± 0.24 (± 0.21) GeV
123
124
125
126
127
128
m_H [GeV]



March 2020
tan β
60
40
30
20
10
5
4
3
2
1
200
300
400
1000
2000
m_A [GeV]
ATLAS Preliminary
hMSSM, 95% CL limits
Observed
Expected
A/H→ττ
√s = 13 TeV, 139 fb⁻¹
arXiv:2002.12223 [hep-ex]
H⁺→ τν
√s = 13 TeV, 36.1 fb⁻¹
JHEP 09 (2018) 139
H⁺→ tb
√s = 13 TeV, 36.1 fb⁻¹
JHEP 11 (2018) 085
Hb → bbb
√s = 13 TeV, 27.8 fb⁻¹
arXiv:1907.02749 [hep-ex]
H→ ZZ→ 4l/llνν
√s = 13 TeV, 36.1 fb⁻¹
Eur. Phys. J. C (2018) 78: 293
gg→ A→ Zh
√s = 13 TeV, 36.1 fb⁻¹
JHEP 03 (2018) 174
H→ WW→ lνlν
√s = 13 TeV, 36.1 fb⁻¹
Eur. Phys. J. C 78 (2018) 24
H→ hh→ 4b,
→ bb γγ/ττ,
√s = 13 TeV, 27.5 - 36.1 fb⁻¹
Phys. Lett. B 800 (2020) 135103
h couplings [κ_V, κ_u, κ_d]
√s = 13 TeV, 36.1 - 79.8 fb⁻¹
Phys. Rev. D 101, 012002 (2020)

FILE EDIT VIEW INSERT CELL KERNEL WIDGETS HELP Not Trusted ROOT C++

Code

First try without starting values for the parameters This defaults to 1 for each param. this results in an ok fit for the polynomial function however the non-linear part (lorenzian) does not respond well.

In [7]:
```
fitFcn->SetParameters(1,1,1,1,1,1);
histo->Fit("fitFcn","0");
```

```
 FCN=58.9284 FROM MIGRAD    STATUS=CONVERGED     618 CALLS         619 TOTAL
                     EDM=1.54329e-09    STRATEGY= 1  ERROR MATRIX UNCERTAINTY   1.2 per cen
t
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0          -8.64715e-01   8.87889e-01   3.02210e-05  -3.15277e-06
   2  p1           4.58434e+01   2.64076e+00   6.35729e-04   1.78463e-05
   3  p2          -1.33214e+01   9.77307e-01  -1.31737e-04   3.73302e-05
   4  p3           1.38074e+01   2.20785e+00  -1.29864e-03  -9.22424e-06
   5  p4           1.72308e-01   3.72077e-02  -5.22394e-06  -1.45631e-03
   6  p5           9.87281e-01   1.13098e-02   2.92804e-06  -3.44378e-04
```

Second try: set start values for some parameters

In [8]:
```
fitFcn->SetParameter(4,0.2); // width
fitFcn->SetParameter(5,1);   // peak

histo->Fit("fitFcn","V+","ep");
```

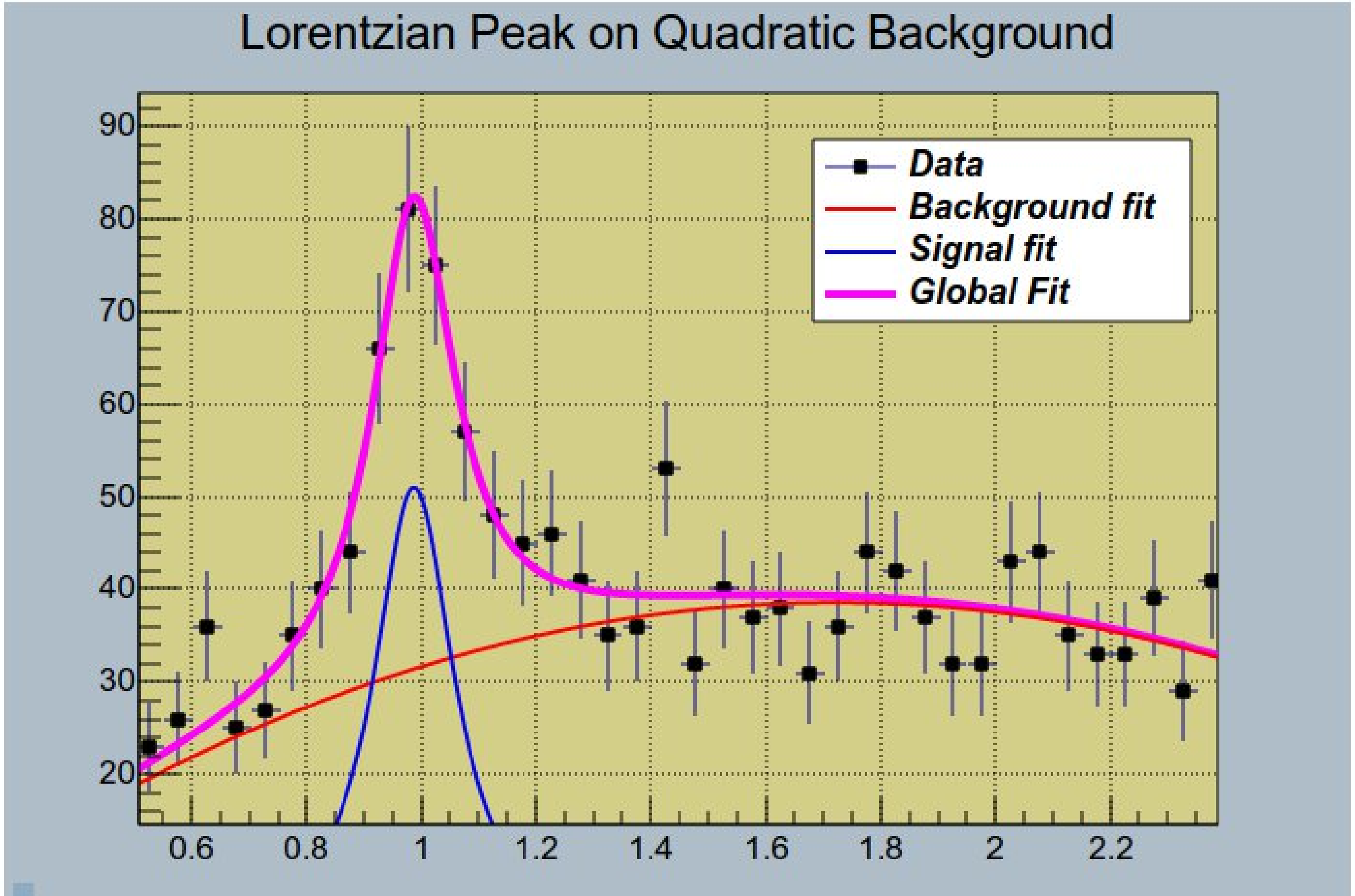
FILE EDIT VIEW INSERT CELL KERNEL WIDGETS HELP

Trusted | ROOT C++

Markdown

In [12]:

```
%jsroot on
gROOT->GetListOfCanvases()->Draw()
```

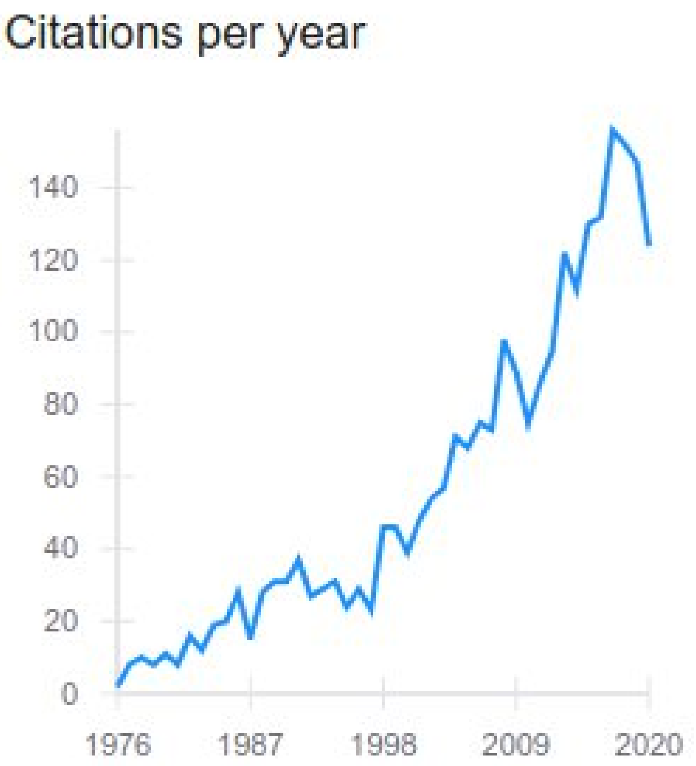


Minuit: A System for Function Minimization and Analysis of the Parameter Errors and Correlations

F. James (CERN), M. Roos (CERN)
Jul 1, 1975

38 pages
Published in: *Comput.Phys.Commun.* 10 (1975) 343-367
DOI: 10.1016/0010-4655(75)90039-9
Report number: CERN-DD-75-20
View in: ADS Abstract Service, CERN Document Server

cite   2,542 citations



https://en.wikipedia.org/wiki/MINUIT

- Eventually rewritten in C++ (original FORTRAN)
- Java version + Python front-end

Fillrandom

FillRandom example

Author: Wim Lavrijsen

This notebook tutorial was automatically generated with ROOTBOOK-izer from the macro found in the ROOT repository on Wednesday, November 11, 2020 at 09:34 AM.

In [1]:

```
from ROOT import TCanvas, TPad, TFormula, TF1, TPaveLabel, TH1F, TFile
from ROOT import gROOT, gBenchmark



c1 = TCanvas( 'c1', 'The FillRandom example', 200, 10, 700, 900 )
c1.SetFillColor( 18 )

pad1 = TPad( 'pad1', 'The pad with the function',  0.05, 0.50, 0.95, 0.95, 21 )
pad2 = TPad( 'pad2', 'The pad with the histogram', 0.05, 0.05, 0.95, 0.45, 21 )
pad1.Draw()
pad2.Draw()
pad1.cd()

gBenchmark.Start( 'fillrandom' )
```

```
Welcome to JupyROOT 6.20/06
```

A function (any dimension) or a formula may reference an already defined formula

In [2]:

```
form1 = TFormula( 'form1', 'abs(sin(x)/x)' )
sqroot = TF1( 'sqroot', 'x*gaus(0) + [3]*form1', 0, 10 )
sqroot.SetParameters( 10, 4, 1, 20 )
pad1.SetGridx()
pad1.SetGridy()
pad1.GetFrame().SetFillColor( 42 )
pad1.GetFrame().SetBorderMode( -1 )
pad1.GetFrame().SetBorderSize( 5 )
sqroot.SetLineColor( 4 )
sqroot.SetLineWidth( 6 )
sqroot.Draw()
lfunction = TPaveLabel( 5, 39, 9.8, 46, 'The sqroot function' )
lfunction.SetFillColor( 41 )
lfunction.Draw()
c1.Update()
```

In [4]:

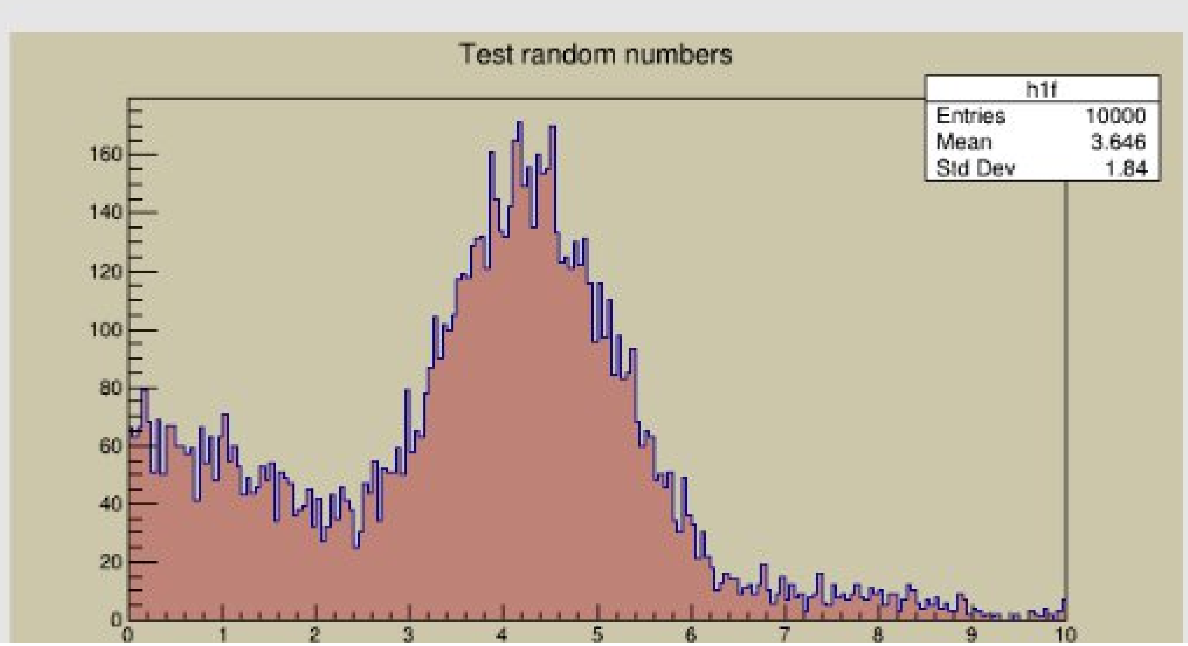
```
myfile = TFile( 'py-fillrandom.root', 'RECREATE' )
form1.Write()
sqroot.Write()
h1f.Write()
myfile.Close()
gBenchmark.Show( 'fillrandom' )
```

```
fillrandom: Real Time =    2.51 seconds Cpu Time =    0.43 seconds
```

Draw all canvases

In [5]:

```
from ROOT import gROOT
gROOT.GetListOfCanvases().Draw()
```

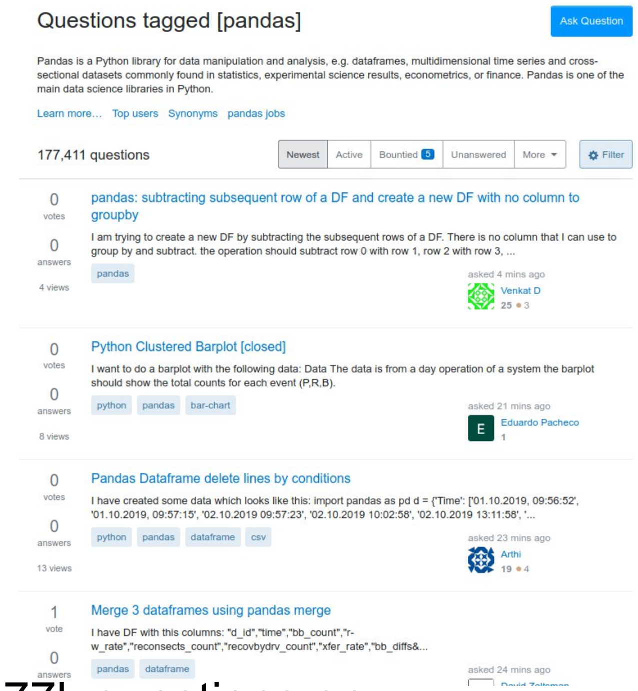


How does it compare with Matplotlib/Pandas/... ?

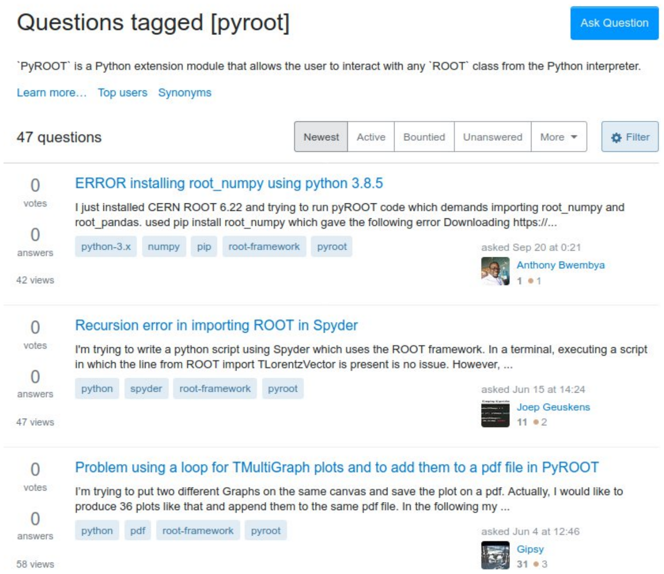
N.B. What follows is personal, subjective, disputable...

How does it compare with Matplotlib/Pandas/...

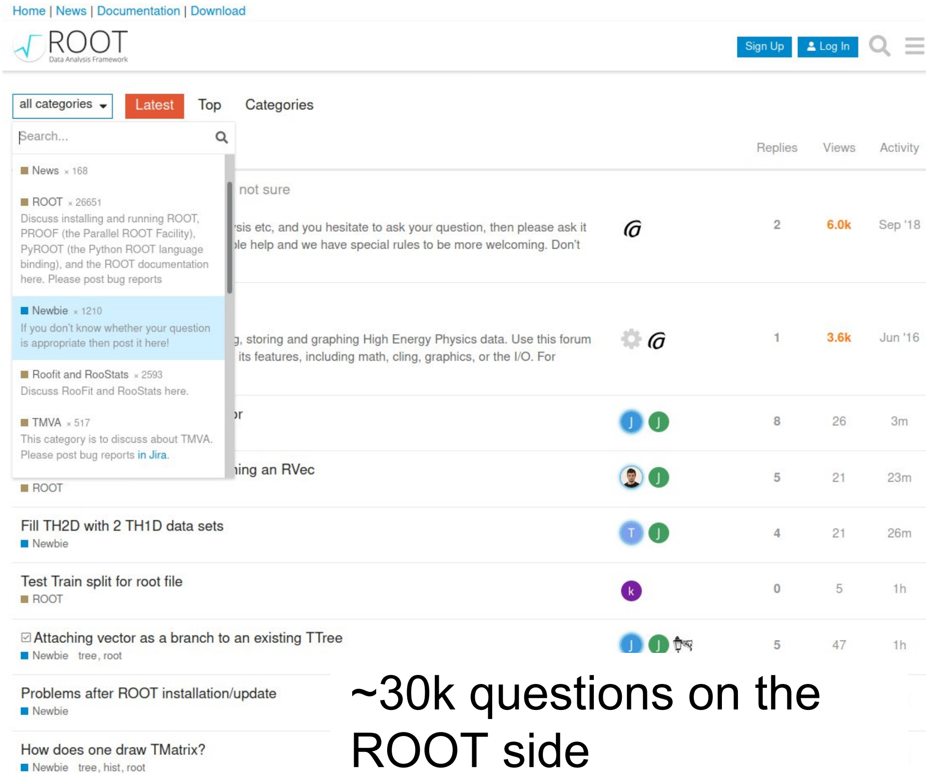
- Lots of interest on new packages (typically python based) also in HEP
  - New active communities, cross fertilisation
  - Difficult to compare. Personally I believe that the adoption numbers of ROOT are impressive (after all HEP is a small community and a special case of computing, analytics, etc...)
  - Balancing the usage of standard tools with a specialised one probably best



177k questions on Pandas (StackOverflow)



47 questions on pyRoot (StackOverflow)



~30k questions on the ROOT side


S MB 2220

How does it compare with Matplotlib/Pandas/...

- Often I complain of inconsistencies in ROOT (as relics from the past, implicit stuff that can be difficult to guess)
- Integration is always painful

- But consider this example from SciPy

7.2.1 Fit with a Predefined Function

To fit a histogram with a predefined function, simply pass the name of the function in the first parameter of `TH1::Fit`. For example, this line fits histogram object `hist` with a Gaussian.

```
root[] hist.Fit("gaus");
```

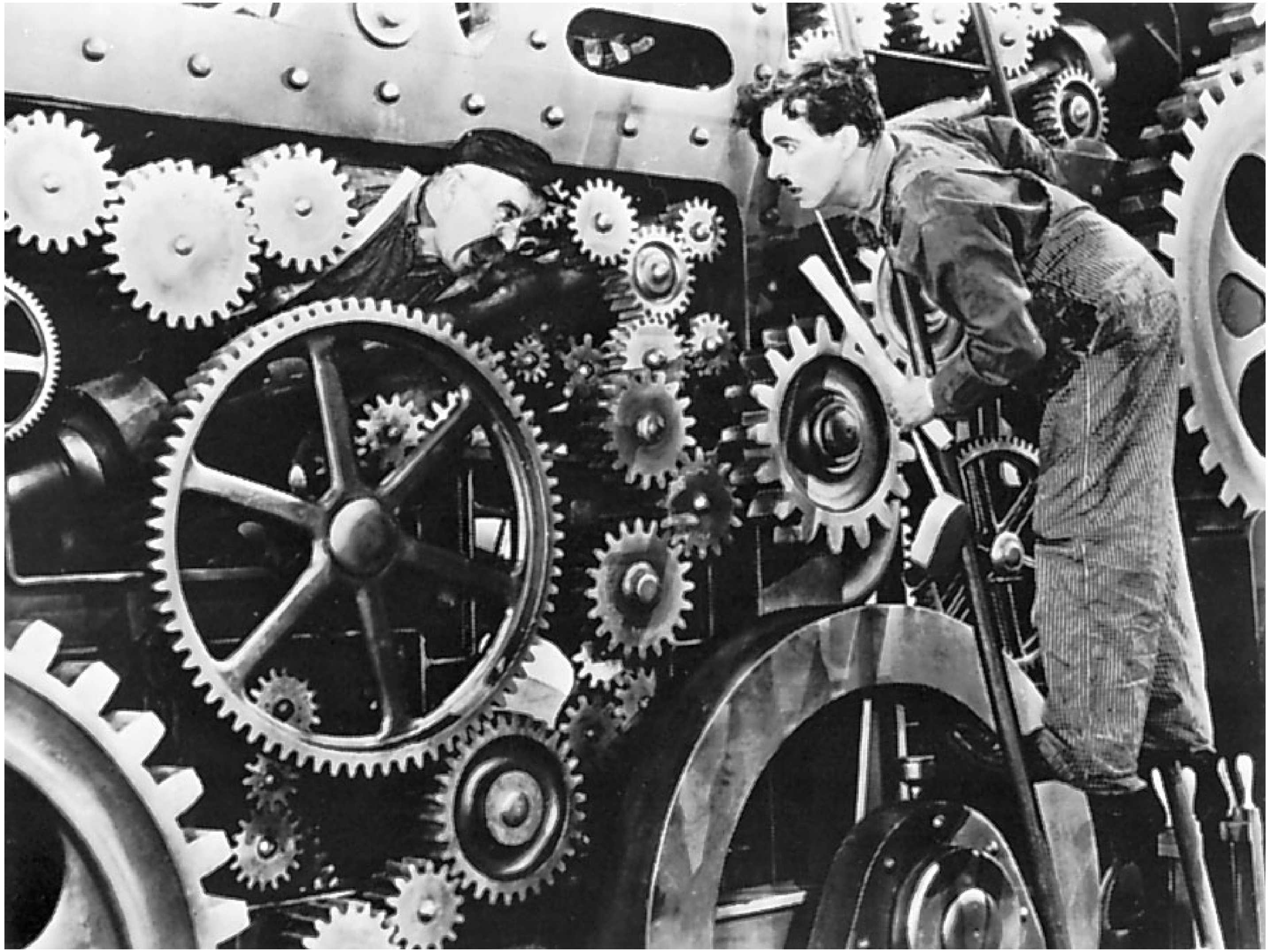
The initial parameter values (and eventual limits) for pre-defined functions are set automatically. For overriding the default limits values use the fit option `B`.

The list of pre-defined functions that can be used with the `Fit` method is the following:

- "`gaus`" Gaussian function with 3 parameters: `f(x) = p0*exp(-0.5*((x-p1)/p2)^2)`
- "`expo`" An Exponential with 2 parameters: `f(x) = exp(p0+p1*x)`
- "`pol` *N*" A polynomial of degree *N*, where N is a number between 0 and 9: `f(x) = p0 + p1*x + p2*x2 +...`
- "`chebyshev` *N*" A Chebyshev polynomial of degree *N*, where N is a number between 0 and 9: `f(x) = p0 + p1*x + p2*(2*x2-1) +...`
- "`landau`" Landau function with mean and sigma. This function has been adapted from the `CERNLIB` routine `G110 denlan` (see `TMath::Landau`).
- "`gausn` Normalized form of the gaussian function with 3 parameters `f(x) = p0*exp(-0.5*((x-p1)/p2)^2)/(p2 *sqrt(2PI))`

```
def smart_linregress(x,y,minx=None,maxx=None): # How come stats.lingres does not handle NAN gracefully?!
    xx = []
    yy = []
    for i,j in zip(x,y):
        if np.isnan(i): continue
        if np.isnan(j): continue
        if minx and i<minx: continue
        if maxx and i>maxx: continue
        xx.append(i)
        yy.append(j)

    gradient, intercept, r_value, p_value, std_err = stats.linregress(xx,yy)
    return gradient, intercept, r_value, p_value, std_err
```



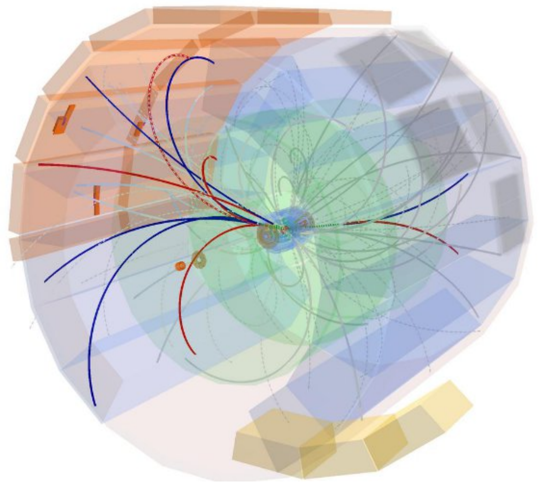
How does it compare with Matplotlib/Pandas/...

- Parallel processing
  - Pandas probably does it right (in the sense it pushes you to produce terse code)

  - Histo filling is a perfect map-reduce example (file-based parallelism; batch processing)

```
# pseudo code (non declarative)

for ti in tracks:
   for tj in tracks:
      if ti==tj: continue
      if ti.charge()==tj.charge(): continue
      if not ti.interesting(): continue
      vxyz = vertex(i,j)
```



How does it compare with Matplotlib/Pandas/...

```
# Load the Pandas libraries with alias 'pd'
import pandas as pd

# Read data from file 'filename.csv'
# (in the same directory that your python process is based)
# Control delimiters, rows, column names with read_csv (see later)
data = pd.read_csv("filename.csv")

# Preview the first 5 lines of the loaded data
data.head()
```

```
root [0] auto *file1 = new TFile("histograms.root"); # Open an histo file
root [1] file1->ls()
TFile**          histograms.root
 TFile*          histograms.root
  KEY: TH1F      hComplete;1
  KEY: TCanvas   cComplete;1      Distribution of all the topologies
root [2]
```

Since I am nasty, I picked up the C++ ROOT :)
I am *not* a fan of C++ syntax

Nice integration example (ROOT + numpy + matplotlib + seaborn)

Get the fit parameters and plot in Python

Even though the ROOT offers plenty of functions to style your plots for presentations and publications I usually prefer the style from Python. Importing the seaborn package also gives you a good starting point style wise and also some extra functions for plotting.

In [4]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.ion()

#Activate seaborn styling
sns.set()
sns.set_context('talk', font_scale = 1.2)
sns.set_style('white')


#Function to calculate y
def pol2(x, p0, p1, p2):
    return p0+p1*x+p2*x**2

#Get fit parameters and generate a label
```

Minimal plotting example using PyROOT

First off we start with a simple example to plot some numpy data using PyROO
is straight forward but we need to watch out with the data type that we send to

In [1]:

```
import numpy as np
from ROOT import TCanvas, TGraph

#Some data
x = np.arange(10)
y = x**2
```

Fit the data

ROOT has lots of build in functions but for this example we will define a custom function using the TF1 class.

In [3]:

```
#First we need a function to fit
from ROOT import TF1

#Use a custom function (altough the build in pol2 would also work)
func = TF1('func', '[0] + [1]*x + [2]*x**2', 0, 10)
fit = g.Fit('func', 'S')

c.Draw()
g.Draw('AP')
```

http://www.erikfrojdh.com/python/plot-and-fit-in-root-unsing-pyroot-and-then-back-to-python/

How does it compare with Matplotlib/Pandas/...

- Importance of data structures
  - DF in Pandas!

- Initially some of the specialised data structure were needed to shield from the simplistic (...) backend
  - In reality data sciences means data (a lot of)
  - In-memory analysis is OK but up to a point...

    'Cause when the going gets tough... the tough gets going! Who's with me? (John Belushi, circa 1978)

- In reality the available data structures guide you in the usage of the system and your analysis
  - Promote parallelism
  - Simpler code and faster execution
  - Transparent execution without handling memory (RAM) limits

Examples (if time allows)

- Simple histo example:
  - https://swan001.cern.ch/user/laman/notebooks/SWAN_projects/Students/histobasic.ipynb

- CMS muon data (opendata in CSV + python)
  - https://swan001.cern.ch/user/laman/notebooks/SWAN_projects/Students/cms_muons_starting_point.ipynb
  - 15 MB 0.1 M events ("records", "lines")

- "Realistic" input data size
  - https://swan001.cern.ch/user/laman/notebooks/SWAN_projects/Students/Untitled.ipynb
  - 2 GB 60 M events