Hands-on Session

Introduction to ROOT

Data Analysis Framework

& GIT

(Based on the introduction by A. Calandri, L. Gerritzen & L. Vigani)

Tamasi Kar

Physikalisches Institut, Universität Heidelberg

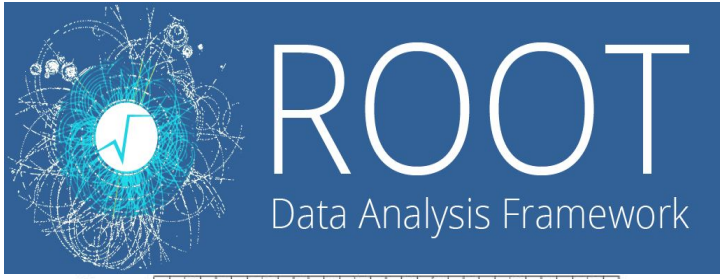
For the PSI Practical Course, Sep. 29 2023

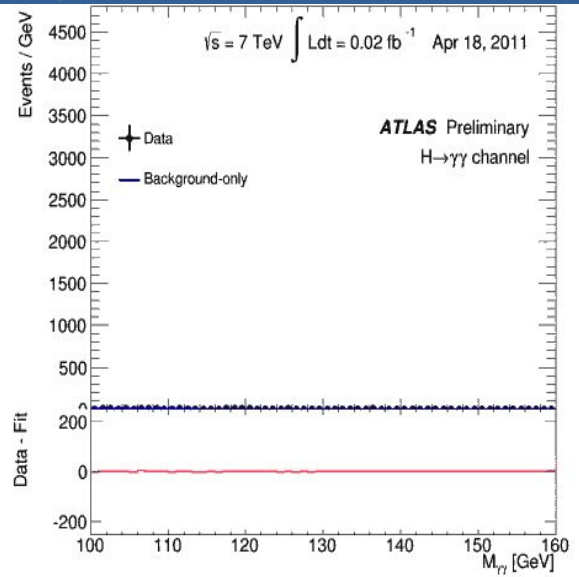
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

ETH zürich

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

What will we do in this hands-on session?





Outline

For this lecture/hands-on session

- Why ROOT?
  success stories, examples...
- General features of ROOT language
- Introduction to ROOT classes: functions, histograms, fits, graphs, trees
- A brief overview on GIT for software version control
- Tutorials
- Resources for further reference

ROOT Data Analysis Framework

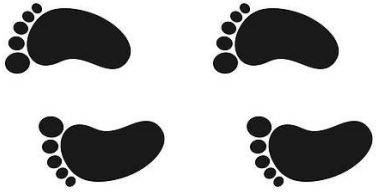
- ROOT is the Swiss Army Knife of High Energy Physics
- But it doesn't look like this

*Swiss army knife idea taken from here

ROOT Data Analysis Framework

- ROOT is the Swiss Army Knife of High Energy Physics
- But it rather looks like this

The goal of this hands-on session is to help you to take your first steps into the ROOT Jungle



*Swiss army knife idea taken from here

Let’s Dive into ROOT6

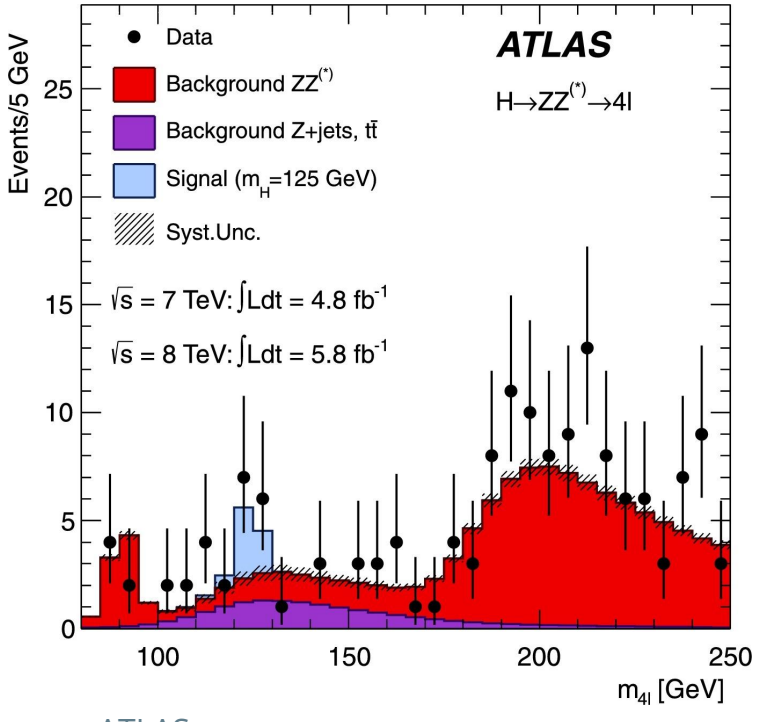
ROOT Data Analysis Framework

- ROOT is a free software developed at CERN that is used extensively in particle physics

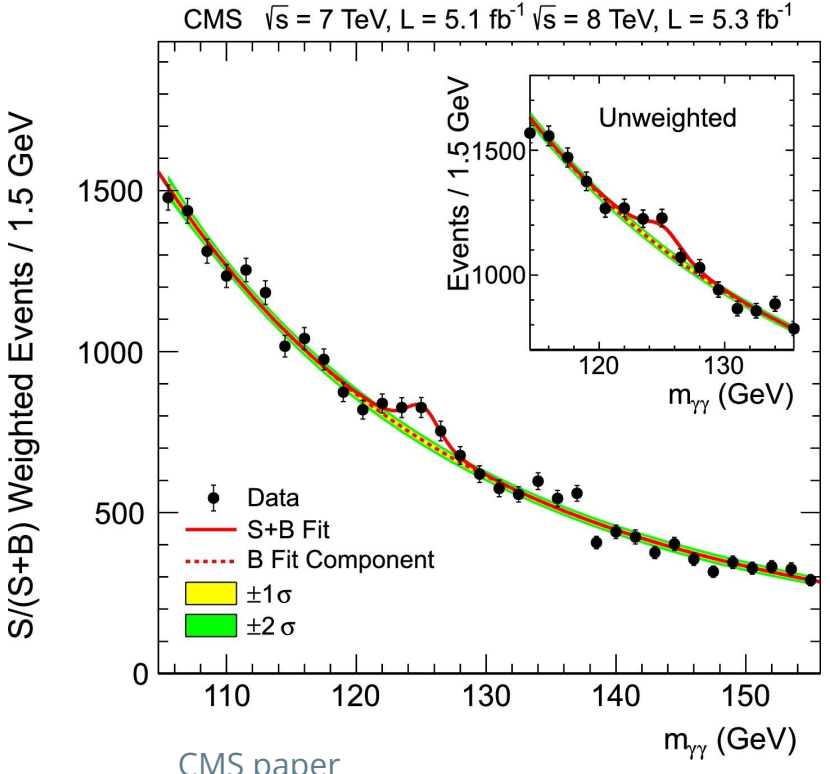
- Three main aspects are:
  - ➢ Plotting / Graphics: histograms, graphs, functions
  - ➢ Data Analysis: math, statistical libraries
  - ➢ Data Storage: data structures for event-based analysis

ROOT Plots

The **Higgs Boson** was **discovered** using a ROOT plot in 2012...

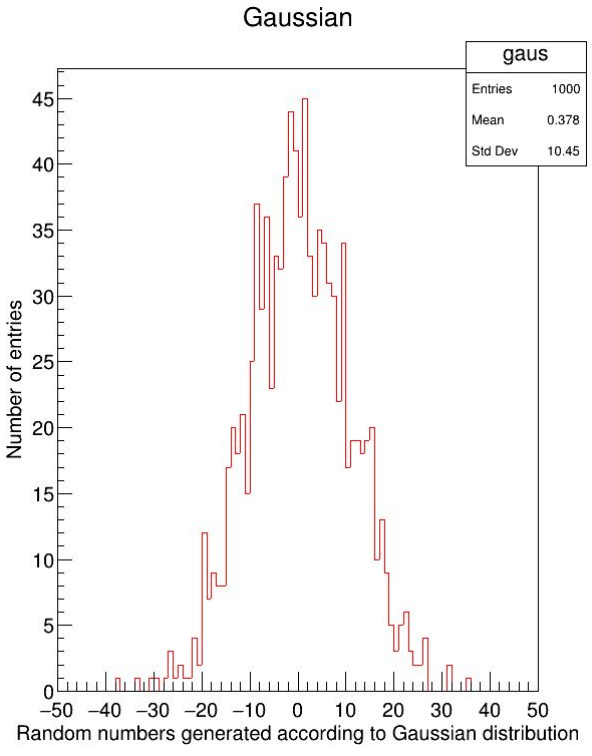


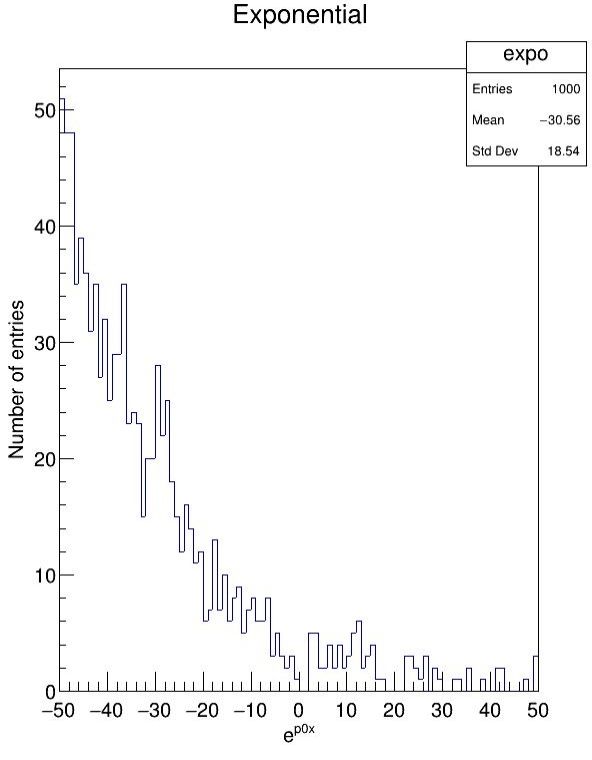
ATLAS paper

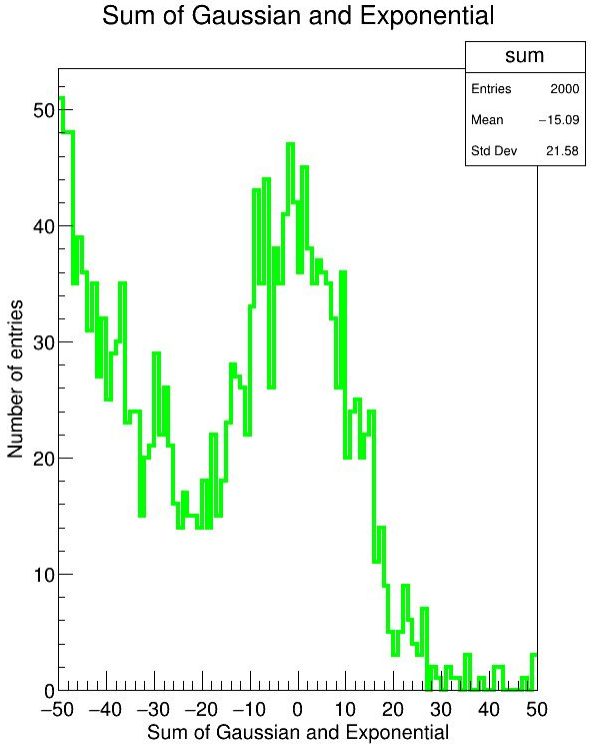


CMS paper

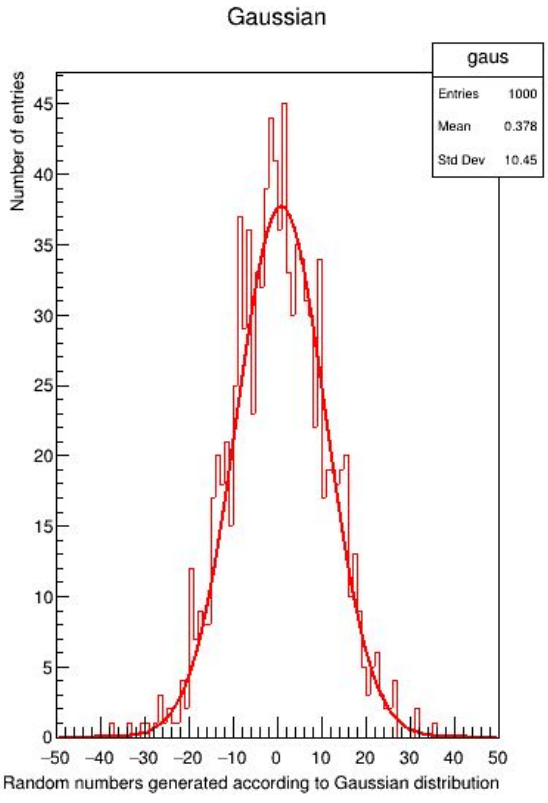
Histograms

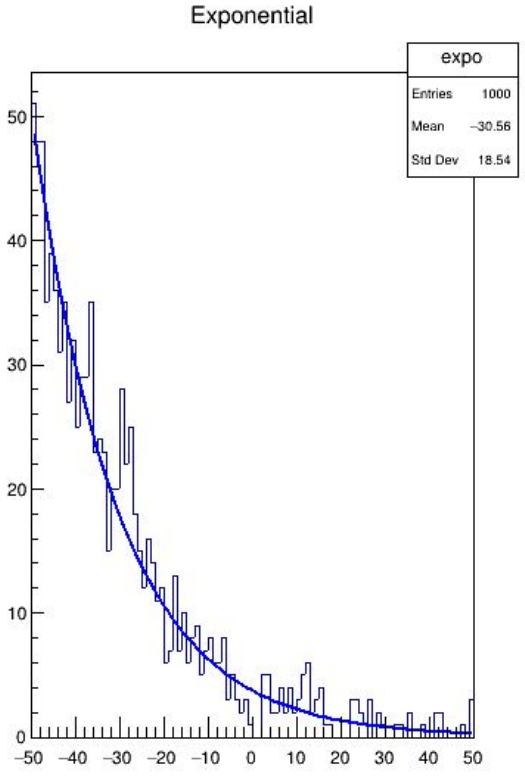

Gaussian
gaus
Entries 1000
Mean 0.378
Std Dev 10.45
Number of entries
Random numbers generated according to Gaussian distribution

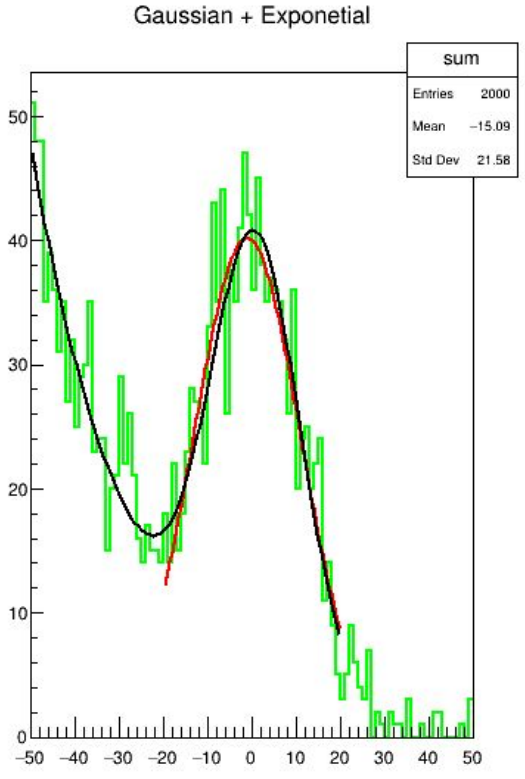

Exponential
expo
Entries 1000
Mean −30.56
Std Dev 18.54
Number of entries
e^p0x


Sum of Gaussian and Exponential
sum
Entries 2000
Mean −15.09
Std Dev 21.58
Number of entries
Sum of Gaussian and Exponential

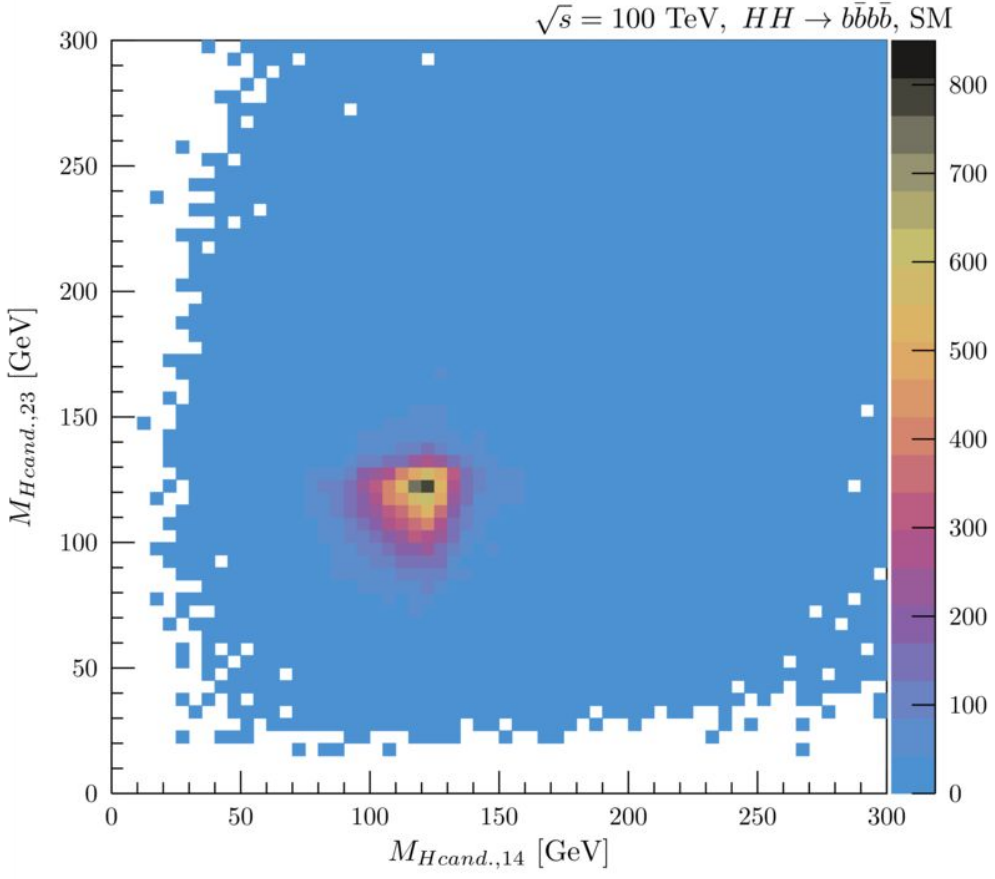
Fitting functions







Scatter / Correlation plots



Scatter plot for the invariant masses of the two reconstructed Higgs candidates, ref: T.Kar thesis

Getting started with ROOT

- ROOT interface is prompt based and speaks **C++**
- root -l → w/o loading splash screen
- root -b → run in batch mode (w/o graphics)

You can use the ROOT prompt like any other prompt / command line

➢ Simple computations

```
[kar@chameleon data]$ root -l
root [0] 1+2
(int) 3
root [1] 2*(1+2)/3.
(double) 2.0000000
root [2] sqrt(2.)
(double) 1.4142136
root [3] TMath::Pi()
(double) 3.1415927
root [4] TMath::TwoPi()
(double) 6.2831853
```

```
[kar@chameleon data]$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.22/06                          https://root.cern |
  | (c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Nov 27 2020, 15:14:08                 |
  | From tags/v6-22-06@v6-22-06                                      |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------
```

Getting started with ROOT

- ROOT interface is prompt based and speaks **C++**
- root -l → w/o loading splash screen
- root -b → run in batch mode (w/o graphics)

You can use the ROOT prompt like any other prompt / command line

- ➢ Simple computations
- ➢ Multi-line commands (better to use macros)
- ➢ Bash commands can be used with **.!**

```
[kar@chameleon work]$ root -l
root [0] .!pwd
/user/tkar/work
root [1] .!ls -t
fastjet  my_analysis  test_genetators  repos
root [2] .!echo "Welcome to ROOT"
Welcome to ROOT
root [3] .!echo "To quit root do .q"
To quit root do .q
root [4] .q
[kar@chameleon work]$
```

← Quit ROOT

```
[kar@chameleon work]$ root -l
root [0] for(int i = 0; i < 4; i++){cout<<"i = " << i <<endl;} cout<<"Line 2 in the ROOT prompt\n";
i = 0
i = 1
i = 2
i = 3
Line 2 in the ROOT prompt
```

ROOT macros and libraries

- Useful to create macros

combine lines of code for simple data analysis scripts - files will be interpreted by the C++ interpreter (CLING in case of ROOT6)

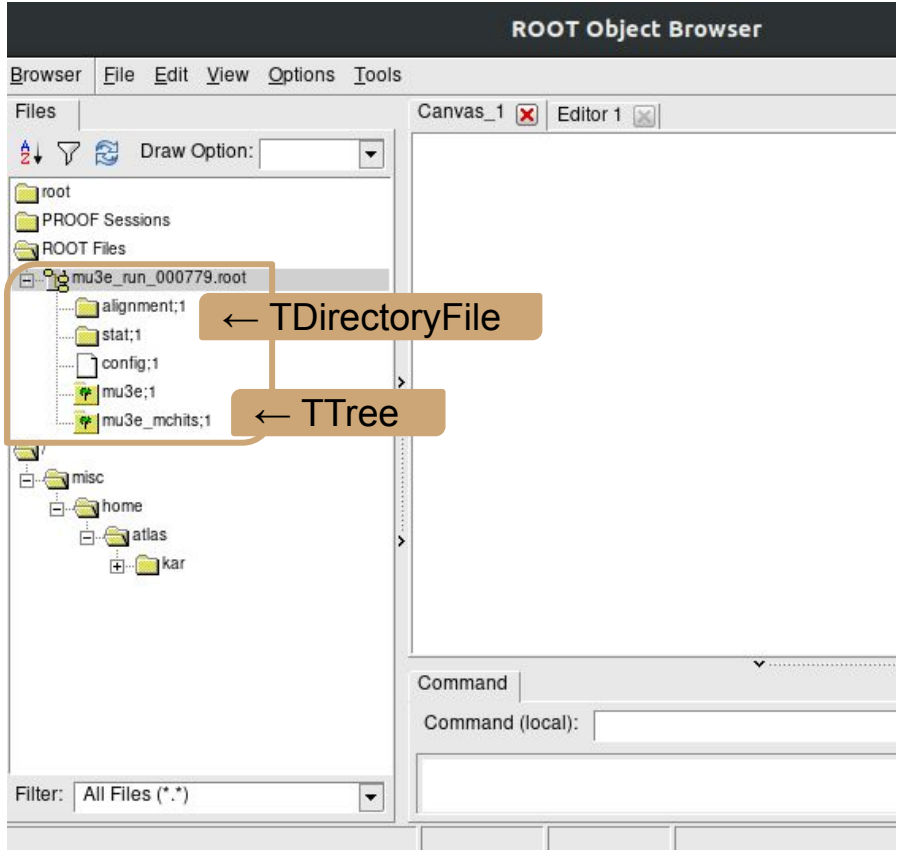
```
// Example macro
void ExampleMacro(string name = "Tamasi")
{
  cout<< "Hi " << name << "! Below are 5 random numbers."<<endl;
  TRandom r;
  for(int i = 0; i < 5; i++)
  {
    cout<< r.Rndm() <<endl;
  }
  return;
}
```

```
[kar@chameleon work]$ root -l ExampleMacro.C
root [0]
Processing ExampleMacro.C...
Hi Tamasi! Below are 5 random numbers.
```

```
root [0] .L ExampleMacro.C
root [1] ExampleMacro("Luigi")
Hi Luigi! Below are 5 random numbers.
0.1543
0.423245
0.7906
0.246387
0.193708
root [2] .x ExampleMacro.C
Hi Tamasi! Below are 5 random numbers.
0.1543
0.423245
0.7906
0.246387
0.193708
root [3]
```

ROOT data format & TBrowser

```
[kar@chameleon data]$ root -l mu3e_run_000779.root
root [0]
Attaching file mu3e_run_000779.root as _file0...
(TFile *) 0x2c0c7b0
root [1] TBrowser m
(TBrowser &) Name: Browser Title: ROOT Object Browser
root [2] _file0->ls()
TFile**         mu3e_run_000779.root
 TFile*         mu3e_run_000779.root
  KEY: TDirectoryFile   alignment;1     alignment
  KEY: TDirectoryFile   stat;1  stat
  KEY: TObjString       config;1        Collectable string class
  KEY: TTree    mu3e;1  mu3e
  KEY: TTree    mu3e_mchits;1   mu3e_mchits
root [3] _file0->cd("alignment")
(bool) true
root [4] _file0->ls()
TFile**         mu3e_run_000779.root
 TFile*         mu3e_run_000779.root
  TDirectoryFile*               alignment       alignment
   KEY: TDirectoryFile  target;1        target
   KEY: TTree   sensors;1       sensors
   KEY: TTree   tiles;1 tiles
   KEY: TTree   fibres;1        fibres
   KEY: TTree   mppcs;1 mppcs
  KEY: TDirectoryFile   alignment;1     alignment
  KEY: TDirectoryFile   stat;1  stat
  KEY: TObjString       config;1        Collectable string class
  KEY: TTree    mu3e;1  mu3e
  KEY: TTree    mu3e_mchits;1   mu3e_mchits
root [5]
```

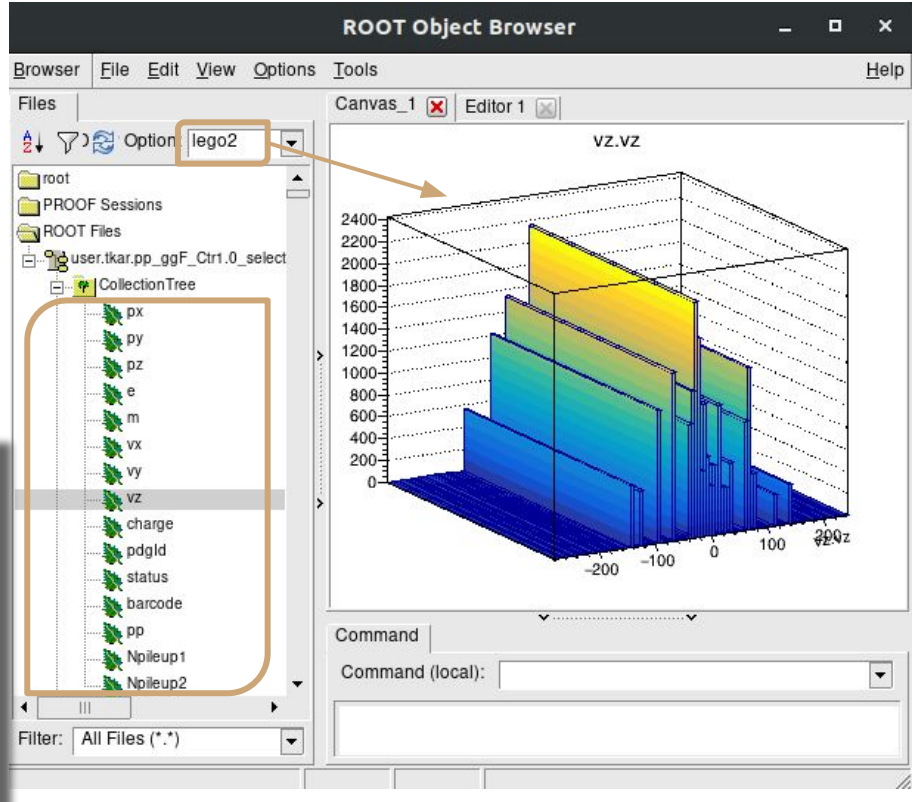


ROOT data format & TBrowser

```
root [1] _file0->ls()
TFile**         user.tkar.pp_ggF_Ctr1.0_selectedSignalEvents_000999.root
 TFile*         user.tkar.pp_ggF_Ctr1.0_selectedSignalEvents_000999.root
  KEY: TTree    CollectionTree;1        CollectionTree
root [2] CollectionTree->Print()
******************************************************************************
*Tree    :CollectionTree: CollectionTree
*Entries :       25 : Total =         1068007 bytes  File  Size =     442706 *
*        :          : Tree compression factor =   2.40                       *
******************************************************************************
*Br    0 :px        : vector<float>                                           *
*Entries :       25 : Total  Size=      81960 bytes  File Size  =      76935 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   1.06     *
*.....................................................                        *
*Br    1 :py        : vector<float>                                           *
*Entries :       25 : Total  Size=      81960 bytes  File Size  =      77002 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   1.06     *
*.....................................................                        *
*Br   11 :barcode   : vector<int>                                             *
*Entries :       25 : Total  Size=      81995 bytes  File Size  =      31224 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   2.61     *
*.....................................................                        *
*Br   12 :pp        : vector<int>                                             *
*Entries :       25 : Total  Size=      81960 bytes  File Size  =       1228 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=  66.35     *
*.....................................................                        *
*Br   13 :Npileup1  : Npileup1/F                                              *
*Entries :       25 : Total  Size=        682 bytes  File Size  =        106 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.75     *
*.....................................................                        *
*Br   14 :Npileup2  : Npileup2/F                                              *
*Entries :       25 : Total  Size=        682 bytes  File Size  =        106 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.75     *
*.....................................................                        *
```

vectors

floats



Basic classes in ROOT

- **TObject:** base class for all ROOT objects
- **TMath:** class for math routines
- **TRandom3:** random generator class
- **TF1:** base class for functions
- **TH1:** base class for 1-, 2-, 3-D histograms
- **TStyle:** class for style of histograms, axis, title, markers, etc...
- **TFile:** class for reading/writing root files
- **TTree:** basic storage format in ROOT
- **TGraph**: class of graphic object based on x and y arrays
- **TCanvas:** class for graphical display
- **TBrowser:** browse your files

Complete List of classes: https://root.cern/doc/master/classes.html

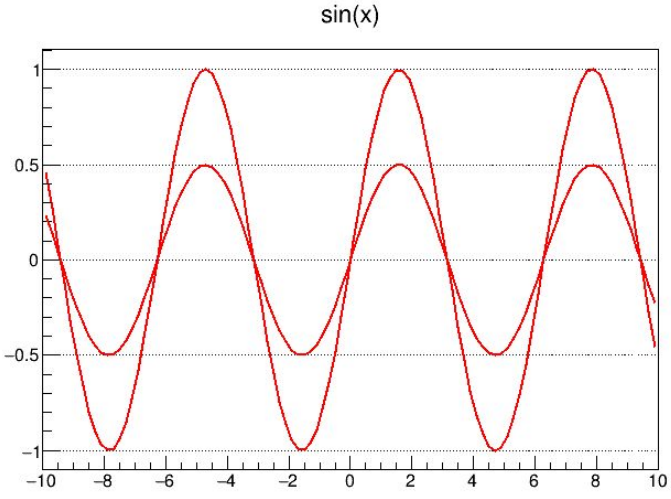
Functions

- TF1, TF2, TF3 for 1-D to 3-D functions…
- e.g. TF1 creation for c*sin(x)

```
root [0] f1 = new TF1("f1","sin(x)",-10,10)
(TF1 *) 0x563f64ce6460
root [1] f1->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2] f2 = new TF1("f2","[0]*sin(x)",-10,10)
(TF1 *) 0x563f65c93320
root [3] f2->SetParameter(0,0.5)
root [4] f2->Draw("same")
root [5] c1->SetGridy()
```

default canvas

Function with one parameter



- Gaussian and exponential distributions…

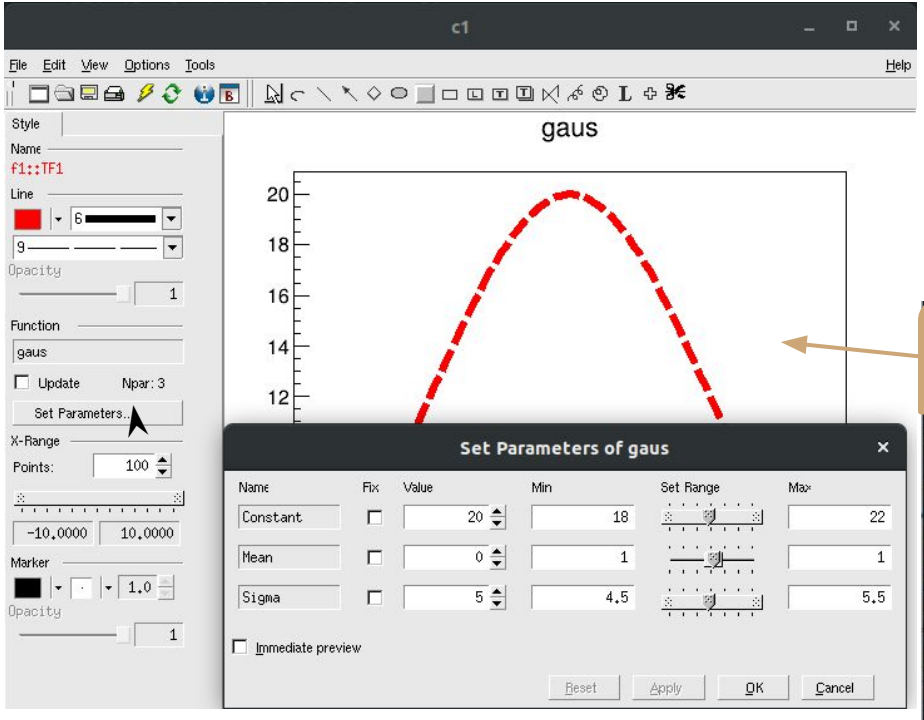
```
root [0] TF1 *f1 = new TF1("f1","gaus",-10,10)
(TF1 *) 0x5651679526d0
root [1] f1->GetFormula()->Print()
                f1 : gaus Ndim= 1, Npar= 3, Number= 100
 Formula expression:
       [Constant]*exp(-0.5*((x-[Mean])/[Sigma])*((x-[Mean])/[Sigma]))
```

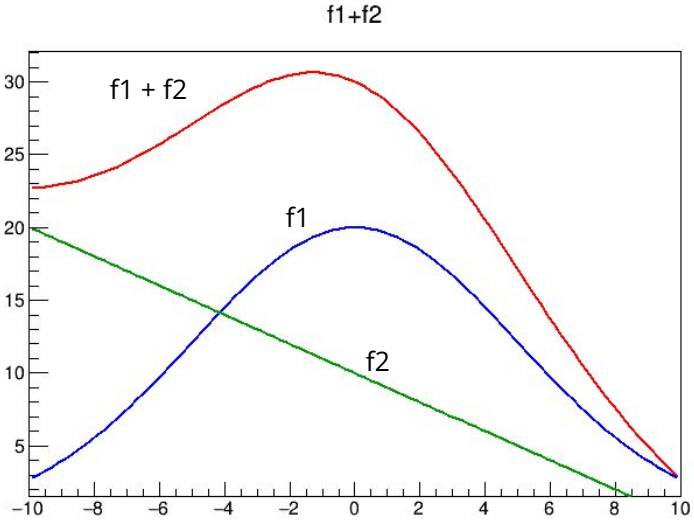
```
root [0] TF1 *f1 = new TF1("f1","expo",-10,10)
(TF1 *) 0x564647bee720
root [1] f1->GetFormula()->Print()
                f1 : expo Ndim= 1, Npar= 2, Number= 200
 Formula expression:
       exp([Constant]+[Slope]*x)
```

Many more pdfs:
https://root.cern.ch/doc/v610/group__PdfFunc.html

Functions

- TF1, TF2, TF3 for 1-D to 3-D functions…
- e.g. sum of a gaussian and linear distribution






```
root [0] TF1 *f1 = new TF1("f1","gaus",-10,10)
(TF1 *) 0x55f8f0448310
root [1] f1->SetParameters(20,0,5)
root [2] TF1 *f2 = new TF1("f2","10.-x",-10,10)
(TF1 *) 0x55f8f161f4d0
root [3] TF1 *f3 = new TF1("f3","f1+f2",-10,10)
(TF1 *) 0x55f8f195a8c0
root [4] f3->Draw("")
Info in <TCanvas::MakeDefCanvas>:  created default
root [5] f1->SetLineColor(kBlue)
root [6] f2->SetLineColor(kGreen+2)
root [7] f1->Draw("same")
root [8] f2->Draw("same")
```

1-D Histograms - TH1F/TH1D

- Binned data: backbone of all physics analyses and (likely) most important/used class in HEP data analysis
- Histogram creation: unique name, title + x/y-axes, number of bins, bin range

```
root [0] h1 = new TH1F("h1","title; Xaxis title; Yaxis title",10,0,10)
```

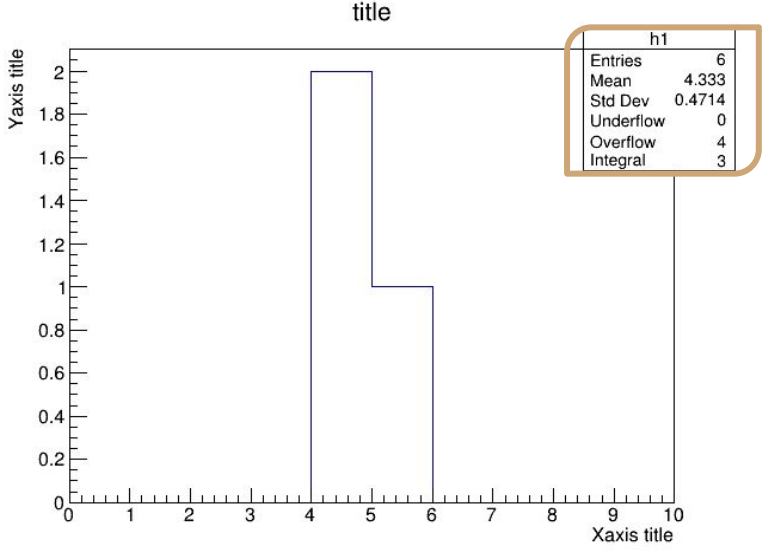
- A bin includes the lower limit but excludes upper limit
- TH1::Fill() method of the TH1 class fills histogram – events are weighted according to the weight
- TH1::Draw() method draws the histogram – several options available e.g. Draw("e"), Draw("hist")

```
root [1] h1->Fill(10)
(int) -1
root [2] h1->Fill(11)
(int) -1
root [3] h1->Fill(15)
(int) -1
root [4] h1->Fill(10)
(int) -1
root [5] h1->Fill(4,2)
(int) 5
root [6] h1->Fill(5,1)
(int) 6
root [7] h1->Draw("hist")
```

overflow bin

bin number

bin number

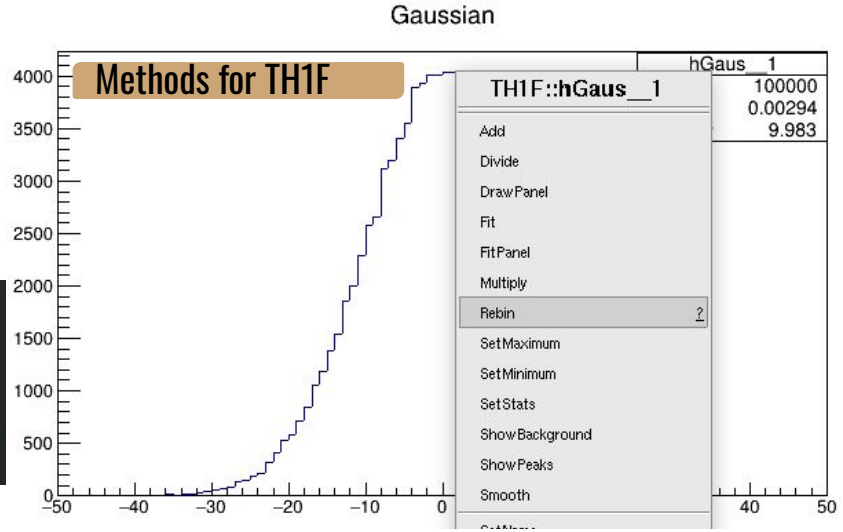


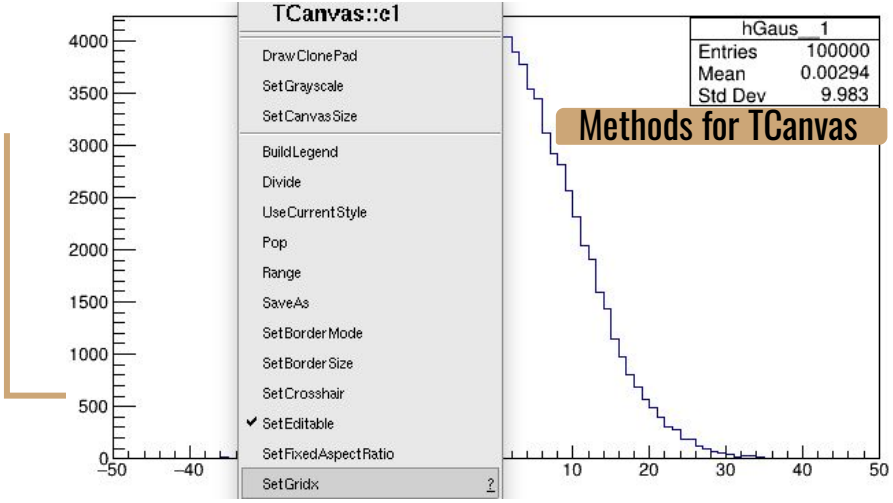
- TH1::DrawNormalized() draws histogram whose integral is normalized to unity

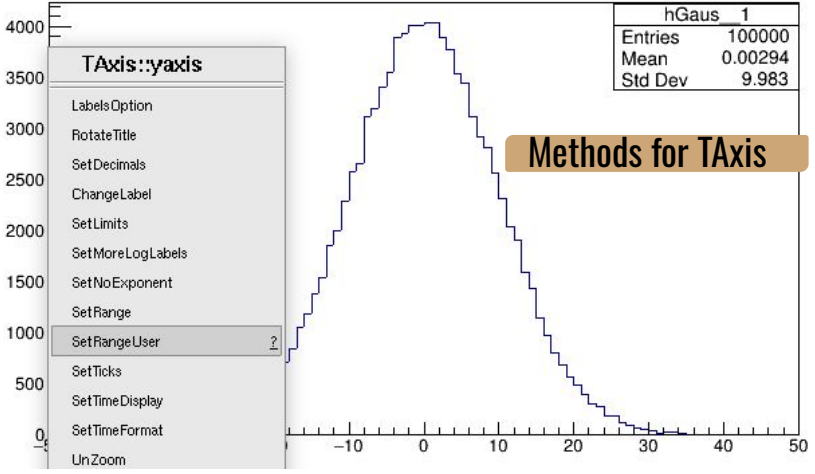
1-D Histograms - TH1F/TH1D

- Several functions available for data analysis inherit from TH1
- Useful to rebin histogram, change x/y axis range, set logarithmic scales on x/y axis...

```
root [0] TH1F *hGaus = new TH1F("hGaus","Gaussian",100,-50,50)
(TH1F *) 0x5616a6bd9eb0
root [1] TRandom3 *r3 = new TRandom3();
root [2] gRandom = r3;
root [3] for(int i = 0; i<100000; i++) {hGaus->Fill(r3->Gaus(0,10));}
root [4] hGaus->Draw("hist")
```





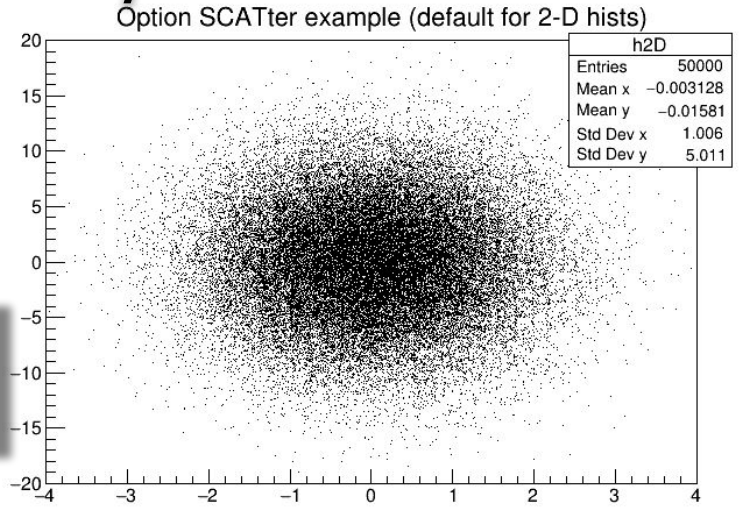


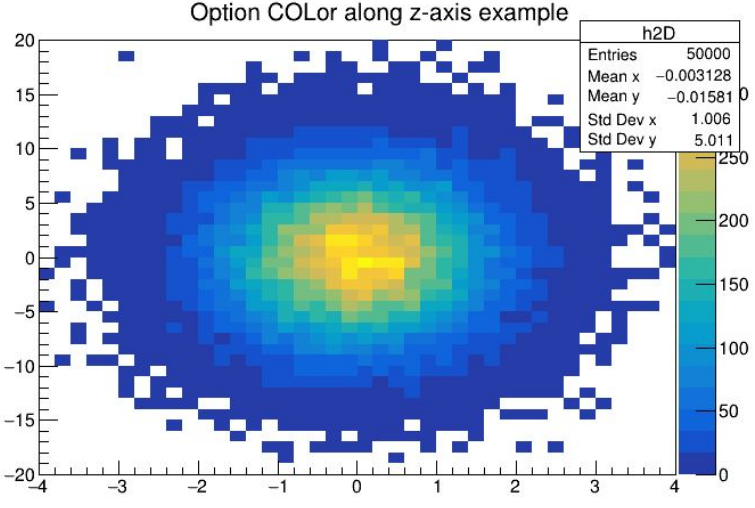
2-D Histograms - TH2F/TH2D

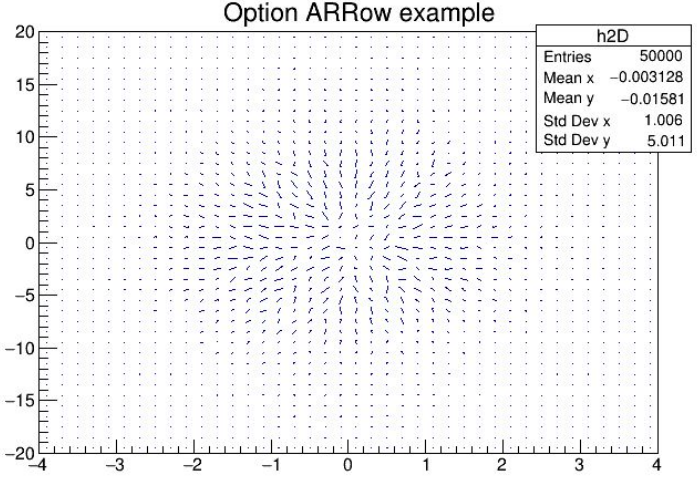
- Same concept as in TH1F but now in 2-D : useful to study simple linear correlations between observables

  – easily extendable to 3D case, several visualisations available

```
TH2F *h2D = new TH2F("h2D","2-D hist", 40,-4,4,40,-20,20);
TRandom3  *r3 = new TRandom3(0);
for(int i=0;i<5e4;i++){h2D->Fill(r3->Gaus(0,1),r3->Gaus(0,5));}
h2D->Draw("colz")
```



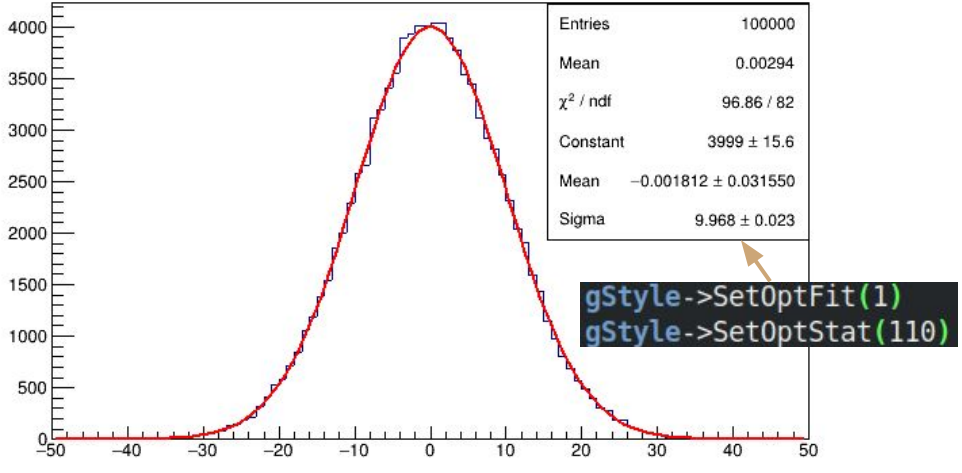


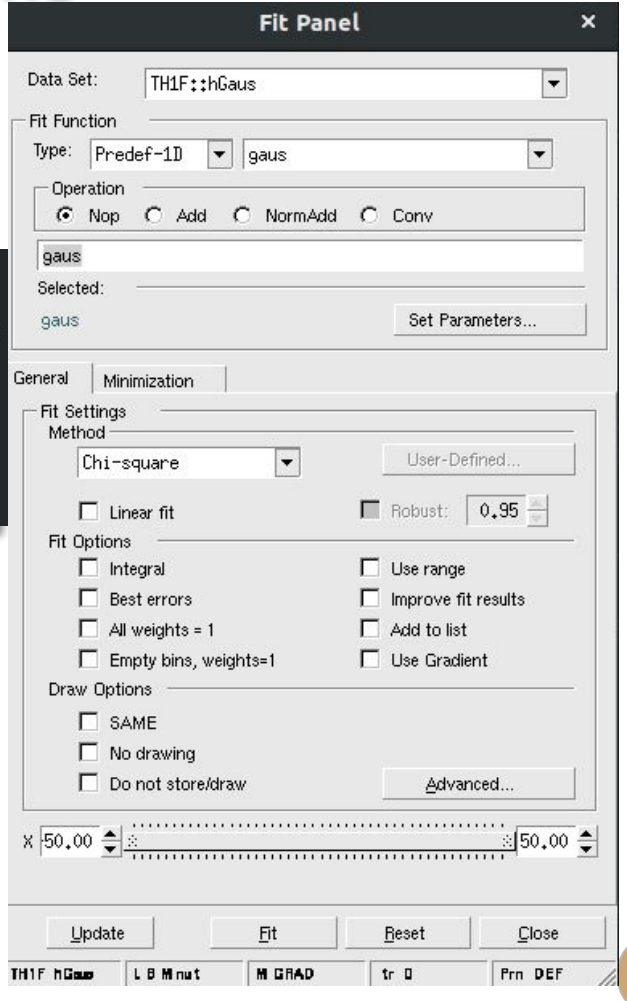


Fitting Histograms

- Simple interactive interface to perform quick fits on given histogram shapes
- Also possible to use prompt command h->Fit("gaus"), define fitting function (expo, landau,..) and the fit range

```
root [1] TFitEditor::DoFit - using function PrevFitTMP  0x55a93d5f5300
 FCN=96.8556 FROM MIGRAD    STATUS=CONVERGED      56 CALLS          57 TOTAL
                     EDM=8.27604e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     3.99857e+03   1.55632e+01   6.11097e-02  -9.17048e-06
   2  Mean        -1.81219e-03   3.15501e-02   1.52394e-04  -4.92108e-06
   3  Sigma        9.96799e+00   2.26156e-02   2.96719e-06  -1.83613e-01
```





Graphs- TGraph/TGraphErrors

- TGraph/TGraphErrors/TGraphAsymmErrors - objects containing discrete points without/with symmetric/asymmetric error bars

  – value of x/y points defined and input to TGraph object as well as the errors associated

  – TGraph::SetPoint() method accesses x/y coordinates of TGraph. Alternatively TGraph can be filled using C++ arrays as inputs

- Graph style can be accessed via command line or by clicking on the canvas (Editor and Toolbar)

```
TGraph *gr1 = new TGraph (n,x,y);
```

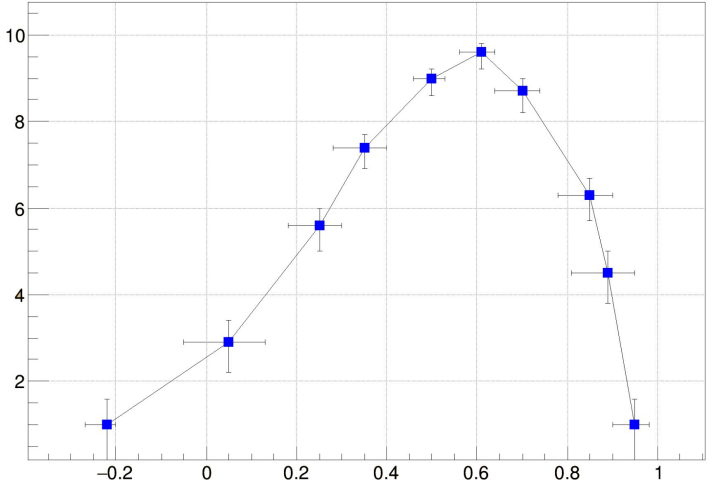
```
TGraphErrors *gr2 = new TGraphErrors (n,x,y,ex,ey);
```

```
TGraphAsymmErrors *gr2 = new TGraphAsymmErrors (n,x,y,ex1,exh,ey1,eyh)
```

Filling TGraphObject

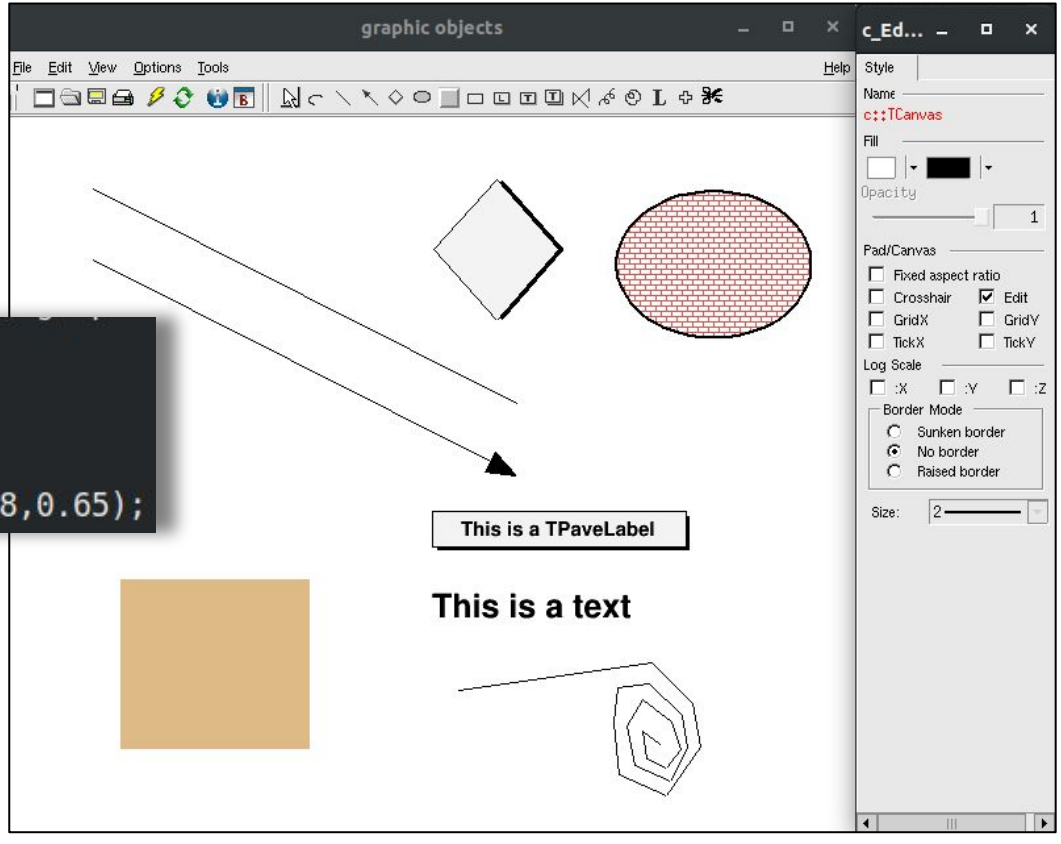
```
root [0] graph = new TGraph;
root [1] graph->SetPoint(graph->GetN(),1,10);
root [2] graph->SetPoint(graph->GetN(),2,15);
root [3] graph->Draw("AP");
Info in <TCanvas::MakeDefCanvas>:  created defau
root [4] graph->SetMarkerStyle(22);
root [5] graph->GetYaxis()->SetRangeUser(0,10);
root [6] graph->GetXaxis()->SetTitle("X-axis");
root [7] graph->GetYaxis()->SetTitle("Y-axis");
root [8] graph->Draw("AP");
root [9] graph->GetYaxis()->SetRangeUser(0,20);
root [10] graph->Draw("AP");
root [11] graph->Draw("ACP");
```

TGraphAsymmErrors Example



Other graphic objects

```
root [1] TLine line(.1,.9,.6,.6);
root [2] line.Draw();
root [3] TText text(.5,.2,"Hello");
root [4] text.Draw();
root [5] TPaveLabel tlabel(0.5,0.6,0.8,0.65);
```



ROOT Files – TFile

- Class TFile is used to store ROOT object on disk

```
root [0] f1 = TFile::Open("file.root","NEW");
root [1] f1->ls();
TFile**         file.root
 TFile*         file.root
root [2] h1 = new TH1F("h1","Example histo",10,0,10);
root [3] h1->Fill(5,3); h1->Fill(0.,2); h1->Fill(4,1);
```

UPDATE / RECREATE / READ

```
root [6] h1->Write();
root [7] f1->ls();
TFile**         file.root
 TFile*         file.root
  OBJ: TH1F     h1      Example histo
  KEY: TH1F     h1;1    Example histo
root [8] f1->Close()
```

- If you open the root file from outside the ROOT prompt: e.g. root -l file.root
  → ROOT opens the file with a default pointer _file0

```
root [0]
Attaching file file.root as _file0...
```

- To retrieve an object (TH1F in this case) from a file: use the method Get or GetObject

```
TH1F *h = 0; _file0->GetObject("h1", h);
```

ROOT Trees – TTree (Writing)

- TTree – supports large collection of objects
  - allows direct access to any entry
  - is the main data storage for HEP data analysis - can store basic types
- Trees are structured into branches and leaves

```
root [0] f = TFile::Open("events.root","RECREATE");
root [1] TTree *t = new TTree("Events","Event Tree");
```

Creating a new TTree in file events.root

```
root [2] Int_t var1;
root [3] Float_t var2;
root [4] Double_t var3;
root [5] t->Branch("var1", &var1, "var1/I");
root [6] t->Branch("var2", &var2, "var2/F");
root [7] t->Branch("var3", &var3, "var3/D");
```

Declaring local variables

Initializing the TTree using the TTree::Branch() method

```
root [8] var1 = 5; var2 = 3.1; var3 = 1e3;
root [9] t->Fill();
root [10] var1 = 2; var2 = 7.; var3 = 9.5e2;
root [11] t->Fill();
root [12] t->Write();
root [13] f->Close();
```

Filling the TTree using the TTree::Fill() method

ROOT Trees – TTree (Reading)

- Once the tree is produced by the analysis code and stored in a TFile object, its content can be accessed and read
- Accessing tree and link the branch with the TTree pointer (TTree::Get() & TTree::SetBranchAddress()) methods)

In a macro

```
Float_t lvar2; // local variable
t->SetBranchAddress("var2", &lvar2);
Int_t nentries = t->GetEntries();
for(Int_t i=0; i<nenetries; i++)
{
  t->GetEntry(i);
  cout<<lvar2<<endl;
}
```

```
root [0] TFile *f = new TFile("events.root","READ");
root [1] TTree *t = (TTree*)f->Get("Events");
root [2] t->Print();
******************************************************************************
*Tree    :Events    : Event Tree                                               *
*Entries :        2 : Total =            2079 bytes  File  Size =        751 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :var1      : var1/I                                                   *
*Entries :        2 : Total  Size=        562 bytes  File Size  =         81 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :var2      : var2/F                                                   *
*Entries :        2 : Total  Size=        562 bytes  File Size  =         81 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :var3      : var3/D                                                   *
*Entries :        2 : Total  Size=        578 bytes  File Size  =         89 *
*Baskets :        1 : Basket Size=      32000 bytes
*.............................................
```

```
root [5] t->Scan();
************************************************
*    Row   *      var1 *      var2 *      var3 *
************************************************
*        0 *         5 * 3.0999999 *      1000 *
*        1 *         2 *         7 *       950 *
************************************************
```

```
root [3] t->Show(1);
======> EVENT:1
 var1            = 2
 var2            = 7
 var3            = 950
root [4] t->Show(2);
Error in <TTree::Show()>: Cannot read entry 2 (entry does not exist)
```

A brief introduction to software version control using GIT

What is version control?

Practice of tracking and managing changes to your software code over time

Version Control Systems (VCS):

- software tools that keep track of every modification to the code in a database.
- **Git** was originally developed by the creator of the Linux OS, Linus Torvalds.
- Git is a **distributed VCS** that is free and and actively maintained **open source** project.
- Has been designed with performance, security and flexibility in mind.

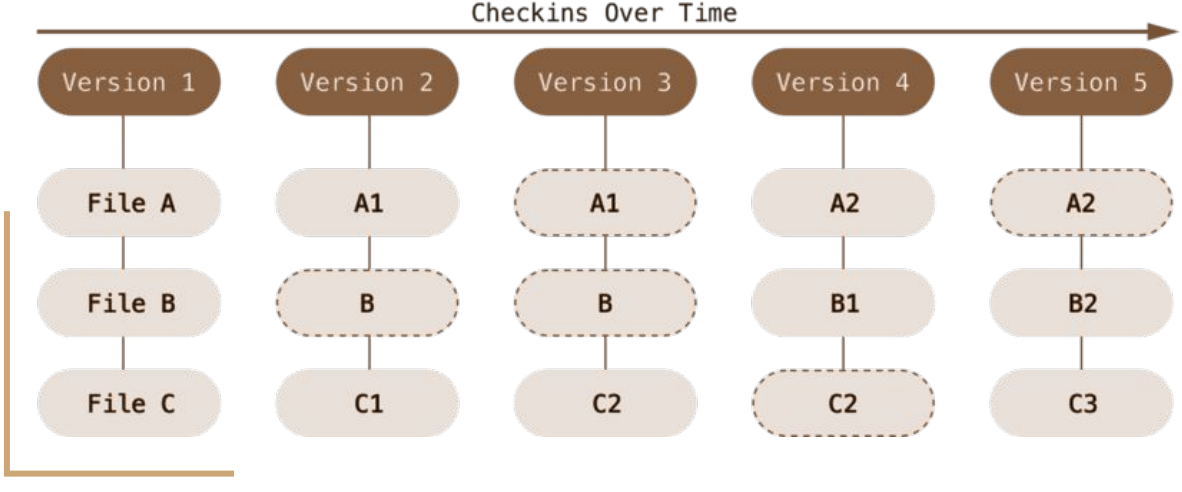
Many version control tools/**platforms** that offer git repository management:

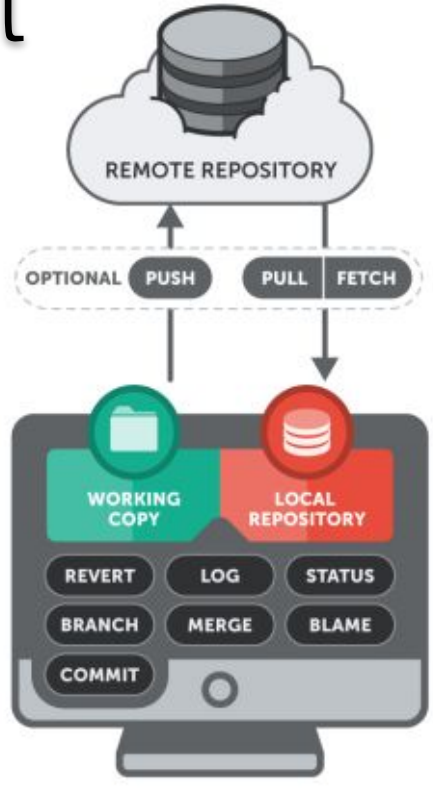
→ GitHub, Bitbucket, GitLab

Source: https://www.atlassian.com/git/tutorials/what-is-version-control

Software version control - Git

- Stores snapshots and not differences
- Git has distributed architecture: many copies of the repository out there
- Nearly every operation in Git is **local**; i.e. all version control actions can be done **offline**
- **Online** access is needed only to **share** your work with others and **get** the changes introduced by others





Snapshots

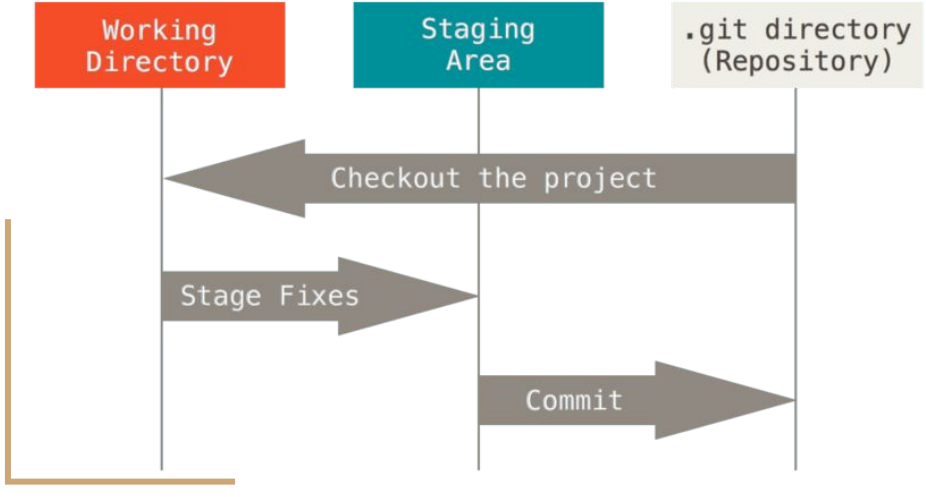
file — 'Link' to a unchanged file from a previous version

file — Changed file from a new version

Software version control - Git

Three main states your files can reside in:

- **Modified** means that you have changed the file but have not committed it to your database yet.
- **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot
- **Committed** means that the data is safely stored in your local database.



<u>You will typically:</u>

- **Clone** repository from gitlab
- **Create** your own branch (recommended)
- Work in **local workspace** (normal files on your pc)
- **Add** changes & **commit** changes (still in your local version history)
- **Push** commit(s) to remote repository (gitlab)

Git cheat sheet

To be done only once

Git configuration

```
$ git config --global user.name "Your Name"
```
Set the name that will be attached to your commits and tags.

```
$ git config --global user.email "you@example.com"
```
Set the e-mail address that will be attached to your commits and tags.

```
$ git config --global color.ui auto
```
Enable some colorization of Git output.

Starting A Project

```
$ git init [project name]
```
Create a new local repository. If **[project name]** is provided, Git will create a new directory name **[project name]** and will initialize a repository inside it. If **[project name]** is not provided, then a new repository is initialized in the current directory.

```
$ git clone [project url]
```
Downloads a project with the entire history from the remote repository.

Day-To-Day Work

```
$ git status
```
Displays the status of your working directory. Options include new, staged, and modified files. It will retrieve branch name, current commit identifier, and changes pending commit.

```
$ git add [file]
```
Add a file to the **staging** area. Use in place of the full file path to add all changed files from the **current directory** down into the **directory tree**.

```
$ git diff [file]
```
Show changes between **working directory** and **staging area**.

```
$ git diff --staged [file]
```
Shows any changes between the **staging area** and the **repository**.

```
$ git checkout -- [file]
```
Discard changes in **working directory**. This operation is **unrecoverable**.

```
$ git reset [file]
```
Revert your **repository** to a previous known working state.

```
$ git commit -m "[descriptive message]"
```
Create a new **commit** from changes added to the **staging area**. The **commit** must have a message!

source: https://about.gitlab.com/images/press/git-cheat-sheet.pdf

Git cheat sheet

Browse and inspect the evolution of project files

$ git log
Lists version history for the current branch

$ git log --follow [file]
Lists version history for a file, including renames

Synchronize changes

Synchronize your local repository with the remote repository on GitHub.com

$ git fetch
Downloads all history from the remote tracking branches

$ git merge
Combines remote tracking branch into current local branch

$ git push
Uploads all local branch commits to GitHub

$ git pull
Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. `git pull` is a combination of `git fetch` and `git merge`

Git branching model

$ git branch [-a]
List all local branches in repository. With **-a**: show all branches (with remote).

$ git branch [branch_name]
Create new branch, referencing the current **HEAD**.

$ git checkout [-b][branch_name]
Switch **working directory** to the specified branch. With **-b**: Git will create the specified branch if it does not exist.

$ git merge [from name]
Join specified **[from name]** branch into your current branch (the one you are on currently).

$ git branch -d [name]
Remove selected branch, if it is already merged into any other. **-D** instead of **-d** forces deletion.

source: https://training.github.com/downloads/github-git-cheat-sheet.pdf

Git take aways

Things to keep in mind

Pull = fetch+merge

- Pull before you push
  Otherwise you will end up with merge conflicts
- Write meaningful commit messages (“Fixed a bug” is not meaningful)
- Commit regularly
- Work on your branch and merge master into your branch regularly
- Don’t break the master branch

Hands-on (ROOT)

- git clone https://gitlab.rlp.net/psi2023/rootIntro.git

- Web-based interface: https://gitlab.rlp.net/psi2023/rootIntro

- Three ROOT example exercises mostly focussing on most important ROOT classes and tools discussed in this lecture/tutorial
  RootExampleMacro_ex1_students.C
  RootExampleMacro_ex2_students.C
  RootExampleMacro_ex3_students.C

Note: Output file produced by RootExampleMacro_ex2_students.C will be used as an input to the RootExampleMacro_ex3_students.C

- Walk through the three ROOT exercises- understand the code and go through the comments provided for each section of the macros.

Hands-on (GIT)

- Instructions to familiarise with git :

  https://gitlab.rlp.net/psi2023/gitplayground

Resources

For further reference

- **ROOT Website:** https://root.cern
- **Introduction material:** https://root.cern/get_started/ (Includes a booklet for beginners: the “ROOT Primer”)
- **Reference Guide:** https://root.cern/doc/master/index.html
- **Training material:** https://github.com/root-project/training
- **Forum:** https://root-forum.cern.ch/t/welcome-to-the-root-forum/8/2

ROOT Installation

- You can follow instructions here: https://root.cern/install/
- In case of errors/problems, get in touch
- Note: you can also install ROOT from source files located in git repositories