ROOT - Einleitung

Der Vergleich von Messungen mit theroretischen Modellen ist ein wichtiger Bestandteil der Experimentalphysik. Dazu müssen in der Regel Daten selektiert, statistisch analysiert, modelliert, Modelparameter angepasst und Fehler abgeschätzt werden.

- Mit den bisher gelernten C++ Elementen könnten wir im Prinzip die Daten eines Experimentes analysieren, aber es fehlt z.B. noch
  - graphische Darstellung
  - Datenanpassung
  - I/O für komplexere Datenstrukturen (HEP Experimente > 10^6 Kanäle)
  - Speichern von Objekten

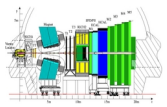
ROOT Data Analysis Framework entwickelt seit 1995 am CERN als Open Source Project unter GNU LGPL zur Datenanalyse am LHC als C++ Klassenbibliothek.

- 10 Vollzeit-Entwickler am CERN
- Offener Quellcode → software wird von den Usern mit- und weiterentwickelt
- Dokumentation, Tutorials und Quellcode: https://root.cern.ch/

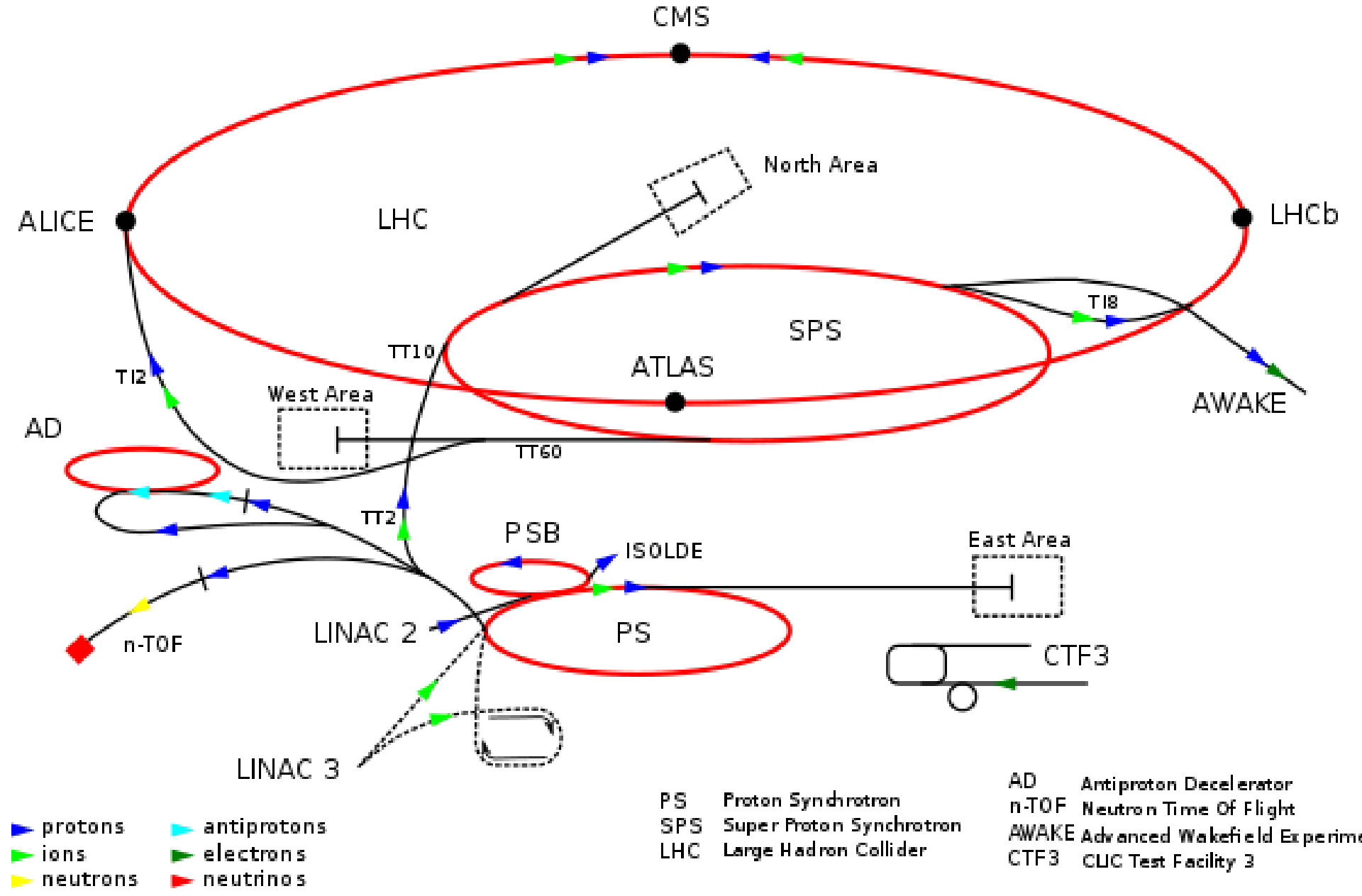
Large Hadron Collider at CERN

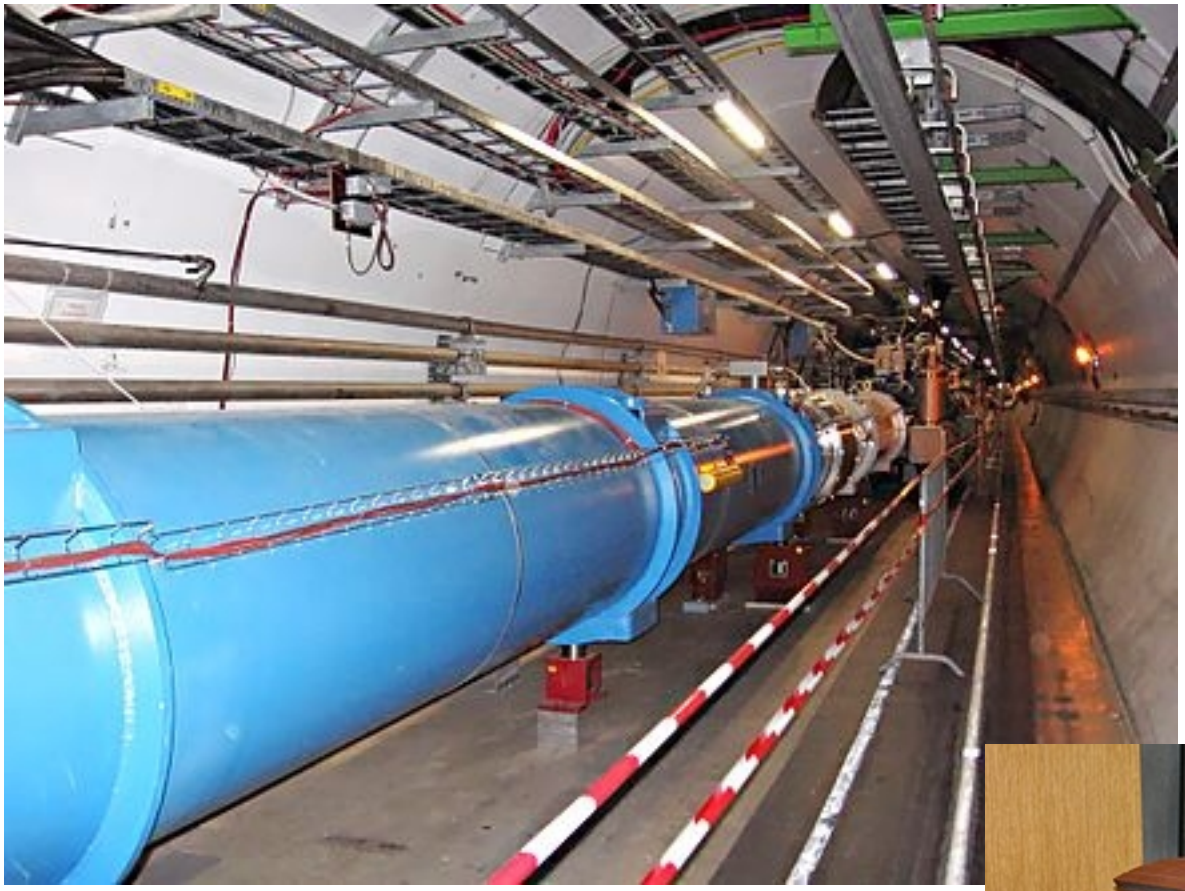
CERN – in a Nutshell

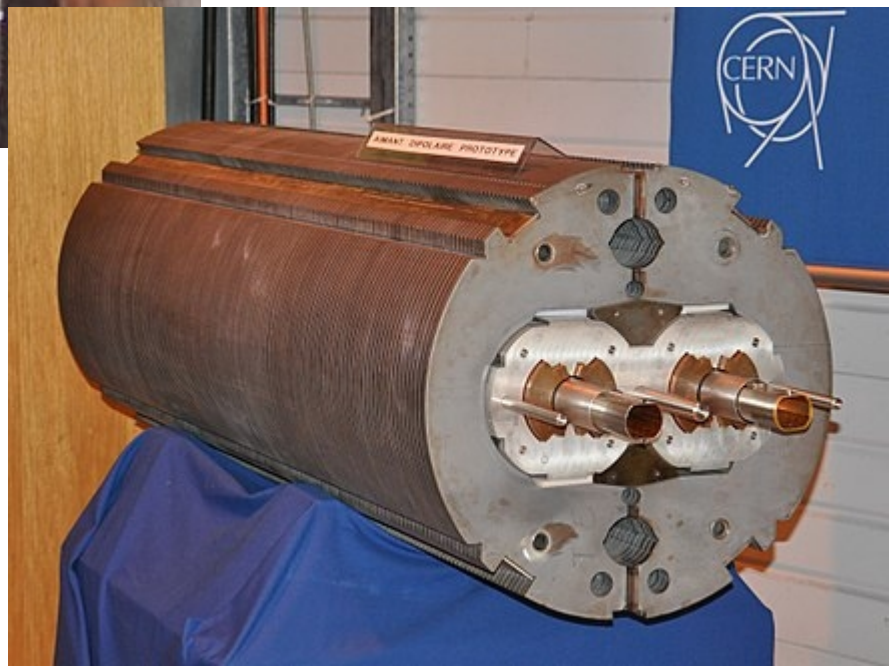
- CERN = Conseil Europeen pour la Recherche Nucleaire
- Is the largest research facility for particle physics world wide
- Is located in Switzerland, in Meyrin close by Geneva
- It has 23 member states from Europe with others associated
- ~ 3300 employees and about 13500 guest scientists
- The annual budget is about 750 million Euro
- The facility has 3 accelerators, PS, SPS and the LHC

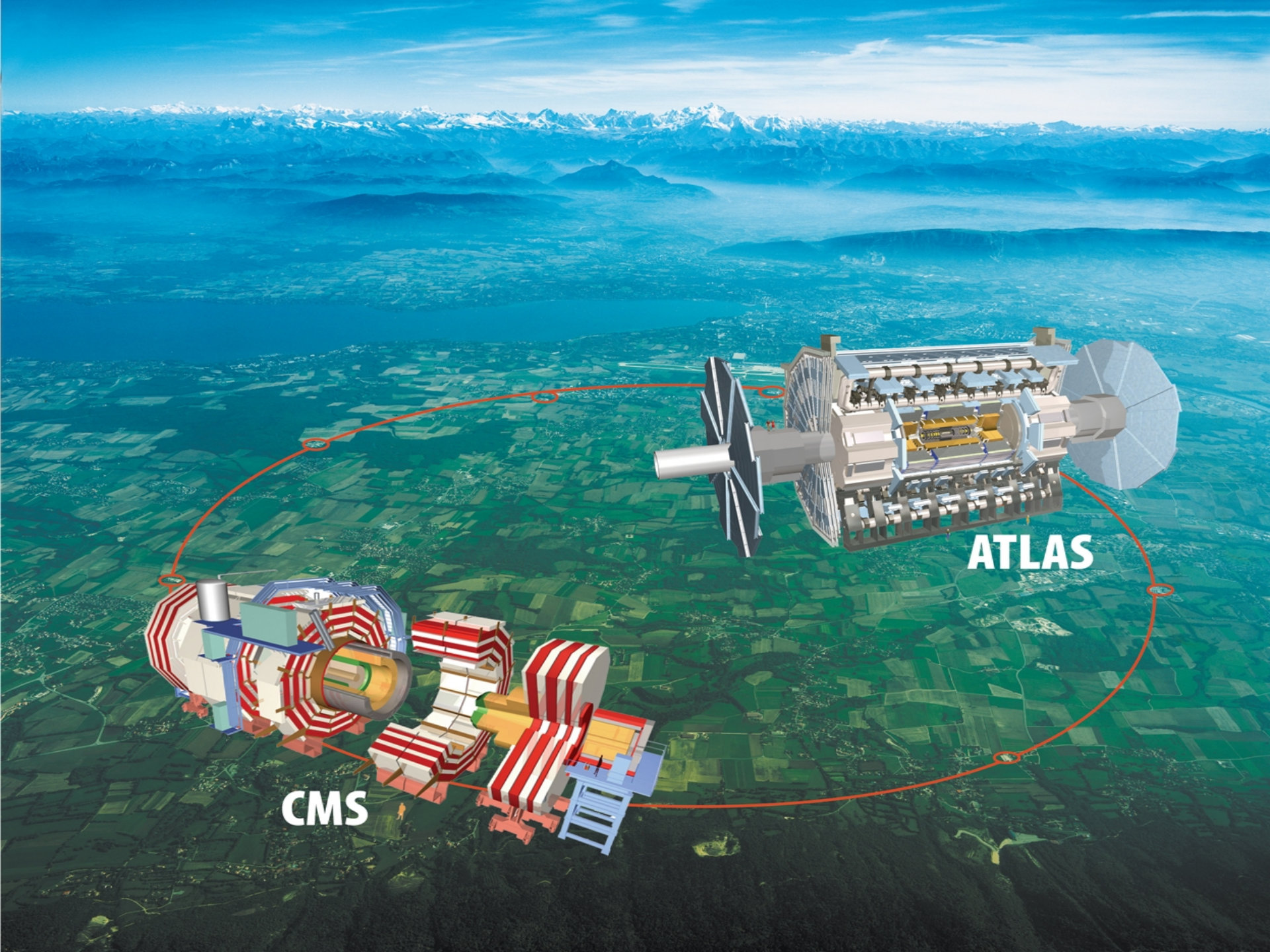


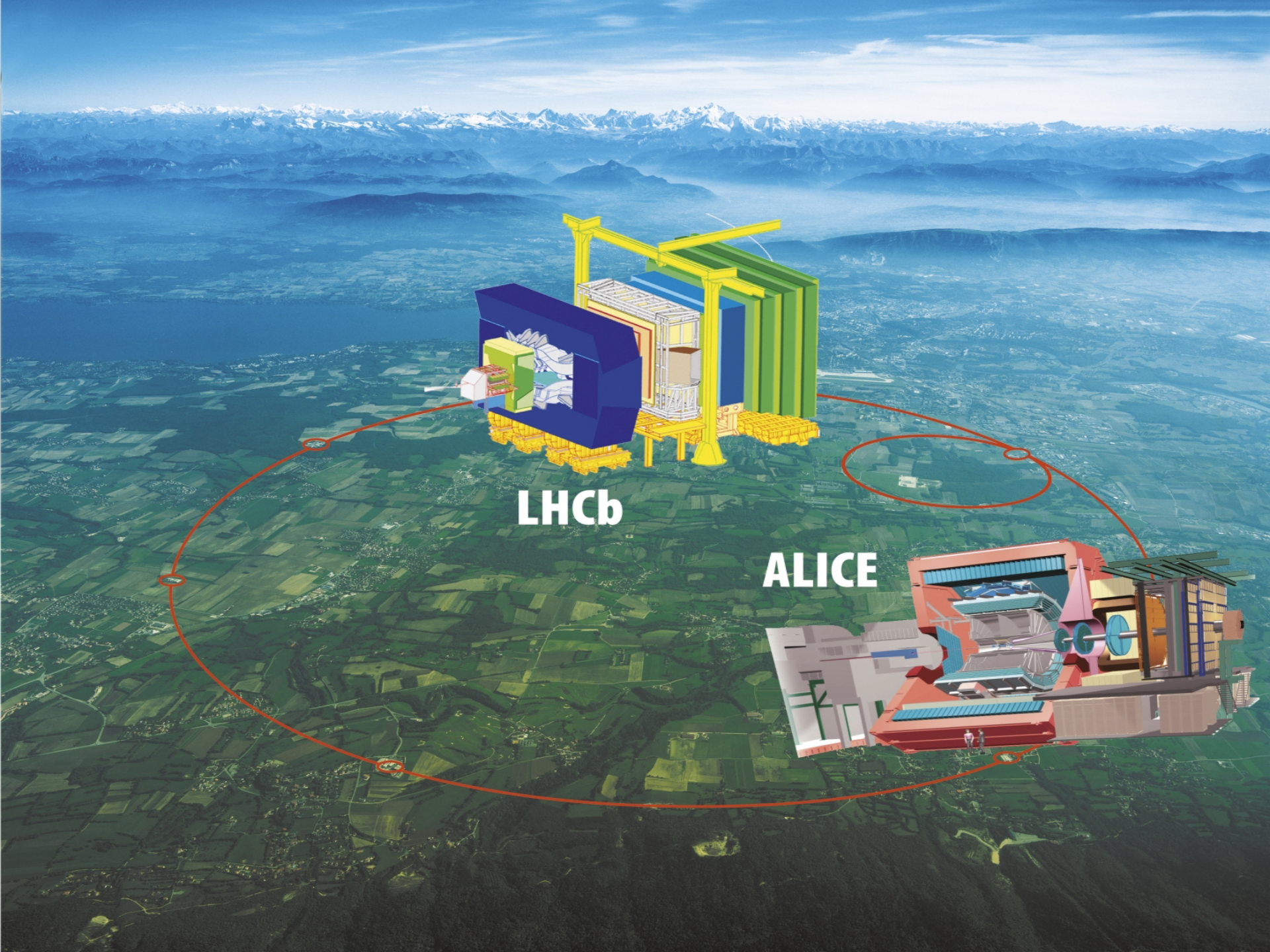

LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
LHC 27 km


CMS
ALICE
LHC
LHCb
North Area
SPS
TI8
AWAKE
ATLAS
TT10
TI2
West Area
AD
TT60
TT2
PSB
ISOLDE
East Area
n-TOF
LINAC 2
PS
CTF3
LINAC 3
protons
antiprotons
ions
electrons
neutrons
neutrinos
PS Proton Synchrotron
SPS Super Proton Synchrotron
LHC Large Hadron Collider
AD Antiproton Decelerator
n-TOF Neutron Time Of Flight
AWAKE Advanced Wakefield Experime
CTF3 CLIC Test Facility 3




CERN


ATLAS
CMS


LHCb
ALICE

ROOT - Einleitung

ROOT

- Framework zum Prozessieren grosser Datenmengen
  - Data @ LHC: > 10 PetaByte / Jahr und Experiment
  - Selektion von wenigen Ereignissen aus 10^{12} Kandidaten
- stellt statistische Analyse Algorithmen zur Verfügung
- mathematische Bibliothek mit nicht trivialen und optimierten Funktionen
- Multivariante Analyse Werkzeuge und neuronale Netze
- Werkzeug zur Visualisierung von Daten und Experimentgeometrie
- Interface zur Simulation von physikalischen Ereignissen in Detektoren
- Plattform für das parallele Bearbeiten von Ereignissen (PROOF)
- Implementiert auf unterschiedlichen Betriebssystemen
  Linux, MacOS X, Windows

RootInstall.pdf

ROOT - Start

ROOT setup: Installationspfade von ROOT müssen den environment variablen $PATH und $LD_LIBRARY_PATH hinzugefügt werden

```
export ROOTSYS=/cern/root
export PATH=.:$PATH:$ROOTSYS/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
```

Check: `echo $PATH`
`echo $LD_LIBRARY_PATH`

Im CIP Pool bitte ausführen:

```
$> source setroot.sh
```

ROOT startup im Terminal Fenster mit

```
$> root
root [0] _
root [1] .?
root [2] .q
root [3] qqqqq
```

ROOT history:
`$HOME/.root_hist`

- root prompt versteht C++ mit dem Interpreter CINT (V5.xx) / CLing (V6.xx)
  Python binding über PyROOT mit "import ROOT"
  Benutzen Sie die keys ⇧ ⇩ um Befehle erneut zu verwenden
- einfache Benutzung als Taschenrechner auch mit definierten Funktionen
- Darstellen von Funktionen

```
root [4] TF1 *f1 = new TF1("f1","[0]*sin([1]*x)/x",0.,10.);
root [5] f1->SetParameter(0,1); f1->SetParameter(1,1);
root [6] f1->Draw();
```

ROOT – as Calculator

```
marks@jma:~> root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.06/02                        http://root.cern.ch |
  |                                  (c) 1995-2014, The ROOT Team     |
  | Built for linuxx8664gcc                                           |
  | From heads/master@v6-07-02-437-gb06340c, Mar 02 2016, 19:01:57    |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'        |
   ------------------------------------------------------------------

root [0] gROOT->GetVersion()
(const char *) "6.06/02"
root [1] double x=.5
(double) 0.500000
root [2] int N=30 ; double gs = 0 ;
root [3] for(int i=0;i<N;i++) gs+=TMath::Power(x,i)
root [4] TMath::Abs(gs - (1-TMath::Power(x,N-1))/(1-x))
(Double_t) 1.86265e-09
root [5] double val = 0.992 ;
root [6] sin(val)
(double) 0.837122
root [7] TMath::Pi()
(Double_t) 3.14159
root [8] .!pwd
/local/home/marks
root [9] .!whoami
marks
root [10]
```

Get functions from TMath:
`TMath::function`

Execute system commands:
`.!<unix command>`

ROOT – as Calculator

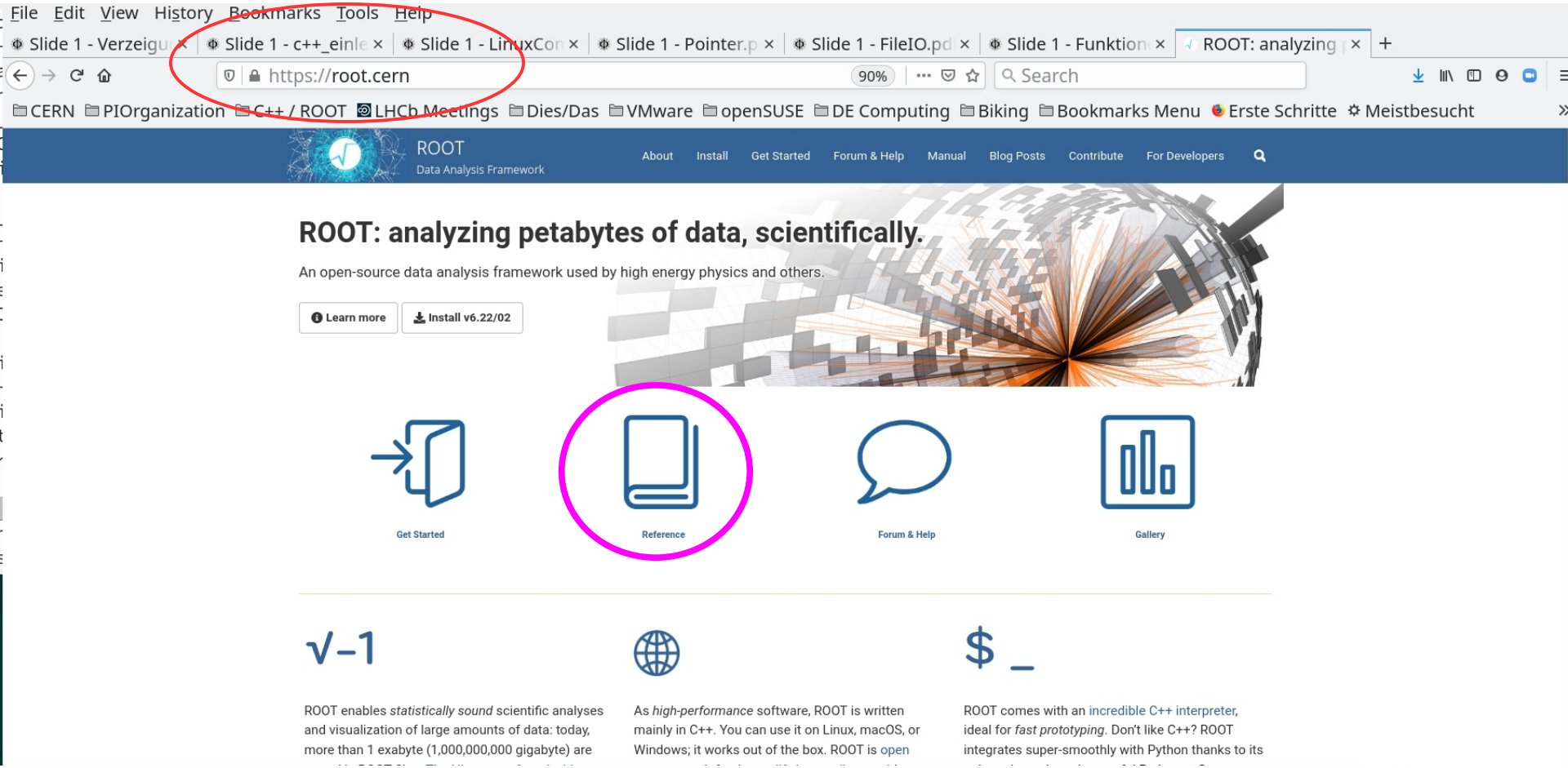
```
marks@jma:~> root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.06/06                        http://root.cern.ch |
  |                                  (c) 1995-2016, The ROOT Team |
  | Built for linuxx8664gcc                                         |
  | From heads/v6-06-00-patches@v6-06-04-66-gb9c1d82, Jul 06 2016, 18:28:55 |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'      |
   ------------------------------------------------------------------

root [0] TVector3 a(6.,10,40);
root [1] TVector3 b(1.,1,40);
root [2] TVector3 c = a + b ;
root [3] TVector3 d = a.Cross(b);
root [4] double s = a.Dot(b);
root [5] c(0)
(Double_t) 7.00000
root [6] c(1)
(Double_t) 11.0000
root [7] c(2)
(Double_t) 80.0000
root [8] a.Angle(b)
(Double_t) 0.249542
root [9] aperp = a.Orthogonal();
root [10] e = aperp.Unit();
root [11] a.RotateY(TMath::Pi());
root [12] a.Rotate(TMath::Pi()/4, c);
root [13] a(0)
(Double_t) -16.0374
root [14] a(1)
(Double_t) 3.79044
root [15] a(2)
(Double_t) -38.2679
```

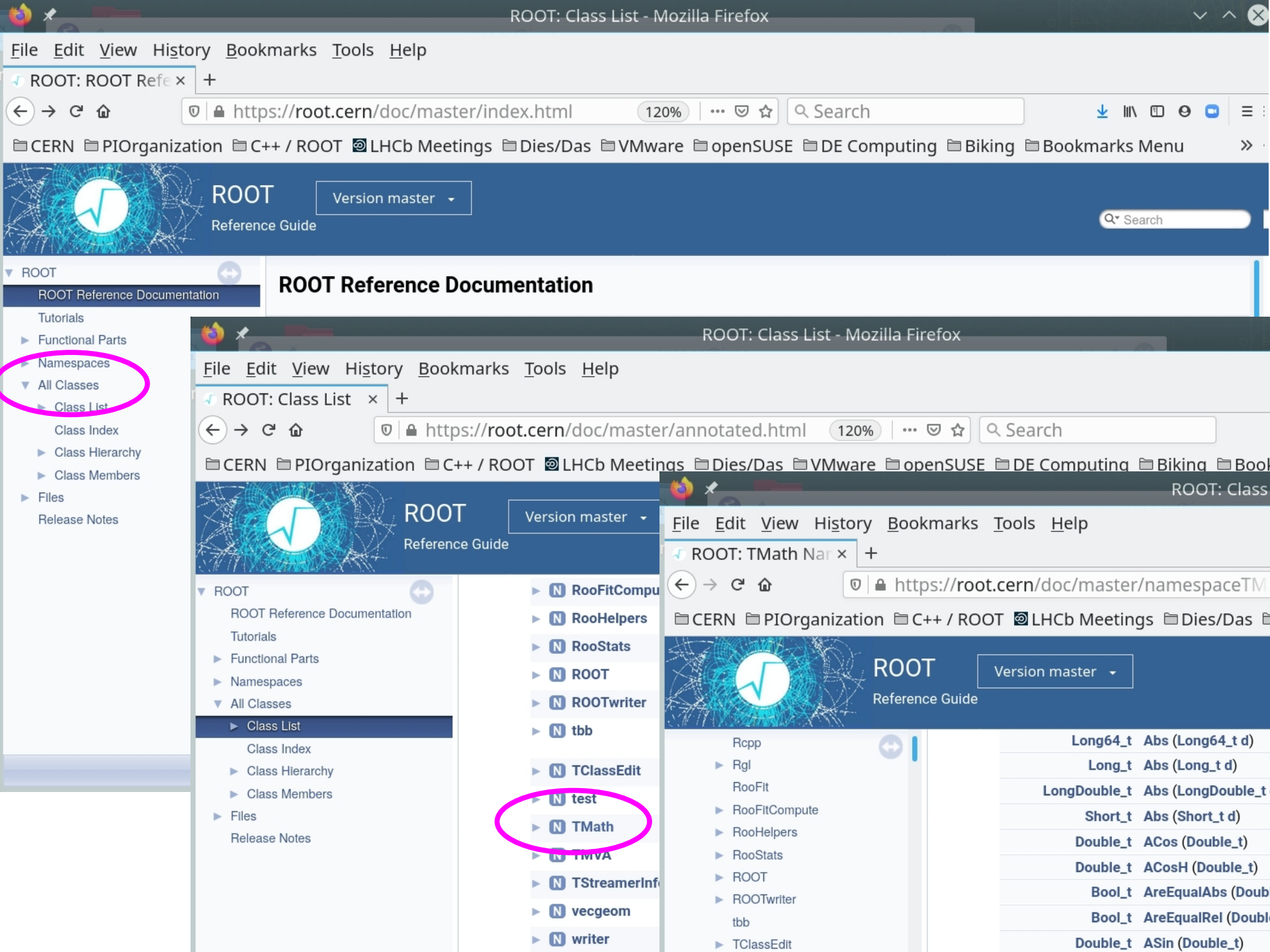
Vektor Rechnung mit dem C++ Interpreter in ROOT

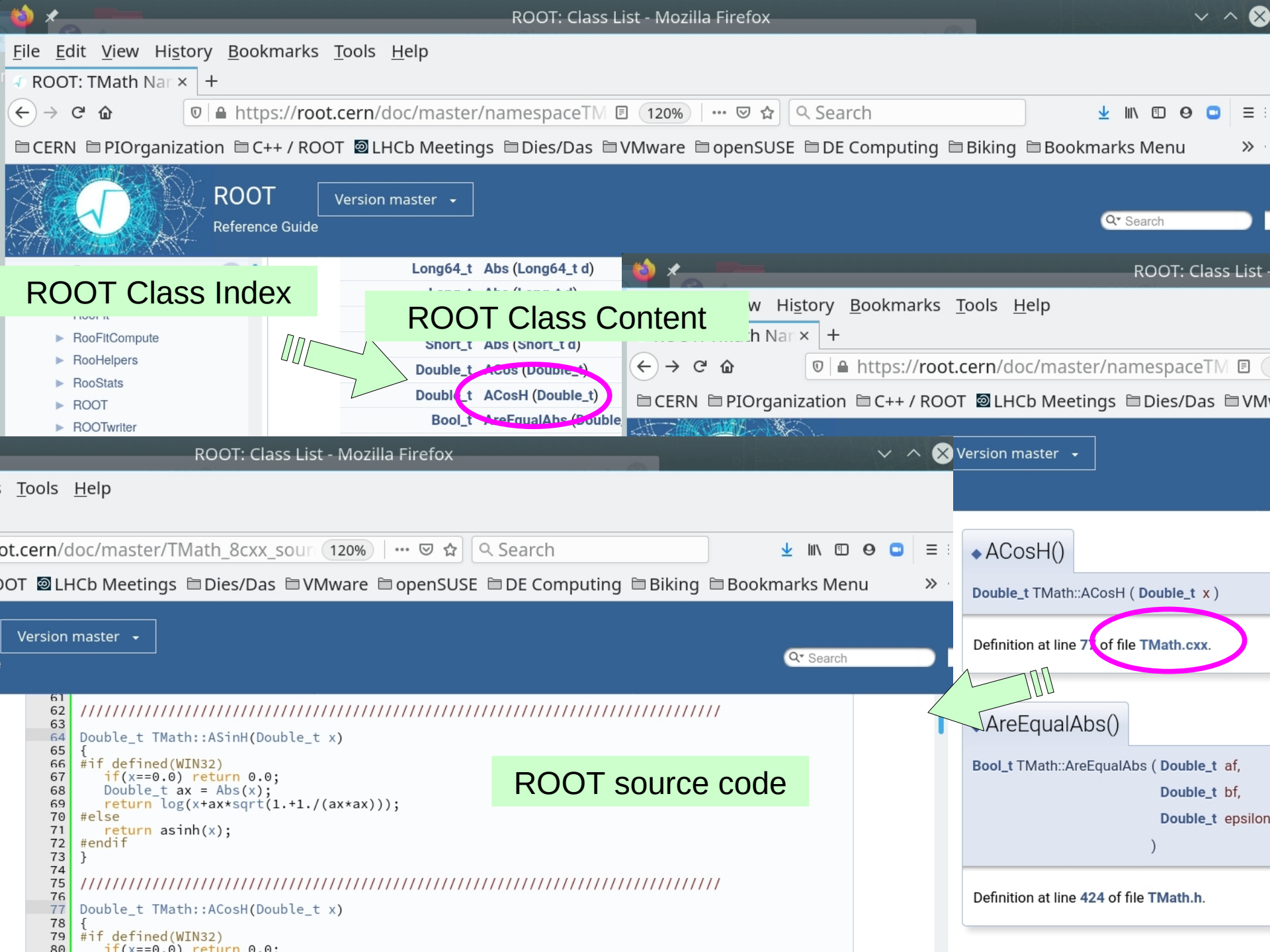
Offensichtlich benutzen wir die Klasse `TVector3`. Woher wissen wir welche Methoden implementiert sind?

```
root[16] .class TVector3
```

ROOT – Dokumentation




ROOT: Class List - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ROOT: ROOT Refe
https://root.cern/doc/master/index.html
120%
Search
CERN PIOrganization C++ / ROOT LHCb Meetings Dies/Das VMware openSUSE DE Computing Biking Bookmarks Menu
ROOT
Version master
Reference Guide
Search
ROOT
ROOT Reference Documentation
Tutorials
Functional Parts
Namespaces
All Classes
Class List
Class Index
Class Hierarchy
Class Members
Files
Release Notes
ROOT Reference Documentation
ROOT: Class List - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ROOT: Class List
https://root.cern/doc/master/annotated.html
120%
Search
CERN PIOrganization C++ / ROOT LHCb Meetings Dies/Das VMware openSUSE DE Computing Biking
ROOT
Version master
Reference Guide
ROOT
ROOT Reference Documentation
Tutorials
Functional Parts
Namespaces
All Classes
Class List
Class Index
Class Hierarchy
Class Members
Files
Release Notes
RooFitCompu
RooHelpers
RooStats
ROOT
ROOTwriter
tbb
TClassEdit
test
TMath
TMVA
TStreamerInf
vecgeom
writer
ROOT: Class
File Edit View History Bookmarks Tools Help
ROOT: TMath Nar
https://root.cern/doc/master/namespaceTM
CERN PIOrganization C++ / ROOT LHCb Meetings Dies/Das
ROOT
Version master
Reference Guide
Rcpp
Rgl
RooFit
RooFitCompute
RooHelpers
RooStats
ROOT
ROOTwriter
tbb
TClassEdit
Long64_t Abs (Long64_t d)
Long_t Abs (Long_t d)
LongDouble_t Abs (LongDouble_t
Short_t Abs (Short_t d)
Double_t ACos (Double_t)
Double_t ACosH (Double_t)
Bool_t AreEqualAbs (Doub
Bool_t AreEqualRel (Doubl
Double_t ASin (Double_t)


ROOT: Class List - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ROOT: TMath Nar
https://root.cern/doc/master/namespaceTM
120%
Search
CERN PIOrganization C++ / ROOT LHCb Meetings Dies/Das VMware openSUSE DE Computing Biking Bookmarks Menu
ROOT
Reference Guide
Version master
Search
ROOT Class Index
RooFitCompute
RooHelpers
RooStats
ROOT
ROOTwriter
ROOT Class Content
Long64_t Abs (Long64_t d)
Short_t Abs (Short_t d)
Double_t ACos (Double_t)
Double_t ACosH (Double_t)
Bool_t AreEqualAbs (Double
ROOT: Class List
History Bookmarks Tools Help
https://root.cern/doc/master/namespaceTM
CERN PIOrganization C++ / ROOT LHCb Meetings Dies/Das VM
ROOT: Class List - Mozilla Firefox
Tools Help
ot.cern/doc/master/TMath_8cxx_sour
120%
Search
OT LHCb Meetings Dies/Das VMware openSUSE DE Computing Biking Bookmarks Menu
Version master
Search
61
62 //
63
64 Double_t TMath::ASinH(Double_t x)
65 {
66 #if defined(WIN32)
67 if(x==0.0) return 0.0;
68 Double_t ax = Abs(x);
69 return log(x+ax*sqrt(1.+1./(ax*ax)));
70 #else
71 return asinh(x);
72 #endif
73 }
74
75 //
76
77 Double_t TMath::ACosH(Double_t x)
78 {
79 #if defined(WIN32)
80 if(x==0.0) return 0.0;
ROOT source code
Version master
ACosH()
Double_t TMath::ACosH (Double_t x)
Definition at line 71 of file TMath.cxx.
AreEqualAbs()
Bool_t TMath::AreEqualAbs (Double_t af,
Double_t bf,
Double_t epsilon
)
Definition at line 424 of file TMath.h.

ROOT - Start

Graphische Darstellung von Daten aus einem File

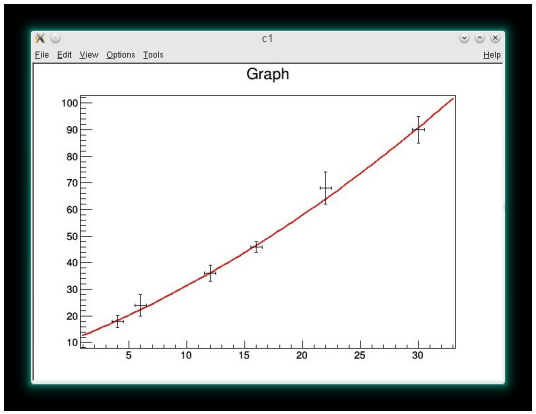
```
$> root
root [0] TGraphErrors *gr = new TGraphErrors("myFile.txt");
root [1] gr->SetTitle("myTitle;xAxis;yAxis");
root [2] gr->Draw("AP");
```

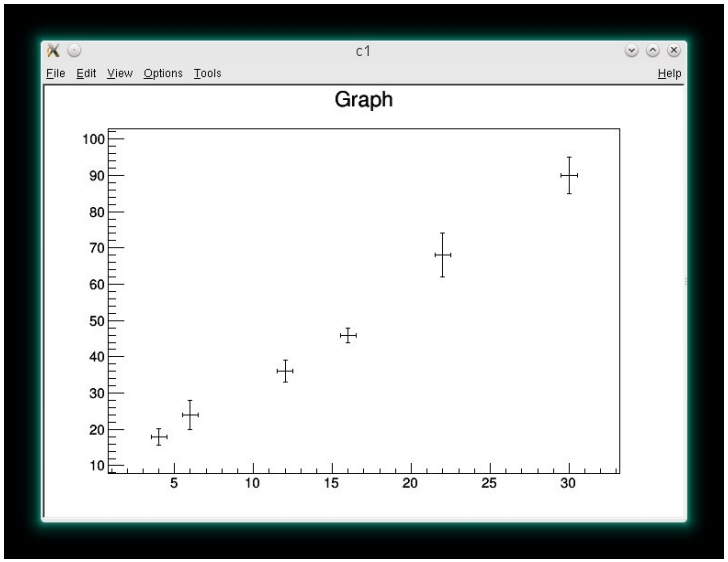
Draw **A**xis and **P**oints

myFile.txt :

```
4    18   0.5  2.3
6    24   0.5  4.1
12   36   0.5  3.0
16   46   0.5  2.0
22   68   0.5  6.0
30   90   0.5  5.0
```

- Graphik kann modifiziert werden:
  - Symbole
  - Achsenbeschriftung / Tick marks
  - beliebiger Text
- Als root file speichern zur weiteren Bearbeitung
- Parameter anpassen mit dem
  Fit Panel unter dem Menue „Tools“





ROOT – Wo arbeiten wir?

- Im CIP Pool → Installation aktuell allerdings 2 Jahre alt
  - Start Cisco AnyConnect
  - Start MobaXterm oder Linux shell
  In MobaXterm shell: `ssh myUserid@physik1.kip.uni-heidelberg.de`
- Joergs Desktop Rechner → neueste ROOT Version, für das python Interface notwendig.
  - Start MobaXterm oder Linux shell
  - Login mit CIP Pool userids mit dem password !KIPinf227
  CIP home dir nicht vorhanden → transfer data mit scp wenn notwendig
  `ssh myUserid@zenon.physi.uni-heidelberg.de`
  - Beim ersten login:
    I) testen ob das keyboard geht → ist y wirklich y
    II) root testen → `root` in der shell eintippen
    III) password ändern → `passwd` eintippen und neues password setzen
- Eigene ROOT Installation RootInstall.pdf

Wir wollen den ROOT Interpreter und die ROOT Klassen verwenden, um das Rechnen mit Vektoren zu demonstrieren und ausserdem Funktionen darzustellen.

Arbeitsvorschlag:

- Benutzen Sie den ROOT Interpreter mit einfachen C++ keywords. So könnten Sie zum Beispiel die Summe $\sum_{k=1}^{N} \frac{x^k}{k}$ für N=10 und x = 3 bestimmen.

  Was ist umständlich?

- Benutzen Sie die Klasse `TLorentzVector` um die invariante Masse der Vierervektoren `u(3.,10.,40.,10.)` und `v(1.,0.,10.,3.)` zu bestimmen. Bestimmen Sie die transversale Komponente des Summenvektors von u und v.

- Laden Sie das Text File myFile.txt mit gedachten Messungen und benutzen Sie die Klasse `TGraphErrors("myFile.txt")` zur Darstellung. Probieren Sie die Fit Funktionalität aus.

- In Ihrem home Verzeichnis gibt es ein File `.root_hist` . Was befindet sich in dem File?

ROOT - Start

Ausführung von Programmen

```
$> root
root [0] .x myMacro.C(2,2)
```

- File myMacro.C wird von CLing interpretiert und ausgeführt

myMacro.C:

```
int myMacro(int a, int b)
{
  int s = a + b ;
  return s ;
}
```

```
$> root
root [0] .x myMacro.C+(2,2)
root [1] myMacro(2,2)
```

- File myMacro.C wird mit ACLiC kompiliert und es wird eine shared library erzeugt myMacro.so
  - system compiler wird benutzt
  - das File wird nur kompiliert, wenn es geändert wurde.
- CINT/CLing versus compiled C++
  - Interpreter → rapid prototyping
  - Compiled code → Syntax check, schneller bei der Ausführung

ROOT - Start

Ausführung von Programmen

- Gemeinsames Ausführen von ROOT und C++ code

```
$> g++ -o throwDice throwDice.cc  `root-config --cflags --glibs`
$> ./throwDice 3
Dice 1: 3
Dice 2: 4
Dice 3: 4
```

Programm wird in der shell unter Hinzufügen der ROOT spezifischen flags kompiliert

`command`: command wird ausgefuehrt
-o FileName : FileName des Programms

```
$> root
root [0] .x throwDice.cc(3)
Dice 1: 5
Dice 2: 4
Dice 3: 2
```

Programm wird in ROOT von CLing interpretiert

```
root [1] .x throwDice.cc+(3)
root [2] throwDice(3)
Dice 1: 2
Dice 2: 2
Dice 3: 2
```

Programm wird in ROOT mit ACLiC kompiliert und eine shared library gebaut, es kann dann in ROOT ausgeführt werden

File Edit Options Buffers Tools C++ Help

Save Undo

```
# include <cstdlib>
# include <iostream>
# include "TRandom3.h"

using namespace std;

# ifndef __CINT__  // the following code will be invisible for CINT interpreter
int rollDice();
void throwDice (int NDice);

int main(int argc, char* argv[])
{
  throwDice (atoi(argv[1])) ;
  return 0 ;
}
# endif   // end ignore

void throwDice (int NDice) {
  for (int I = 1 ; I <= NDice; I++)
     cout << "Dice " << I << ": " << rollDice() << endl;
  return ;
}
int rollDice() {
   UInt_t MaxInt = 6 ;
   TRandom3 *R = new TRandom3();
   R->SetSeed(0);  // set seed to machine clock
   UInt_t NRnd  = R->Integer(MaxInt) ;
   int roll = static_cast <int> (NRnd) + 1 ;
   return roll ;
}
```

-:--- **throwDice.cc** Top (4,0) (C++/l Abbrev)

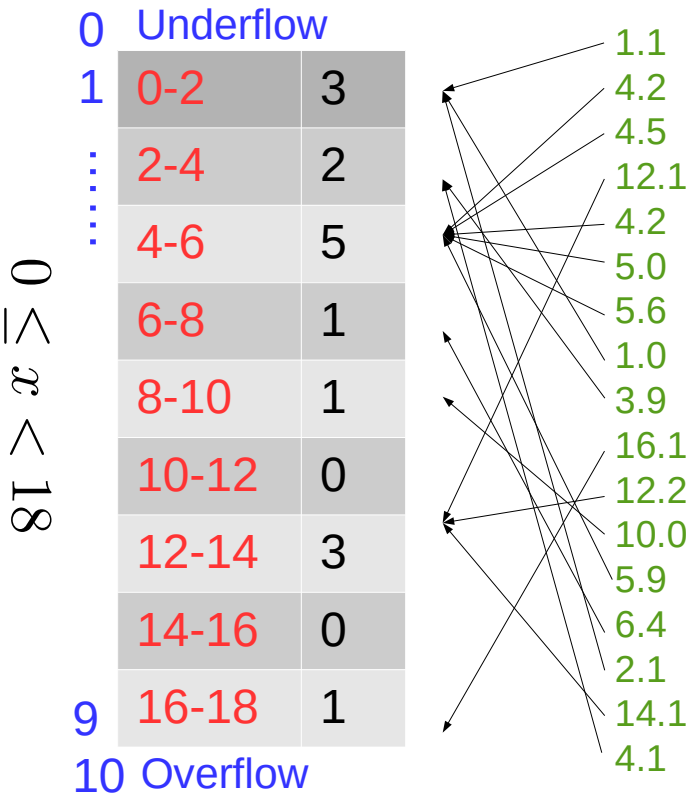
throwDice.cc

https://root.cern/doc/master/classTRandom3.html

Arbeitsvorschlag: Können Sie das Programm so ändern, dass es bevorzugt 3 würfelt.

ROOT - Histogramme

Histogramm: In Abschnitte geteiltes Diagramm bezüglich einer Variablen, in das die Anzahl der Werte für jeden Abschnitt eingetragen wird



- Histogramm Klassen in ROOT:
  TH1F, TH1D (1 dimensional)
  TH2F, TH2D (2 dimemsional)
- Benutzung:
  - Initialisieren
  - Variable füllen
  - Graphische Darstellung und Bearbeitung
  - Resultate in einem ROOT File speichern

Mean

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Standard Deviation

$$s = \sqrt{\frac{1}{n-1}\left[\left(\sum_{i=1}^{n} x_i^2\right) - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right]}$$

Skewness

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^3$$

Curtosis

$$C = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^4$$

Mean +/- Error
Standard Deviation
Skewness
Curtosis

ROOT - Histogramme

Histogramm: In Abschnitte geteiltes Diagramm bezüglich einer Variablen, in das die Anzahl der Werte für jeden Abschnitt eingetragen wird

- Beispiel Macro der Histogramm Klassen in ROOT

myHisto.C

  - Initialisieren

```
TH1F *h = new TH1F ("myhist","Altersverteilung",60,0.,60.);
```

pointer auf das TH1F Objekt h | Histogramm Name | Titel | Anzahl der Bins | Grenzen

  - Variable füllen

```
h->Fill(age,weight=1.);
```

  - Graphische Darstellung und Bearbeitung

```
h->Draw("Options");
Options = "","E","SAME","C","L"
```

Viele Optionen und Beispiele in der Klassenreferenz:
https://root.cern/doc/master/classTHistPainter.html#HP01

Fehlerbalken | in den gleichen Plot | glatte Kurve | Line

  - Resultate in einem ROOT File speichern

```
TFile outFile ("myFile.root","RECREATE");
```
Outputfile öffnen

```
h→Write();
```
Histogramm schreiben

```
outFile.Close();
```
Outputfile schließen

ROOT - Histogramme

Histogramm: In Abschnitte geteiltes Diagramm bezüglich einer Variablen, in das die Anzahl der Werte für jeden Abschnitt eingetragen wird

- Ausführen des Beispiel Macros und Ansehen des Histogramms myHisto.C

```
$> root myHisto.C
root [0]
Processing myHisto.C...
Histogram filled with 20 entries
root [0] .q
$> ls altersverteilung.root
altersverteilung.root
$> root  --web=off altersverteilung.root
Root [0] TBrowser T
```

Programm wird in ROOT von CLing interpretiert

Enthält ein ROOT file mit Histogramm

Mit dem Browser läßt sich das ROOT file ansehen und das Histogramm darstellen. Mit --web=off wird der ROOT mode verwendet

File Edit Options Buffers Tools C++ Help

Save Undo

```
# include <iostream>
# include <TRandom3.h>
# include <TFile.h>
# include <TH1.h>

using namespace std;

void myHisto() {
   int NumberParticipants = 20 ;
   Double_t averageAge= 20. , sigmaAge = 1.0 ;
   Double_t age;

   TRandom3 *R = new TRandom3();
   R->SetSeed(0);  // set seed to machine clock

   TH1F *h = new TH1F ("myhist","Altersverteilung",,0.,60.);

   for (int I = 0 ; I < NumberParticipants ; I++) {
       age = R->Gaus(averageAge,sigmaAge) ;
       h->Fill(age);
   }

   cout << "Histogram filled with "<< NumberParticipants<< " entries" << endl;

   TFile outFile ("altersverteilung.root","RECREATE");
   h->Write();
   outFile.Close();

   return ;
}
```

-:--- **myHisto.C** Top (5,0) (C++/l Abbrev)

myHisto.C

ROOT - Histogramme

- Graphische Darstellung im Programm

  Die graphische Ausgabe von ROOT wird in einem “Canvas“ dargestellt, das vom Displaymanager kontrolliert wird. Es kann in mehrere Canvas Fenster geschrieben werden. Ein Canvas kann in Unterbereiche geteilt werden.

```
TCanvas *C = new TCanvas("myC","Pad Histogram",0,0,800,600);
```

pointer auf das TCanvas Objekt C | Canvas Name | Titel | Pixel Koordinaten von oben links | Breite / Höhe des Fensters

```
C->Divide(2,2);
```

Erzeugen eines Pads mit 2x2 Subfeldern zum Darstellen von Plots

```
C->cd(1);
h->DrawClone();
```

Darstellen des Histogramms h und erzeugen einer Kopie

```
C->cd(2);
h->Scale(1./NumEntries);
h->Draw();
```

Normieren der Anzahl der Histogramm-einträge von h auf 1

```
C->Write();
```

Schreiben des Canvas in das ROOT file

ROOT - Histogramme

- Histogramm Ergebnisse im Programm weiter verwenden und setzen

```
Double_t binContent = h->GetBinContent(22);
Double_t error = h->GetBinError(22);
```

Get Inhalt Bin 22

```
Double_t NumEntries = h->GetEntries();
Int_t MaxBin = h->GetMaximumBin();
Double_t histoMean = h->GetMean();
Double_t histoMeanError = h->GetMeanError();

TAxis *xaxis = h->GetXaxis();
Double_t binCenter = xaxis->GetBinCenter(22);

h->GetXaxis()->SetTitle("X axis title");
h->GetYaxis()->SetTitle("Y axis title");

h->SetTitle("Histogram title;  Another X title Axis");
```

- Histogramm speichern und lesen

```
TFile f("histos.root", "new")
h->Write();
```

Schreiben des Histograms in das ROOT file

```
TFile f("histos.root");
TH1F *h = (TH1F*)f.Get("hgaus");
```

Lesen des Histograms

ROOT - Histogramme

- Operationen mit Histogramm

myHistoExtended.C

```
TH1F *h = new TH1F ("myhist","Altersverteilung",60,0.,60.);
h->GetXaxis()->SetTitle("age[years]");
h->Sumw2(); // get access to proper error treatment

TH1F *hbck = new TH1F ("hbck","Untergrund
                                Altersverteilung",60,0.,60.);
hbck->GetXaxis()->SetTitle("age[years]");
hbck->Sumw2();

TH1F *hsum  =(TH1F*)h->Clone();
hsum->SetName("hsum");
hsum->Add(hbck,1.0);

TH1F *h3 = new TH1F ("h3","h1+h2",nBins,xMin,xMax);
h3->Add(h1,h2);
h3->Add(h1,h2,1.,-1.);
```

Erzeugen einer Kopie des Histogramms h

Addieren von hbck zu h mit Skalenfaktor 1

Addieren von 2 Histogrammen

Subtrahieren von 2 Histogrammen

- Histogramme h1,h2,h3 haben das gleiche binning!
- Sonst müssen Bingrenzen erhalten bleiben

ROOT - Histogramme

myHistoExtended.C

- Histogramme in 2D

```
TH2F *h2D = new TH2F("h2D", "Alter vs Groesse",
                                          60,0.,60.0,50,140,190);
h2D->GetXaxis()->SetTitle("age[years]");
h2D->GetYaxis()->SetTitle("height[cm]");
h2D->Sumw2();
```

Optionen zur Darstellung

```
TH2F *h = new TH2F("h","2D Gauss",40,-4.,4.,40,-4.,4.);
for(int j=0; j<10000;j++){
   double x = R->Gauss(0,1);
   double y = R->Gauss(1,2);
   h->Fill(x,y);
}
h->Draw("COLZ");
h->Draw("CONTZ");
h->Draw("LEGO");
```

Gauss Verteilung, Mean=0 und Breite =1

Gauss Verteilung, Mean=1 und Breite =2

Farbänderungen entsprechend der Einträge in z

Contour Plot bezüglich z

Lego Plot Entries in z

https://root.cern/doc/master/classTHistPainter.html#HP01

ROOT - Histogramme

- Histogramme in 2D

```
TH2F *h2D = new TH2F("h2D", "Alter vs Groesse",
                                    60,0.,60.0,50,140,190);
```

Projektionen auf die Achsen erzeugen 1D Histogramme

```
TH1F *projX =  h2D->ProjectionX("MyProjX",1,-1);
TH1F *projY =  h2D->ProjectionY("MyProjY",1,nBin);
```

Binbereich: -1 =: kopieren bis zum Ende

Profil von 2D Histogrammen: Mittelwerte in Bins bezüglich einer Koordinate

```
TProfile *profX =  h2D->ProfileX("MyProfileX",1,
TProfile *profY =  h2D->ProfileY("MyProfileY",1,
```

Binbereich in y/x

Hier wird die Bin Grösse des 2D Histogramms verwendet. Für die Mittelwertbildung werden nur die Bin Mitten benutzt! Daher haben wir nur eine Näherung.

TProfile: bilde Mittelwerte von y in Bins von x

```
TProfile *prof = new TProfile("prof","Title",
                                    100,0.,60.0,140,190);

prof→Fill(x,y);
```

Bins von x. x Bereich y Bereich

Mittelwert: y Fehler: $rms/\sqrt{n}$

Datenanalyse – erste Schritte

Wir wollen nun beispielhaft eine Messung analysieren, bei der das Ausgangssignal eines Verstärkers wiederholt gemessen wird. Der Messbereich ist [0,450]. Gleichzeitig wird mit jeder Messung die Raumtemperatur aufgezeichnet. Es werden 2 Textfiles geschrieben, signal.txt und temperature.txt

Arbeitsvorschlag:

- Schreiben Sie ein ROOT Macro, das das Verstärkersignal darstellt. Bestimmen Sie die Anzahl der Messungen, den Mittelwert und den Fehler und die Standard-abweichung. Ist das Signal gaussverteilt? Welcher Bereich ist als Signalbereich sinnvoll? Mit welcher Auflösung wird das Signal gemessen?
- Schätzen Sie den Anteil der Untergrundeinträge im Signalbereich ab?
- Gibt es eine Korrelation zwischen Signal und Temperatur. Was bedeutet das für die Auflösung der Signalverteilung. Wie kann man die Temperaturabhängigkeit des Signals korrigieren? Wie groß ist dann die Auflösung?
- Subtrahieren Sie die Untergrundverteilung von der korrigierten Signalverteilung. Wie groß ist nun die Auflösung?

analysis.cc

Datenanalyse – erste Schritte

Wir wollen nun beispielhaft eine Messung analysieren, bei der das Ausgangssignal eines Verstärkers wiederholt gemessen wird. Der Messbereich ist [0,450]. Gleichzeitig wird mit jeder Messung die Raumtemperatur aufgezeichnet. Es werden 2 Textfiles geschrieben, signal.txt und temperature.txt

Lösungsschritte:

- Öffnen der Textfiles mit 2 Eingabeströmen
- Lesen jeweils bis EOF und speichern als vector <double>
- 2 1D Histogramme mit Signal und Temperatur, 1 2D Histogramm Temperatur gegen Signal auftragen.
- Interaktiv Gauss Anpassung, Auflösung = Sigma / Mean
- Integriere Signal im Bereich [350 – 450] (nur Untergrund). Da der Untergrund gleich-verteilt ist, kann damit auf den Untergrund im Signalbereich geschlossen werden.
- Im 2D Plot ist klar eine Korrelation zwischen Signal und Temperatur sichtbar. Die Auflösung ist dadurch überschätzt.
- Bestimme das mittlere Signal in 4 Bins der Temperatur (interaktive Gauss Anpassung). Daraus kann die Änderung mit der Temperatur bestimmt werden.
- Diese Änderung kann durch die Umrechnung des Signals auf T=19 Grad korrigiert werden. Die korrigierten Werten werden als vector gespeichert und histogrammiert.
- Damit kann die Auflösung bestimmt werden.
- Wir kennen die Zahl der Untergrundereignisse im gesamten Signal Bereich. Erzeuge ein Untergrund Histogramm in das Zufallswerte gefüllt werden. Das Untergrund Histogramm kann man abziehen.