ROOT 基础

孙浩凯 计算中心

2022年08月18日

sunhk@ihep.ac.cn

Outline

- 前期准备和说明
- ROOT是什么
- ROOT的主要用途(基础)
- 一点ROOT进阶
- 总结

前期准备和说明

知识背景 I

- C++：只需要最基础的部分。
  1. 《C++ Primer》5th 前六章，无需精读，第一遍后边用边查边学。
  2. CPP Reference [https://en.cppreference.com/w/]权威在线字典
- **概率论和数理统计：使用ROOT做物理分析的逻辑基础**
  1. 均值、期望值、标准差、误差传递、控制样本。。。
  2. （高斯）分布、假设检验、统计显著性、概率密度函数。。。

知识背景 II

- Linux：基本的命令和终端使用。
  1. terminal, shell(bash/zsh)
  2. cd, pwd, ls, mkdir, mv, cp, cat, less/more
  3. vi(->vim/neovim), emacs(spacemacs), vs code
  4. ssh, scp, rsync
  - Xshell, MobaXterm, FinalShell, WindTerm, JuiceSSH

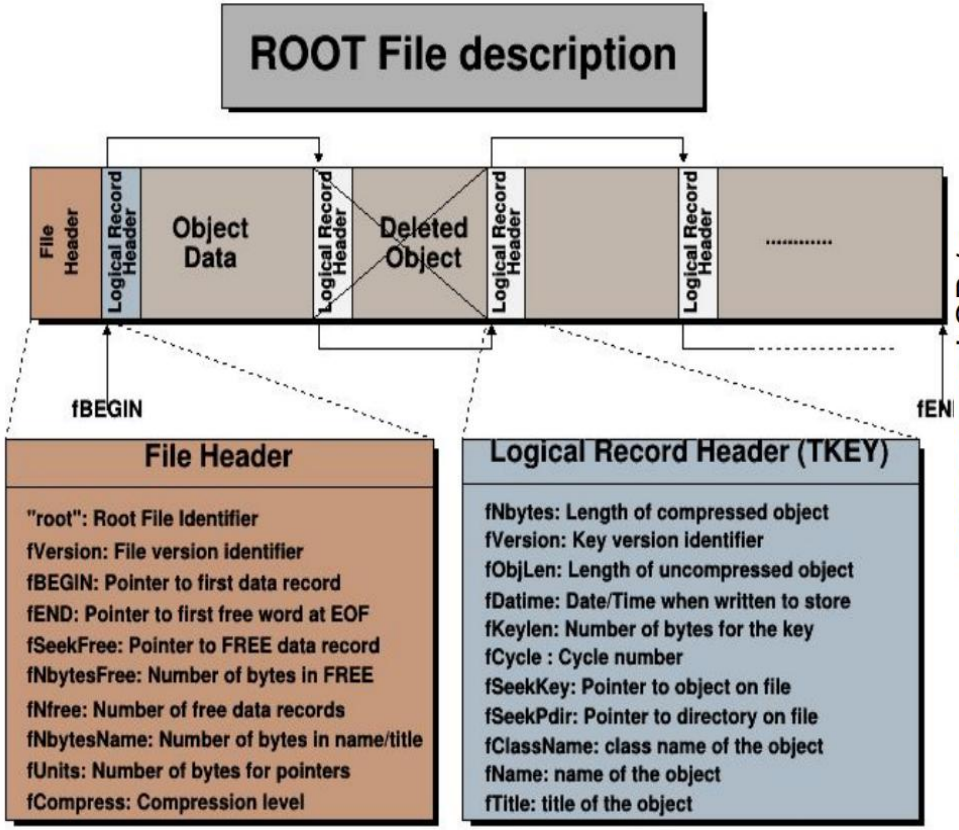
知识背景 III

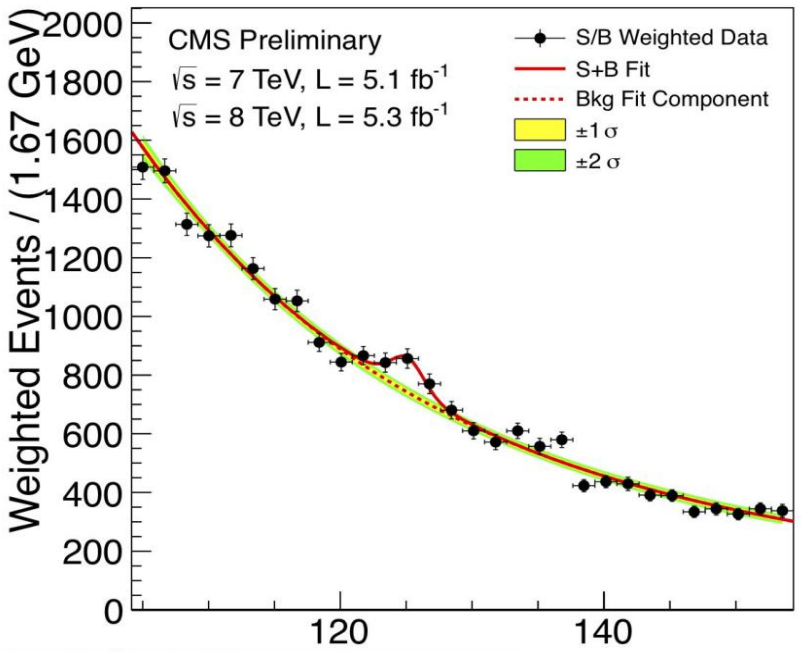
- ✓ ROOT官网 [https://root.cern/]
- ✓ ROOT训练场 [https://github.com/root-project/training]
- ✓ ROOT官方论坛 [https://root-forum.cern.ch/]
- ✓ ROOT文档 [https://root.cern/doc/master/index.html]
- ✓ *本次讲座基于ROOT版本 v6.26*

ROOT是什么

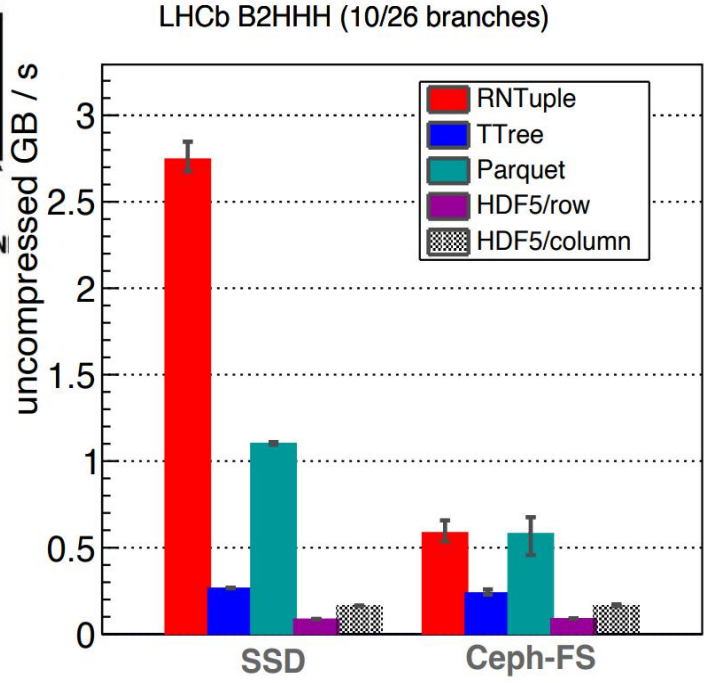
数据存储和读写

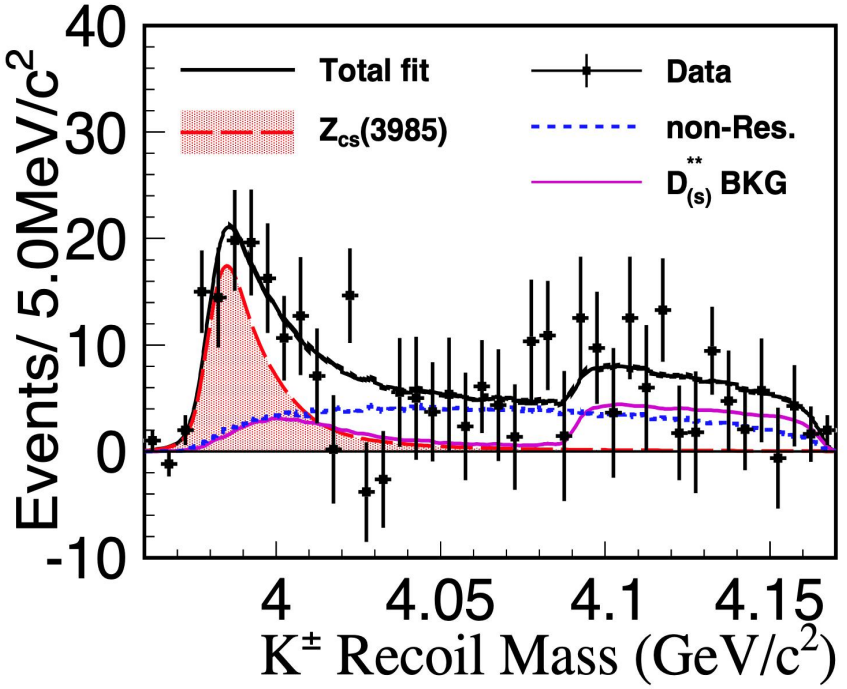
> 1 EB（10^9GB）





画图和拟合





功能丰富、性能强悍的工具集

- 数据分析：histograms, graphs, functions
- 数据读写I/O：row-wise, column-wise storage of any C++ object
- 统计工具(RooFit/RooStats)：rich modeling and statistical inference
- 数学函数： (e.g. Erf, Bessel), optimised math functions
- C++解释器：full language compliance
- 多变量分析(TMVA)：e.g. Boosted decision trees, Neural Nets, AI
- 高级图像功能：(2D, 3D, event display, detector design)
- 声明式分析方法：RDataFrame
- 其他：HTTP servering, JavaScript visualisation

温馨提示

- ROOT，C++功能极其丰富，无须也无法学完。
  有基础后，边用边学(面向Google编程)

- ROOT并不是万能的，或者说实现某些特性
  需要技巧

ROOT基础功能

安装/进入ROOT

- 操作系统：Linux(ubuntu,centos,...)，Mac，Win
- 安装：系统自带的包管理系统(apt,yum,dnf,...)
  或官方的预编译包，或在服务器(lxslc)上使用。
- 为了方便后续的讨论，ROOT工作环境如下，

```
> ssh -XYC user@lxslc711.ihep.ac.cn
> # login IHEP server w/o extra settings
> source /cvmfs/sft.cern.ch/lcg/views/LCG_102/x86_64-centos7-gcc8-opt/setup.sh
> # simple tests
> root --version  # output: ROOT Version: 6.26/04
```

ROOT两类使用场景

- 命令行(Prompt)，交互式

```
> root
```

```
root [0] double x = 0.5
(double) 0.50000000
root [1] int N = 30
(int) 30
root [2] double gs = 0;
root [3] .!ls
besfs  jgroup  jpsig  scrcfs
root [4]
```

- 代码(Macros/Scripts)，编程式

function, no **main**, same name as the file

```
void MacroName() {
    /* ......
    your lines of C++ codes
        ...... */
}
```

```
> root MacroName.C
```

Prompt

```
root [0] double x = 0.5;
root [1] int N = 30;
root [2] double gs = 0.0;
root [3] for (int i = 0; i != N; ++i) gs += pow(x, i)
root [4] std::abs(gs - (1.0 / (1.0 -x)))
(double) 1.8626451e-09
```

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \ldots = \sum_{n=0}^{\infty} x^n$$

- ✓ ROOT Prompt: 强大且功能完善的C++解释器
- ✓ 内置special command `root [0] .<command>`
  - 退出ROOT `root [0] .q`
  - 执行Shell命令 `root [0] .!<Shell_cmd>`
  - 载入脚本 `root [0] .L <file_name>`
  - 帮助信息 `root [0] .help or .?`

Macros/Scipts

- ✓一般的运行方式：

```
> root MacroName.C
```

- ✓Prompt内运行：

```
> root
root [0] .x MacroName.C
```

- ✓或者也可以先载入：

```
> root
root [0] .L MacroName.C
root [1] MacroName()
```

- ✓先编译再运行：

```
> root
root [0] .L MacroName.C+
root [1] MacroName()
```

Macros/Scipts 一点进阶

✓编译成独立的可执行程序(基于ROOT开发应用）：

```
> cat MacroApp.C
  /* ******
  MacroName() function codes
   * ******/

  int main() {
      MacroName();
      return 0;
  }
> g++ -o MacroApp MacroApp.C `root-config --cflags --libs`
> ./MacroApp
```

ROOT常用组件(Class)

ROOT做为C++开发的大型软件，常用组件以封装好的C++ Class的形式来展现：

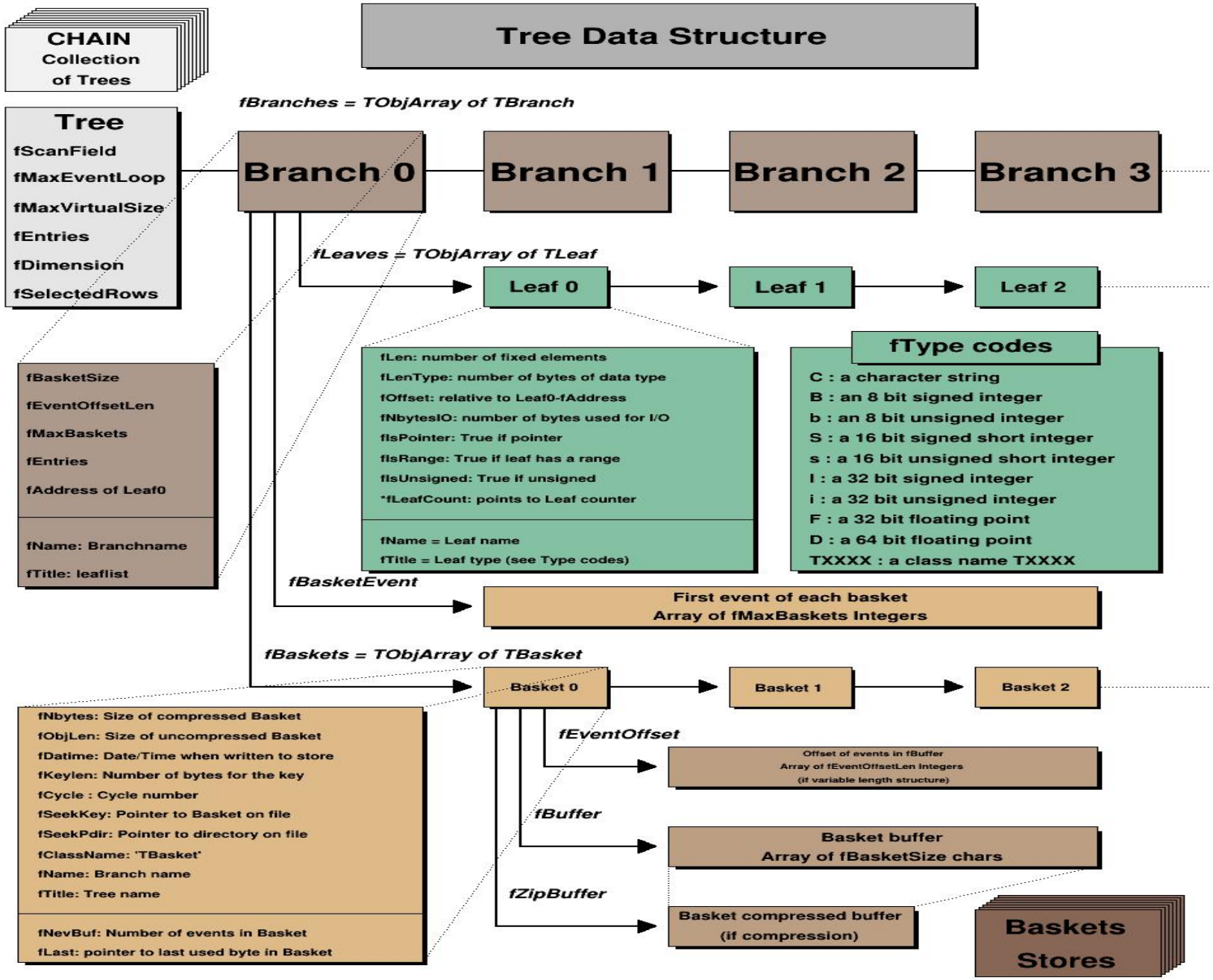
- 目前几乎几乎所有的Class都以大写字母**T**开头，
  - ✓TFile/**TTree**/TBranch
  - ✓TCanvas/**TH1D**/TGraphErrors
  - ✓TF1
  - ✓TLegend/TArrow/TLatex
- 例外：**RooFit/RooStats**，专门用于拟合相关的统计分析工具
- 未来，ROOT7将全部规范为大写字母**R**开头。

ROOT：根

Tree：树

Branch：枝

Leaf：叶

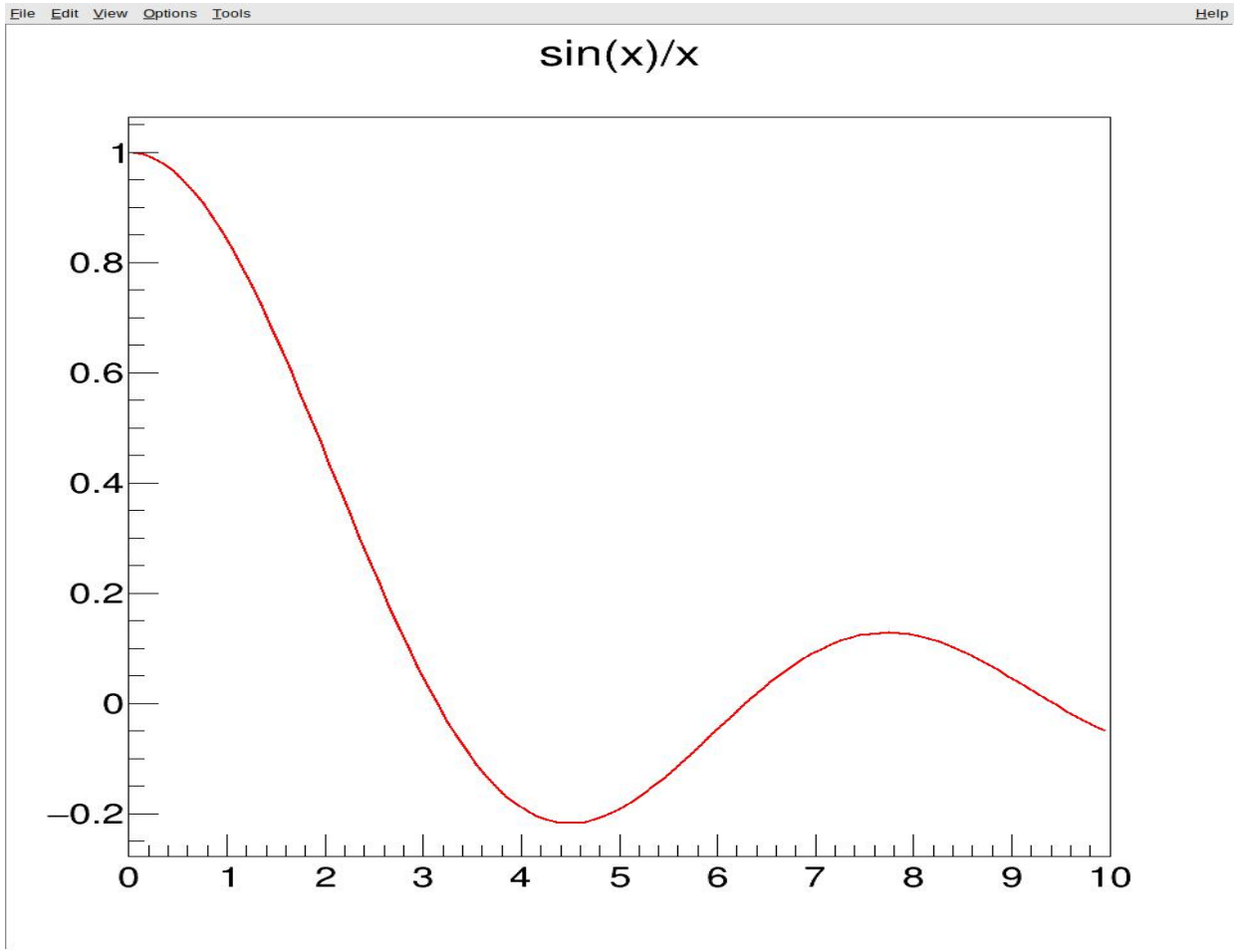


TF1

```
root [0] TF1 f1("f1", "sin(x)/x", 0.0, 10.0);
root [1] f1.Draw();
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```

- T: ROOT
- F: Function
- 1: 1-Dimension

✓ a string
✓ TFormula
✓ precompiled user f.

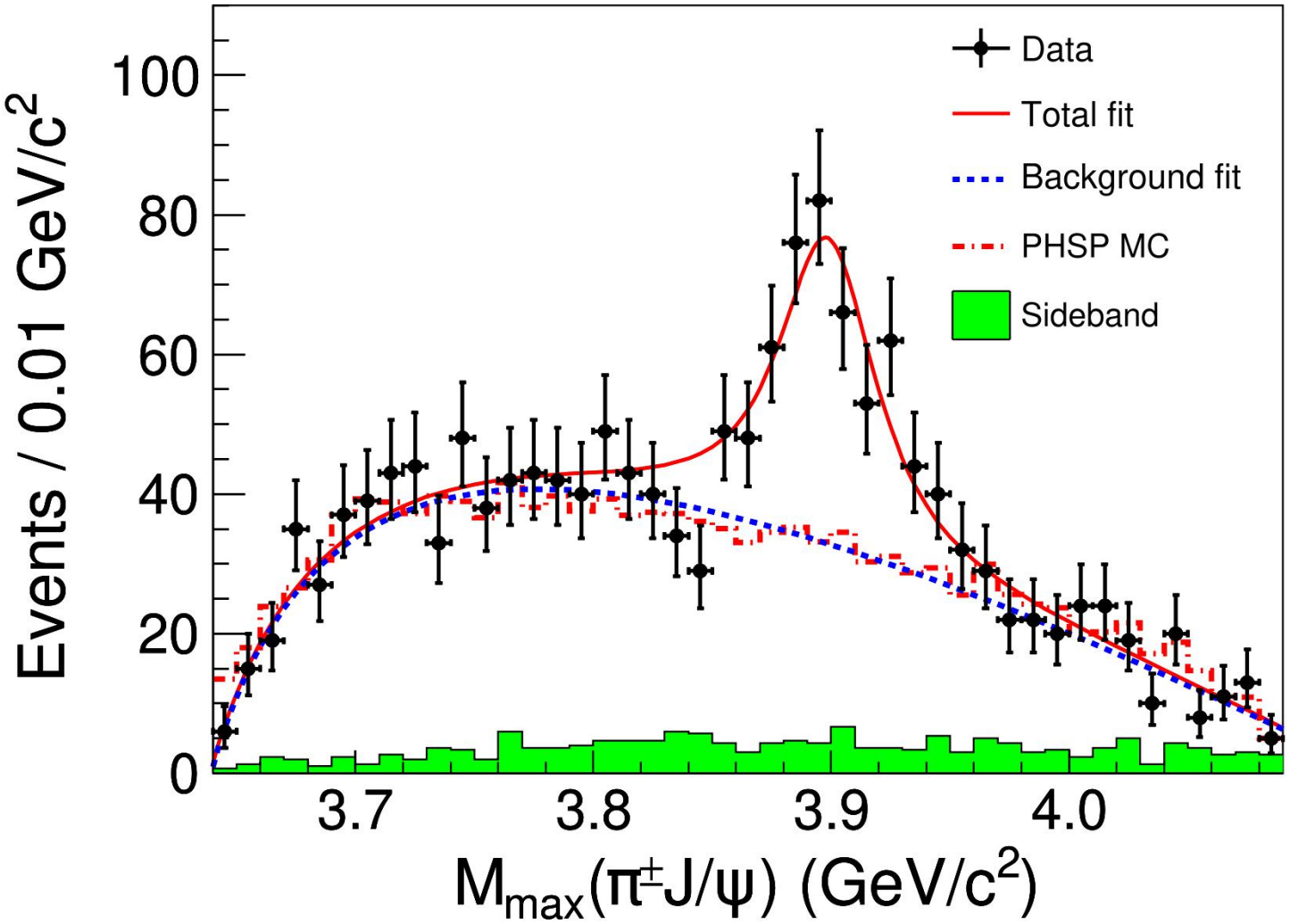


TH1D & TF1

- T: ROOT
- H: Histogram 直方图
- 1: 1-Dimension
- D: Double

✓ 数据简化的直观形式

✓ 分bin展示的方式



TH1D & TF1

name, unique

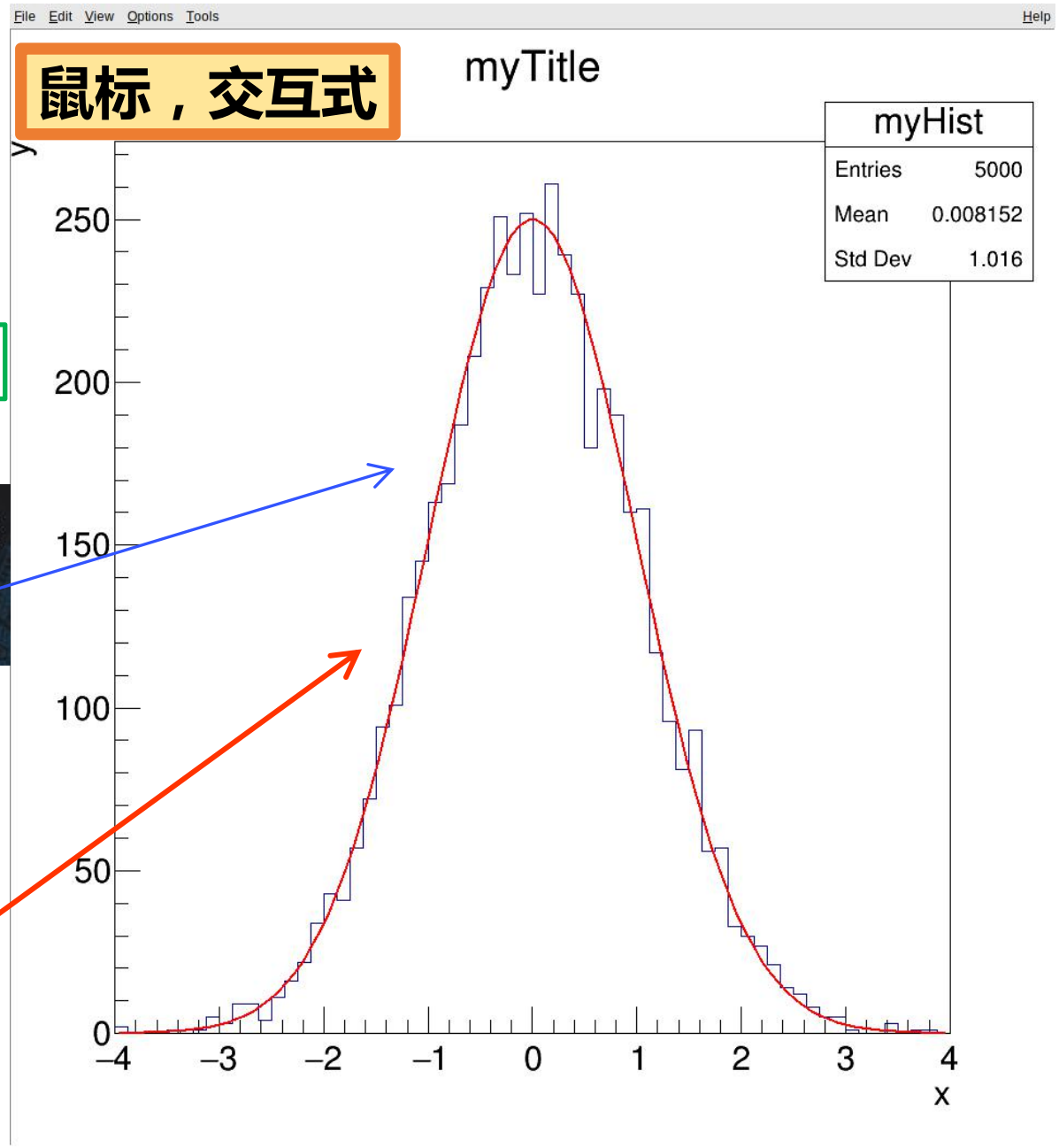
title, x-y axis

bins, min, max

```
root [0] TH1D h("myHist", "myTitle;x;y", 64, -4.0, 4.0);
root [1] h.FillRandom("gaus")
root [2] h.Draw()
```

pre-defined function name

```
root [3] TF1 f("g", "gaus", -8.0, 8.0);
root [4] f.SetParameters(250, 0.0, 1.0)
root [5] f.Draw("SAME")
```



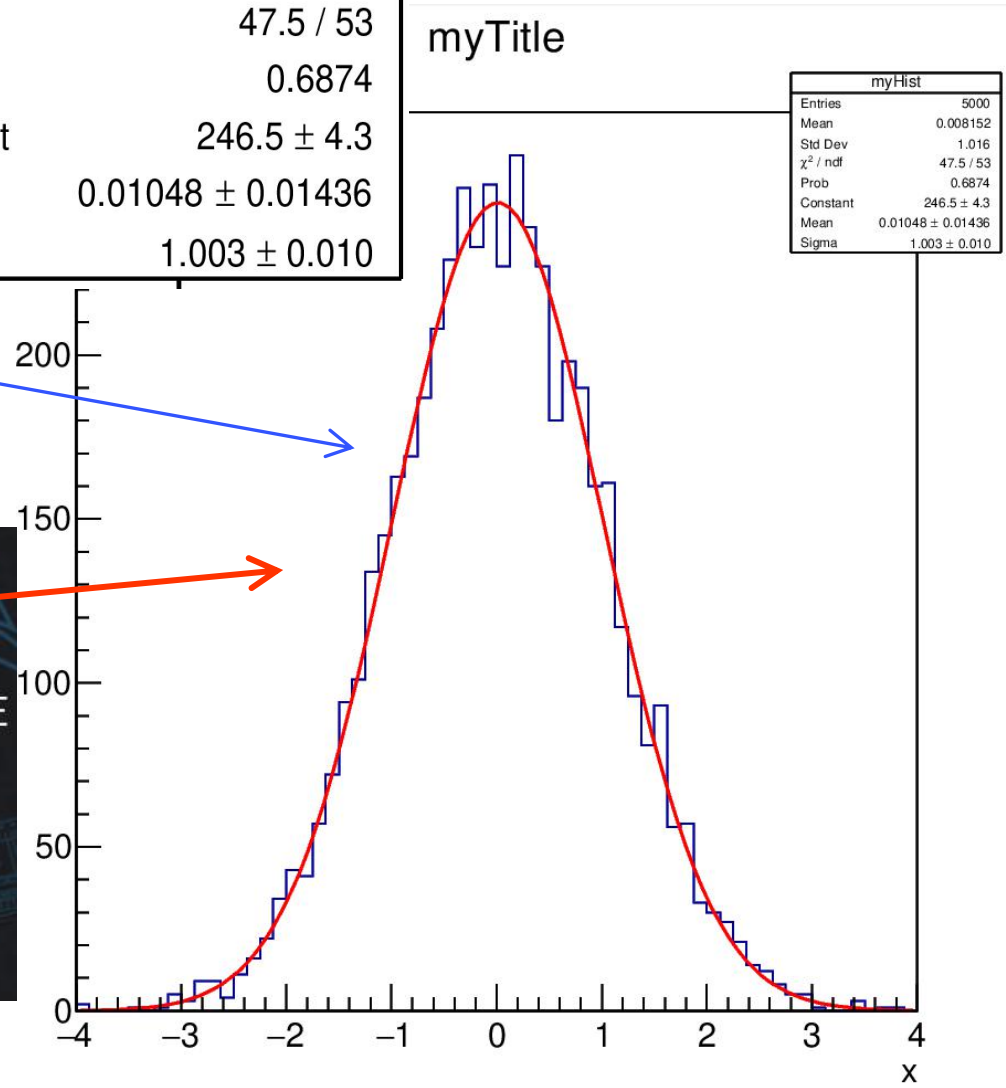
TH1D & Fitting

pre-defined function name

```
root [0] TH1D h("myHist", "myTitle;x;y", 64, -4.0, 4.0);
root [1] h.FillRandom("gaus")
root [2] h.Draw()
```

```
root [3] gStyle->SetOptFit(1111)
root [4] auto r = h.Fit("gaus")
 FCN=47.4997 FROM MIGRAD    STATUS=CONVERGED      53 CALLS          54 TOTAL
                     EDM=8.44224e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     2.46469e+02   4.31494e+00   1.19094e-02  -2.44811e-05
   2  Mean         1.04782e-02   1.43576e-02   4.87656e-05  -6.34020e-03
   3  Sigma        1.00315e+00   1.03818e-02   9.45504e-06  -2.70309e-02
```

myHist	
Entries	5000
Mean	0.008152
Std Dev	1.016
χ^2 / ndf	47.5 / 53
Prob	0.6874
Constant	246.5 ± 4.3
Mean	0.01048 ± 0.01436
Sigma	1.003 ± 0.010



myHist	
Entries	5000
Mean	0.008152
Std Dev	1.016
χ^2 / ndf	47.5 / 53
Prob	0.6874
Constant	246.5 ± 4.3
Mean	0.01048 ± 0.01436
Sigma	1.003 ± 0.010

myHist	
Entries	50000
Mean	−0.0007154
Std Dev	0.9997
χ^2 / ndf	58.11 / 61
Prob	0.5814
Constant	2499 ± 13.6
Mean	-0.000537 ± 0.004463
Sigma	0.9967 ± 0.0031

数据分析中，“统计量就是王道”。ROOT擅长处理大统计量！

“E S”

最小二乘法

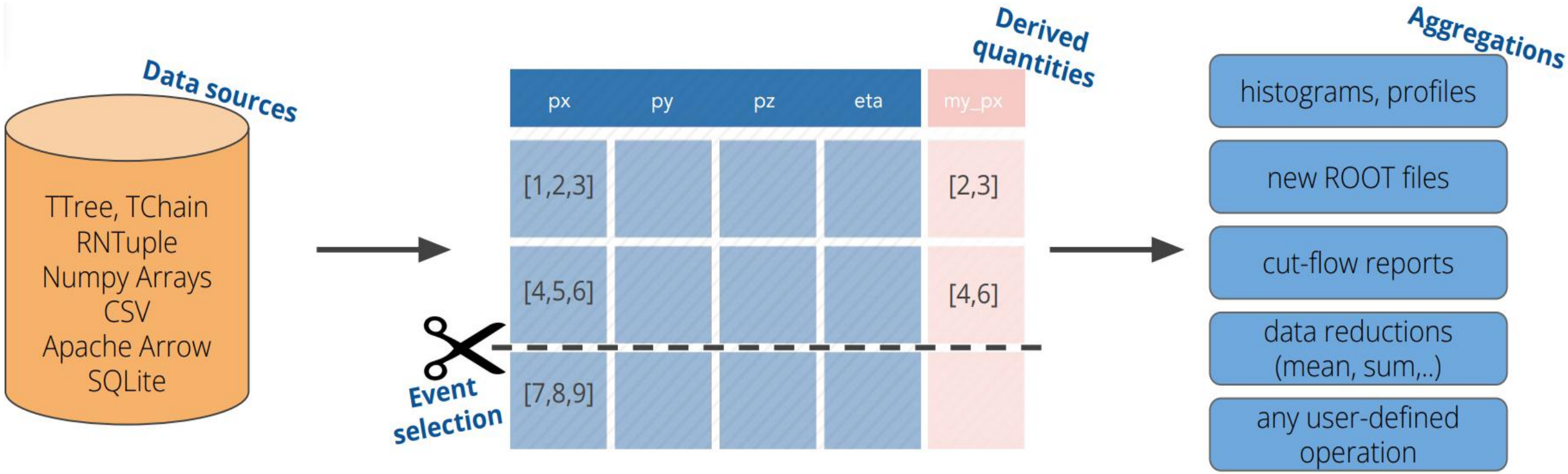
```
root [8] r = h.Fit("gaus", "E S")
 FCN=58.1081 FROM MINOS     STATUS=SUCCESSFUL     20 CALLS         130 TOTAL
                     EDM=2.47365e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                          STEP         FIRST
  NO.   NAME      VALUE            ERROR                 SIZE      DERIVATIVE
   1  Constant     2.49888e+03   1.36482e+01  -2.47671e-02   4.96181e-06
   2  Mean        -5.36955e-04   4.46271e-03   1.08929e-06  -1.43636e-03
   3  Sigma        9.96702e-01   3.12445e-03   3.12445e-03  -3.85305e-03
(TFitResultPtr &)
****************************************
Minimizer is Minuit / Migrad
Chi2                          =      58.1081
NDf                           =           61
Edm                           =  2.47365e-08
NCalls                        =          130
Constant                      =      2498.88   +/-   13.6482      -13.6358     +13.6605     (Minos)
Mean                          = -0.000536955   +/-   0.00446271   -0.00446235  +0.00446317  (Minos)
Sigma                         =     0.996702   +/-   0.00312445   -0.00312931  +0.00311956  (Minos)
```

极大似然法

“E L S”

```
root [4] auto r = h.Fit("gaus", "E L S")
 FCN=26.8031 FROM MINOS     STATUS=SUCCESSFUL     20 CALLS         134 TO
                     EDM=3.9281e-09    STRATEGY= 1      ERROR MATRIX ACCU
  EXT PARAMETER                                          STEP         FIRST
  NO.   NAME      VALUE            ERROR                 SIZE      DERIVATIVE
   1  Constant     2.49416e+03   1.36768e+01   2.45206e-02  -2.50141e-06
   2  Mean        -6.98239e-04   4.47340e-03   2.10652e-08   1.63996e-07
   3  Sigma        9.99753e-01   3.17583e-03   3.17583e-03  -1.34813e-06
                               ERR DEF= 0.5
(TFitResultPtr &)
****************************************
Minimizer is Minuit / Migrad
MinFCN                        =      26.8031
Chi2                          =      59.1375
NDf                           =           61
Edm                           =   3.9281e-09
NCalls                        =          134
Constant                      =      2494.16   +/-   13.6768      -13.6486     +13.7051     (Minos)
Mean                          = -0.000698239   +/-   0.0044734    -0.00447342  +0.00447342  (Minos)
Sigma                         =     0.999753   +/-   0.00317583   -0.00316717  +0.00318452  (Minos)
```

物理实验中的数据分析流程



创建Tree ⟶ 写入数据 ⟶ 存盘

cond_data.C

```
1  void cond_data()
2  {
3      TFile ofile("./cond_data.root", "RECREATE");
4
5      TTree cond_data("cond_data", "Example");
6
7      double pot, cur, temp, pres;
8      cond_data.Branch("Potential", &pot, "Potential/D");
9      cond_data.Branch("Current", &cur, "Current/D");
10     cond_data.Branch("Temperature", &temp, "Temperature/D");
11     cond_data.Branch("Pressure", &pres, "Pressure/D");
12
13     for (int i = 0; i != 1000000; ++i) {
14         pot  = gRandom->Uniform(0.0, 10.0) * gRandom->Gaus(1.0, 0.01);
15         temp = gRandom->Uniform(250.0, 350.0) + gRandom->Gaus(0.0, 0.3);
16         pres = gRandom->Uniform(0.5, 1.5) * gRandom->Gaus(1.0, 0.02);
17         cur  = pot / (10.0 + 0.05 * (temp - 300.0) - 0.2 * (pres - 1.0)) *
18               gRandom->Gaus(1.0, 0.01);
19
20         cond_data.Fill();
21     }
22
23     cond_data.Write();
24     ofile.Close();
25 }
```

创建文件（覆盖）并打开

创建Tree（name，title）

“桥梁”

声明变量和Branch，并做匹配

模拟数据产生

将单行数据“填入”Tree

写入磁盘并关闭文件

数据类型

type	size	C++	identifier
signed integer	32 bit	int	I
	64 bit	long	L
unsigned integer	32 bit	unsigned int	i
	64 bit	unsigned long	l
floating point	32 bit	float	F
	64 bit	double	D
boolean	-	bool	O

```
array, vector, map, HepVector, Foo, ...
```

创建Tree ⟶ 写入数据 ⟶ 存盘

```
-bash-4.2$ time root -l -b -q cond_data.C

Processing cond_data.C...

real    0m6.975s
user    0m2.743s
sys     0m1.582s
-bash-4.2$ ls -alh
total 29M
drwxr-xr-x  2 sunhk physics 4.0K Aug 17 09:56 .
drwxr-xr-x 16 sunhk physics 4.0K Aug 17 09:55 ..
-rw-r--r--  1 sunhk physics  822 Aug 17 09:55 cond_data.C
-rw-r--r--  1 sunhk physics  29M Aug 17 09:56 cond_data.root
```

- `time` bash内置记时命令
- `root` 执行参数
  - `-l` 不显示banner
  - `-b` 批量模式，无图形化
  - `-q` 执行完就退出

读取ROOT，“观察”数据 I

```
-bash-4.2$ root -l cond_data.root
root [0]
Attaching file cond_data.root as _file0...
(TFile *) 0x2a0a900
root [1] .ls
TFile**         cond_data.root
 TFile*         cond_data.root
  KEY: TTree    cond_data;2     Example [current cycle]
  KEY: TTree    cond_data;1     Example [backup cycle]
```

1）获取事例数

```
root [2] cond_data->GetEntries()
(long long) 1000000
```

2）展示单事例

```
root [3] cond_data->Show(0)
======> EVENT:0
 Potential          = 9.95395
 Current            = 0.81225
 Temperature        = 344.711
 Pressure           = 1.45582
```

3）获取事例数（cut）

```
root [6] cond_data->GetEntries("Potential>9.0")
(long long) 100061
root [7] cond_data->GetEntries("Potential>9.0 && Current < 0.8")
(long long) 12485
```

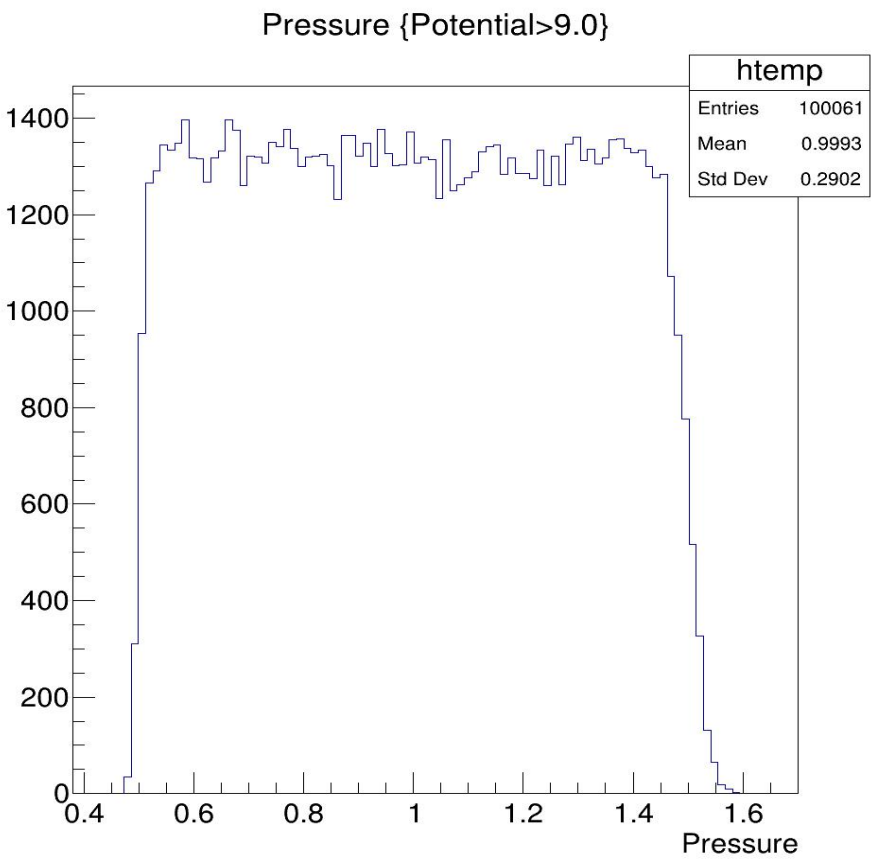
读取ROOT，“观察”数据 II

4）Tree 总览

```
root [9] cond_data->Print()
******************************************************************************
*Tree    :cond_data : Example
*Entries :  1000000 : Total =        32104849 bytes  File  Size =   30171811 *
*        :          : Tree compression factor =   1.06                       *
******************************************************************************
*Br    0 :Potential : Potential/D                                             *
*Entries :  1000000 : Total  Size=    8026684 bytes  File Size  =    7676137 *
*Baskets :      251 : Basket Size=    3200512 bytes  Compression=    1.04     *
*.............................................................................*
*Br    1 :Current   : Current/D                                               *
*Entries :  1000000 : Total  Size=    8026174 bytes  File Size  =    7617437 *
*Baskets :      251 : Basket Size=    3200000 bytes  Compression=    1.05     *
*.............................................................................*
*Br    2 :Temperature : Temperature/D                                         *
*Entries :  1000000 : Total  Size=    8026194 bytes  File Size  =    7297493 *
*Baskets :      251 : Basket Size=    3200512 bytes  Compression=    1.10     *
*.............................................................................*
*Br    3 :Pressure  : Pressure/D                                              *
*Entries :  1000000 : Total  Size=    8025429 bytes  File Size  =    7572128 *
*Baskets :      251 : Basket Size=    3200000 bytes  Compression=    1.06     *
*.............................................................................*
```

5）画直方图（1维）

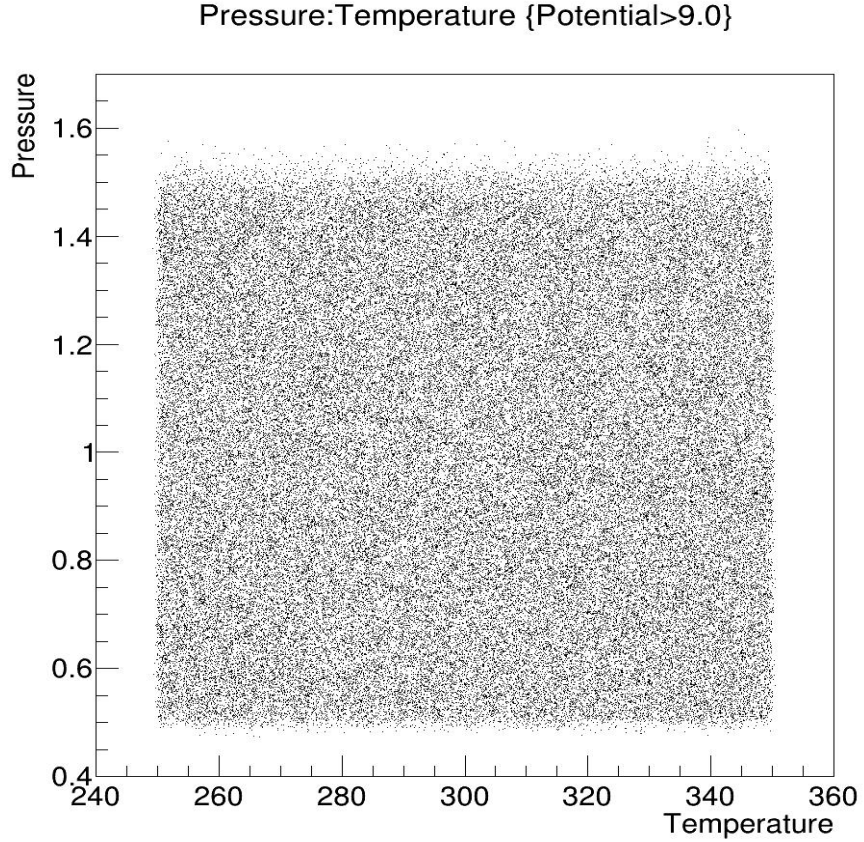
```
root [11] cond_data->Draw("Pressure", "Potential>9.0")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
(long long) 100061
```



读取ROOT，“观察”数据 III

6）画直方图（2维）

```
root [12] cond_data->Draw("Pressure:Temperature", "Potential>9.0")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
(long long) 100061
```



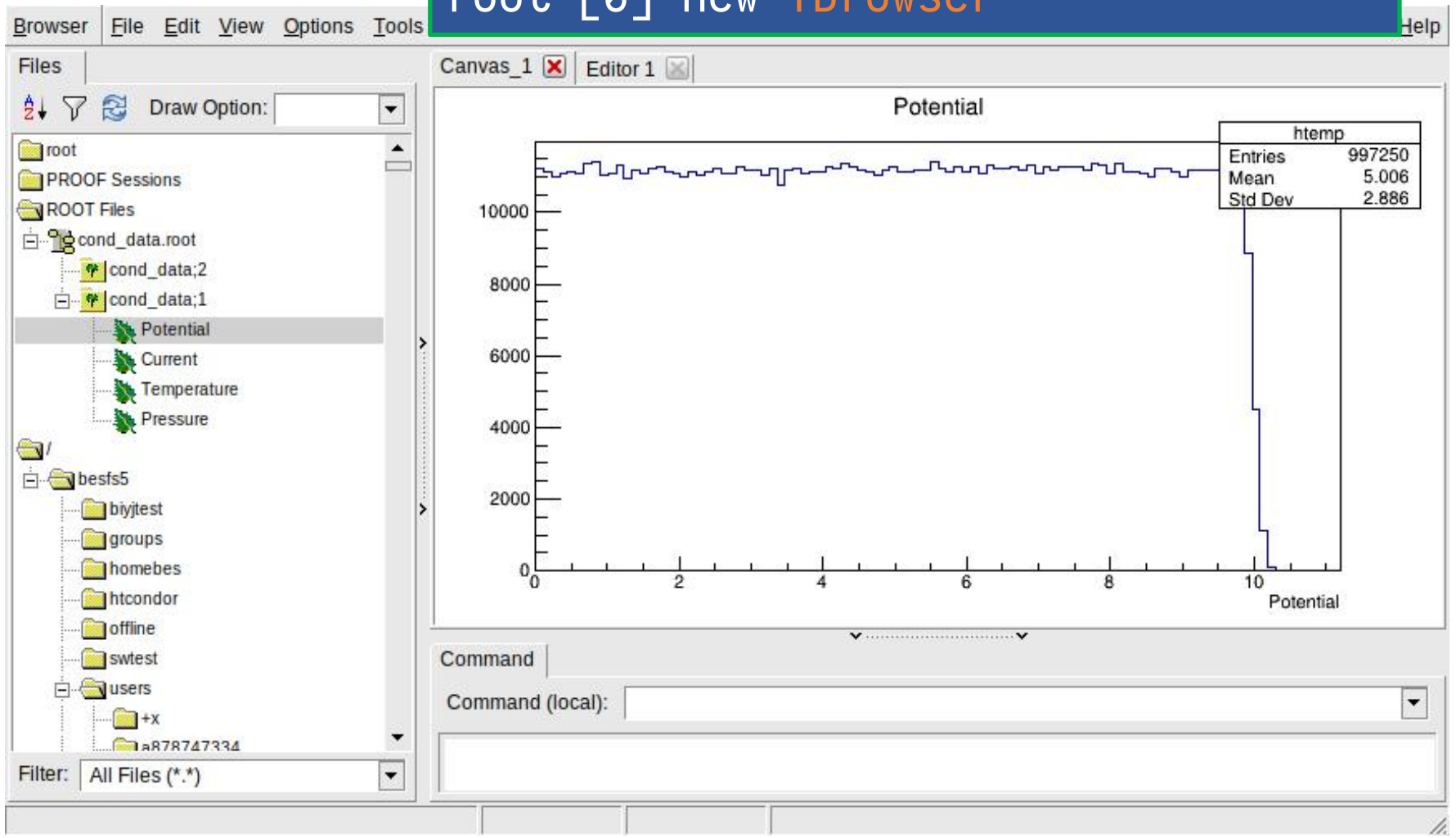
7）多维数据展示

```
root [13] cond_data->Scan("Current:Pressure", "Current>1.0")
************************************
*    Row   *   Current *  Pressure *
************************************
*       27 * 1.2605584 * 1.0810390 *
*       64 * 1.2805236 * 0.8179486 *
*       75 * 1.2557656 * 1.0235446 *
*       77 * 1.0336579 * 0.5346831 *
*      105 * 1.1594679 * 0.7228008 *
*      111 * 1.2731033 * 1.4522743 *
*      116 * 1.1096525 * 0.7128843 *
*      149 * 1.2070604 * 1.1434670 *
*      174 * 1.0594486 * 1.2826776 *
*      193 * 1.1475714 * 0.8946843 *
```

读取ROOT，“观察”数据 IV

8）图形化、交互式

```
> root -l --web=off cond_data.root
root [0] new TBrowser
```



脚本画图，基础版

read_cond.C

```
1  void read_cond()
2  {
3      TChain in_chain("cond_data");
4      in_chain.Add("cond_data*.root"); // add files, wildcards work!
5
6      double pot, cur, temp, pres;
7      in_chain.SetBranchAddress("Potential", &pot);
8      in_chain.SetBranchAddress("Current", &cur);
9      in_chain.SetBranchAddress("Temperature", &temp);
10     in_chain.SetBranchAddress("Pressure", &pres);
11
12     TH1D* h = new TH1D("myHist", "Current;Current;Events", 100, 0.0, 1.5);
13     for (int irow = 0; irow != in_chain.GetEntries(); ++irow) {
14         in_chain.GetEntry(irow);
15         if (pot > 8.0 && temp < 300.0 && pres > 1.0) h->Fill(cur);
16     }
17
18     TCanvas* c1 = new TCanvas("myCanvas", "Current Histo", 900, 600);
19     h->SetLineWidth(2);
20     h->SetLineColor(EColor::kRed);
21     h->Draw();
22
23     c1->Print("current.pdf");
24     delete h;
25     delete c1;
26 }
```

打开root文件（多）

“桥梁”

声明变量和Branch，并做匹配

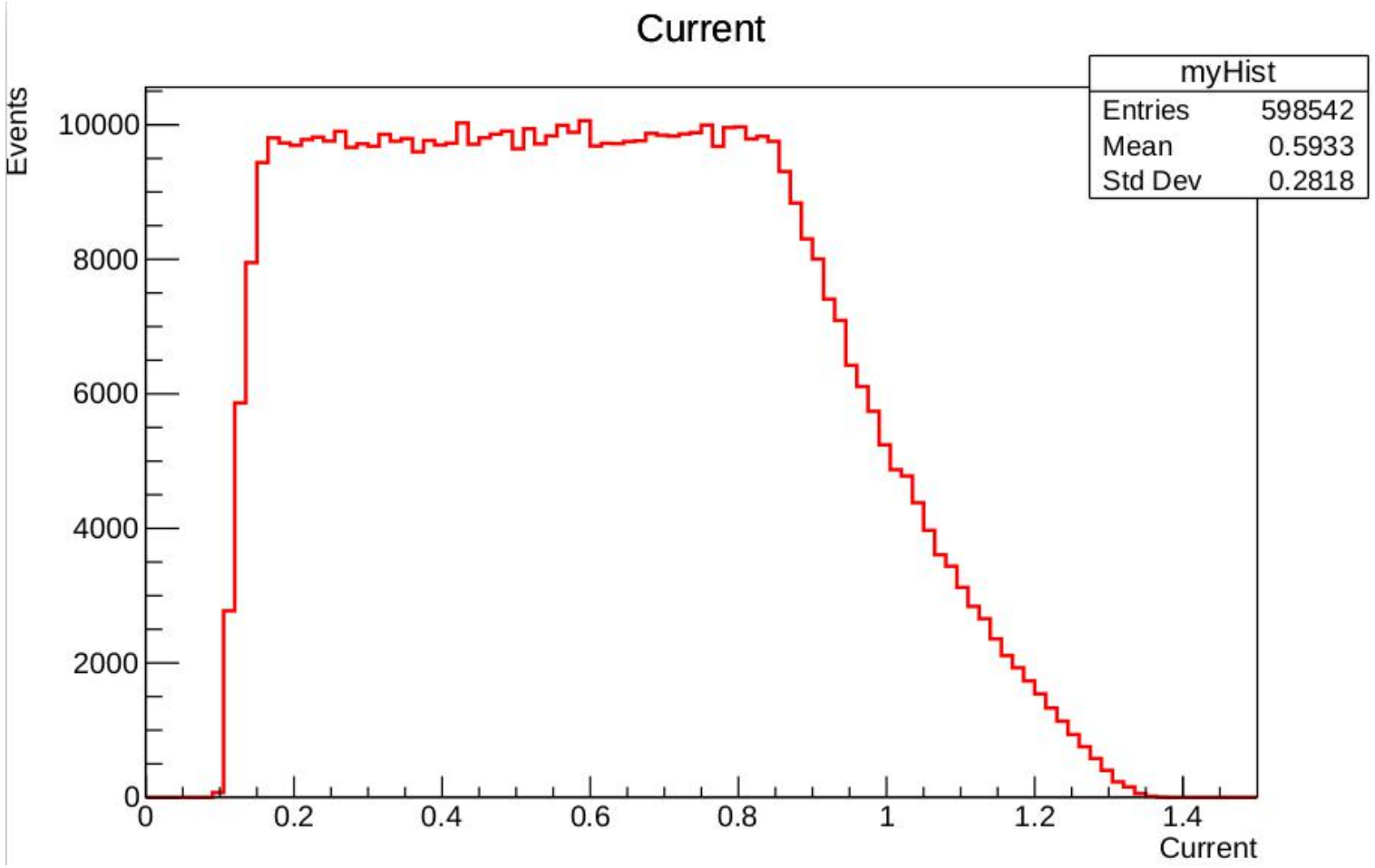
声明Histogram并填入数据

声明画布，设置样式，画图

保存图片，清理

```
-bash-4.2$ root -l -b -q read_cond.C

Processing read_cond.C...
Info in <TCanvas::Print>: pdf file current.pdf has been created
-bash-4.2$ evince current.pdf
```



脚本画图，高级版

```
{
// create template class for Selector to run on a tree
//////////////////////////////////////////////////////////
//
// open root file containing the Tree
    TFile f("conductivity_experiment.root");
// create TTree object from it
    TTree *t; f.GetObject("cond_data",t);
// this generates the files MySelector.h and MySelector.C
    t->MakeSelector("MySelector");
}
```

```
> TChain *ch=new TChain("cond_data", "Chain for Example N-Tuple");
> ch->Add("conductivity_experiment*.root");
> ch->Process("MySelector.C+");
```

ROOT进阶功能

RooFit

- ✓RooFit提供了极其丰富的功能来构建PDF(model)，
  - ➢预定义了丰富的基础组件
    （Gaussian, ArgusBG, Chebychev, Breit-Wigner, etc.）
  - ➢并可以在基础组件上通过加、乘、卷积等运算自由组合
  - ➢也支持从Histogram中抽取PDF
- ✓RooFit为所有的PDF，都提供了，
  - ➢极大似然法拟合
  - ➢toyMC生成器(generator)
  - ➢图形可视化

RooFit拟合 I

```
using namespace RooFit;

void rf_ex()
{
    // Declare observable x
    RooRealVar x("x", "x", 0, 10);

    // Create two Gaussian PDFs for signal model
    RooRealVar mean("mean", "mean of gaussians", 5.0, 4.5, 5.5);
    RooRealVar sigma1("sigma1", "width of gaussians", 0.5, 0.1, 2.0);
    RooRealVar sigma2("sigma2", "width of gaussians", 1.0, 0.1, 2.0);

    RooGaussian sig1("sig1", "Signal component 1", x, mean, sigma1);
    RooGaussian sig2("sig2", "Signal component 2", x, mean, sigma2);

    // Build Chebychev polynomial pdf
    RooRealVar a0("a0", "a0", 0.8, 0.2, 1.2);
    RooRealVar a1("a1", "a1", 0.2, 0.0, 1.0);
    RooChebychev bkg("bkg", "Background", x, RooArgSet(a0, a1));

    // Sum the signal components into a composite signal pdf
    RooRealVar sig1frac("sig1frac", "fraction of component 1 in signal", 0.8,
                        0.0, 1.0);
    RooAddPdf sig("sig", "Signal", RooArgList(sig1, sig2), sig1frac);
```

- **RooRealVar**：自变量Variable

  name, title, mean, min, max

- **RooGaussian**：高斯分布

  name, title, var, μ, σ

- **RooChebychev**：多项式（n阶）

  name, title, var, RooArgSet(n)

- **RooAddPdf**：用加法来组合PDF

  name, title, RooArgList(n), frac

RooFit拟合 II

```
// Sum the composite signal and background
RooRealVar bkgfrac("bkgfrac", "fraction of background", 0.5, 0.1, 1.0);
RooAddPdf model("model", "g1+g2+a", RooArgList(bkg, sig), bkgfrac);

// Generate a data sample of 1000 events in x from model
RooDataSet* data = model.generate(x, 2500);
/* RooCmdArg rf = ImportFromFille("./cond_data.root", "cond_data");
 * RooDataSet* data = new RooDataSet("name", "title", RooArgSet(x), rf);
 */

// Fit model to data
RooFitResult* r = model.fitTo(*data, NumCPU(4), Save());

// Plot data and PDF overlaid
RooPlot* xframe =
    x.frame(Title("Example of composite pdf=(sig1+sig2)+bkg"));
data->plotOn(xframe);
model.plotOn(xframe);

// Overlay the background component of model with a dashed line
model.plotOn(xframe, Components(bkg), LineStyle(kDashed),
             LineColor(EColor::kRed));

// Overlay the signal components of model with a dotted line
model.plotOn(xframe, Components(RooArgSet(sig1, sig2)), LineStyle(kDotted),
             LineColor(EColor::kGreen + 3), LineWidth(3));
```

- **RooDataSet**：需要拟合的样本
  1. generate，由PDF生成(toy MC)
  2. ImportFromFile，读取root文件
- **RooFitResult**：拟合结果(Save())
  1. NumCPU，自动化并行计算
  2. Save，保存拟合结果
- **x.frame**：画图的frame来自于自变量
- **plotOn**：在frame上分别绘制

  Components，Line{Style,Color,...)

RooFit拟合 III

```
// Print structure of composite pdf
r->Print("V");

// Draw the frame on the canvas
TCanvas* c1 = new TCanvas("RooFit", "RooFit Example", 800, 600);
gPad->SetLeftMargin(0.15);
xframe->GetYaxis()->SetTitleOffset(1.4);
xframe->Draw();

c1->Print("./fit.pdf");
delete c1;
```

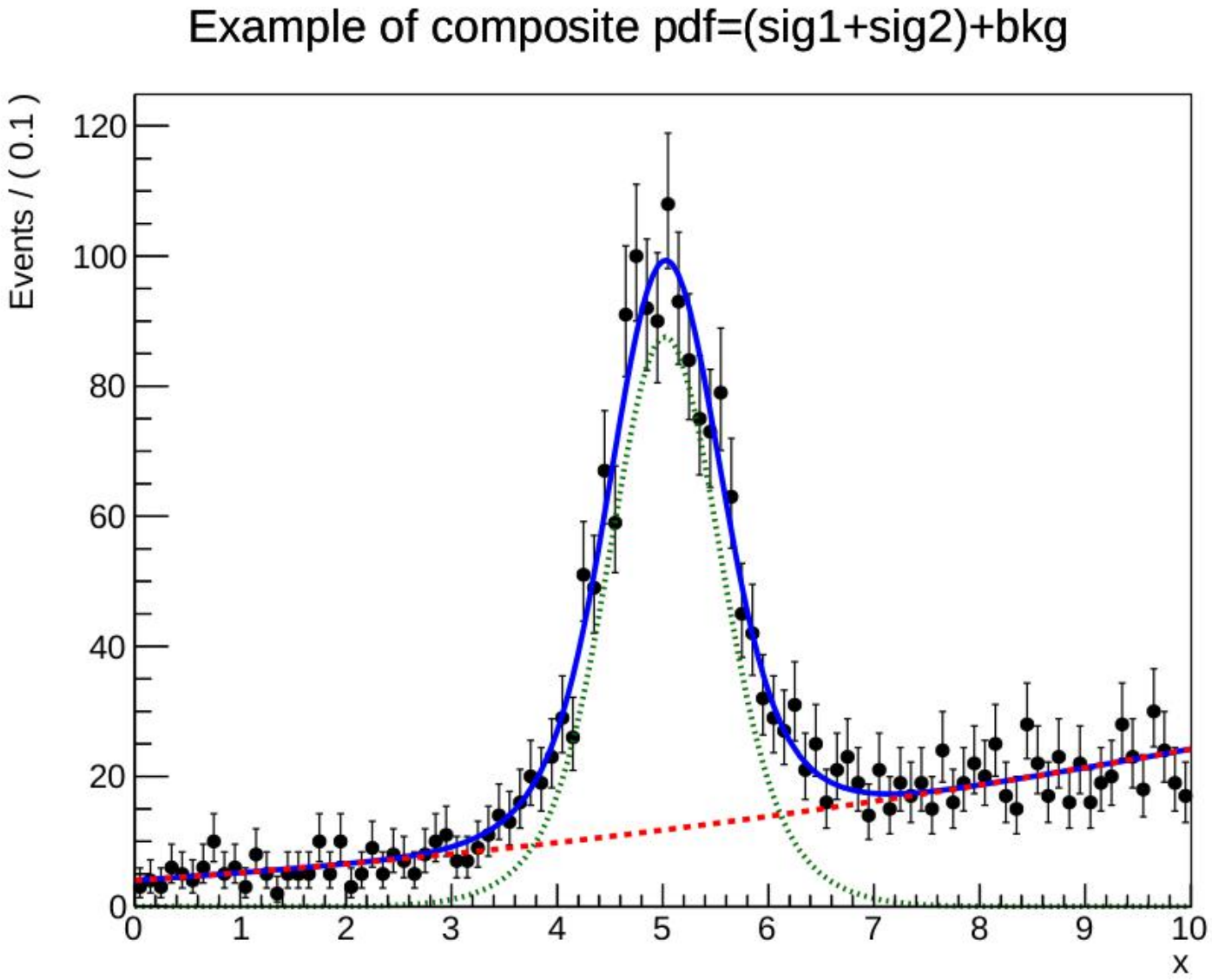
- **Print**：拟合结果详细展示
- **Draw**：将绘制好的frame画在画布上
- 保存图片并清理。

```
RooFitResult: minimized FCN value: 4808.73, estimated distance to minimum: 2.39774e-05
              covariance matrix quality: Full, accurate covariance matrix
              Status : MINIMIZE=0 HESSE=0

  Floating Parameter  InitialValue    FinalValue +/-  Error     GblCorr.
--------------------  ------------  --------------------------  --------
                  a0    8.0000e-01    7.8080e-01 +/-  4.44e-02  <none>
                  a1    2.0000e-01    9.1578e-02 +/-  7.56e-02  <none>
             bkgfrac    5.0000e-01    5.0140e-01 +/-  2.55e-02  <none>
                mean    5.0000e+00    5.0224e+00 +/-  2.14e-02  <none>
            sig1frac    8.0000e-01    6.3904e-01 +/-  3.13e-01  <none>
              sigma1    5.0000e-01    4.8349e-01 +/-  7.65e-02  <none>
              sigma2    1.0000e+00    8.2294e-01 +/-  2.74e-01  <none>
```

```
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5      6      7
        1  0.47707   1.000  0.465 -0.379 -0.081  0.141  0.107  0.280
        2  0.79822   0.465  1.000 -0.763 -0.010  0.211  0.165  0.484
        3  0.85486  -0.379 -0.763  1.000 -0.048 -0.274 -0.214 -0.582
        4  0.14837  -0.081 -0.010 -0.048  1.000  0.019  0.041  0.026
        5  0.98426   0.141  0.211 -0.274  0.019  1.000  0.934  0.885
        6  0.95467   0.107  0.165 -0.214  0.041  0.934  1.000  0.754
        7  0.96855   0.280  0.484 -0.582  0.026  0.885  0.754  1.000
```

RooFit拟合 IV


Example of composite pdf=(sig1+sig2)+bkg
Events / (0.1)
120
100
80
60
40
20
0
0
1
2
3
4
5
6
7
8
9
10
x

TMVA, Tools for Multivariate Analysis

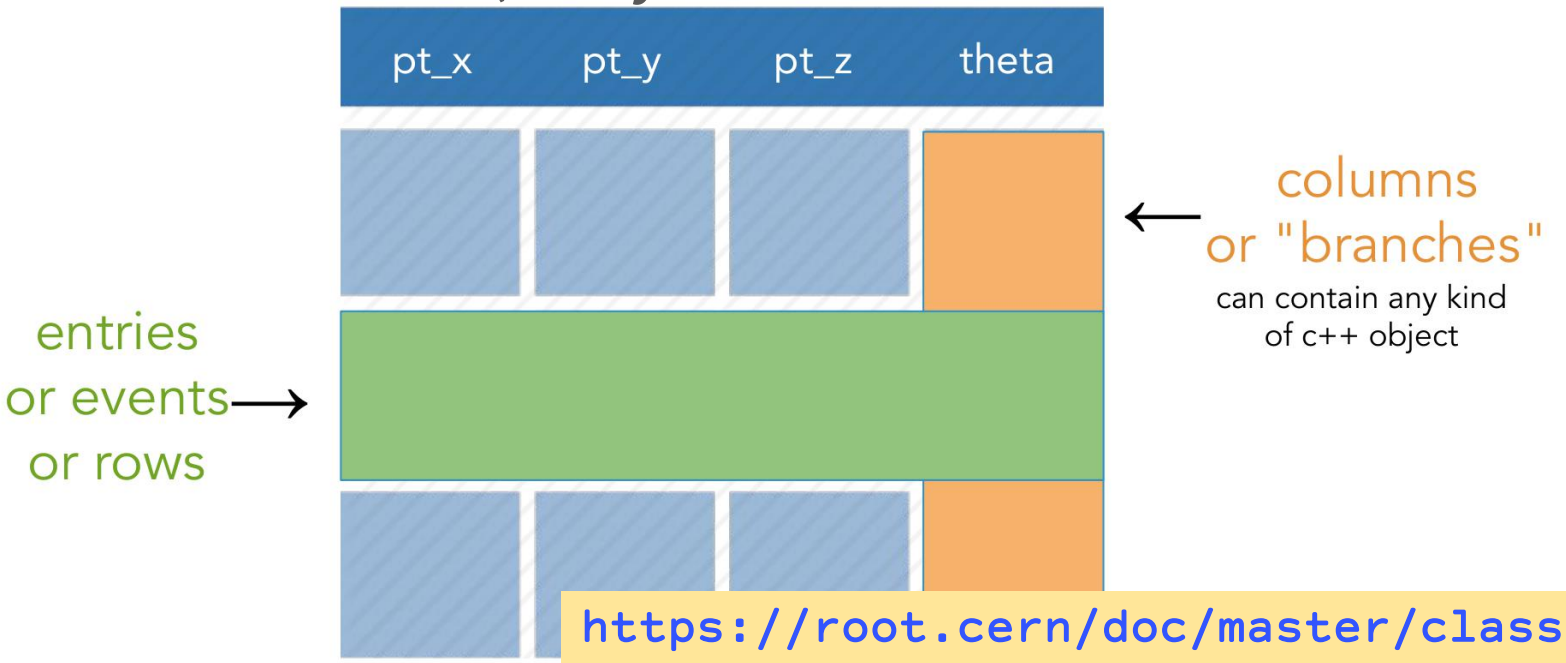
❖TMVA，多变量分析工具包。利用多种变量信息做为输入，从大样本中发现信号（pattern）：分类、学习!

- Neural networks, Deep networks 神经网络，深度学习网络
- Multilayer perceptron 多层感知机
- Boosted/Bagged decision trees 决策树
- Function discriminant analysis (FDA) 判别函数分析
- Multidimensional probability density estimation (PDE - range-search approach) 多维概率密度估计
- Multidimensional k-nearest neighbour classifier 多维K近邻(KNN)算法
- Predictive learning via rule ensembles (RuleFit) 基于规则拟合的预测学习
- Projective likelihood estimation (PDE approach) 投影似然估计
- Rectangular cut optimisation 矩形切割优化
- Support Vector Machine (SVM) 支持向量机

PyTorch

RDataFrame(RDF)

- ROOT中以`TTree`来存储的，是“列式数据集”(Columnar Datasets)
  1. 数据集被称作“树”(Tree)，每一列被称作“枝”(Branch)
  2. 不同的列可以存储不同类型的数据
  3. 能够支持任意类型的对象(object)
  4. 一行就是一项(entry，在高能物理里也被称作事例event)



`https://root.cern/doc/master/classROOT_1_1RDataFrame.html`

RDataFrame 对比

```cpp
void read_cond()
{
    TChain in_chain("cond_data");
    in_chain.Add("cond_data*.root"); // add files, wildcards work!

    double pot, cur, temp, pres;
    in_chain.SetBranchAddress("Potential", &pot);
    in_chain.SetBranchAddress("Current", &cur);
    in_chain.SetBranchAddress("Temperature", &temp);
    in_chain.SetBranchAddress("Pressure", &pres);

    TH1D* h = new TH1D("myHist", "Current;Current;Events", 100, 0.0, 1.5);
    for (int irow = 0; irow != in_chain.GetEntries(); ++irow) {
        in_chain.GetEntry(irow);
        if (pot > 8.0 && temp < 300.0 && pres > 1.0) h->Fill(cur);
    }

    TCanvas* c1 = new TCanvas("myCanvas", "Current Histo", 900, 600);
    h->SetLineWidth(2);
    h->SetLineColor(EColor::kRed);
    h->Draw();

    c1->Print("current.pdf");
    delete h;
    delete c1;
}
```

```cpp
void read_cond_rdf()
{
    ROOT::EnableImplicitMT();
    ROOT::RDataFrame df("cond_data", "./cond_data*.root");

    auto h = df.Filter("Potential>1.2")
               .Filter("Temperature<335.0")
               .Filter("Pressure>0.7")
               .Histo1D({"myHist", "Current;Current;Events", 100, 0.0, 1.5},
                        "Current");

    TCanvas* c1 = new TCanvas("myCanvas", "Current Histo", 900, 600);
    h->SetLineWidth(2);
    h->SetLineColor(EColor::kRed);
    h->Draw();

    c1->Print("current_rdf.pdf");
}
```

越复杂的分析，对比越明显

RDataFrame 官方示例

```
ROOT::EnableImplicitMT();                    // Run a multi-thread event loop
ROOT::RDataFrame df(dataset);                // on this (ROOT, CSV, ...) dataset
auto df2 = df.Filter("x > 0")                // only accept events for which x > 0
             .Define("r2", "x*x + y*y");     // define r2 = x² + y²
auto rHist = df2.Histo1D("r2");              // plot r2 for events that pass the cut
df2.Snapshot("newtree", "out.root");         // write the skimmed data and r2
                                             //   to a new ROOT file
```

Six lines of code! Most RDF analyses are just more of the same.
User-defined computations are naturally made modular.

`Range,Count,Report,Graph,Profile,Take,Vary,...`

RDF@ROOT workshop, 9/5/2022

总结

- ✓ ROOT是高能物理的必备工具
- ✓ 功能强大，应用广泛。快速入门，边用边学。
- ✓ 拥抱新事物：
  - RooFit/RooStats(`Workspace, factory`)
  - `RDataFrame`，未来分析框架的基础
  - `PyROOT`发展迅速，告别`C++`？
  - `TMVA`，`DD4hep`

passwd: 123456