粒子物理基础软件
ROOT 在数据分析中的应用

Very preliminary version

主讲人：魏逸丰

2021 年 8 月

Outline

- 什么是ROOT
- ROOT安装与运行
- 作图与拟合
- ROOT文件格式
- 使用MakeSelector进行分析

Reference

- 参考网站：https://root.cern.ch
- ROOT Beginner’s Guide: https://root.cern/primer
- Slides: 粒子物理与核物理实验中的数据分析 (清华大学 杨振伟) 网上可以找到ppt（本课程有些ppt页以及例子直接取自/修改自杨老师课件）

什么是ROOT

- 参考网站：https://root.cern.ch

ROOT is ...

A modular scientific software toolkit. It provides all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage. It is mainly written in C++ but integrated with other languages such as Python and R.

Start from examples or **try it in your browser!**

Download or Read More ...

ROOT的安装

ROOT
Data Analysis Framework

About | Install

Installing ROOT

→ On this page

- Download a pre-compiled binary distribution
- Install via a package manager

CentOS

ROOT is available on CentOS via EPEL. To install ROOT on CentOS, just run

```
$ yum install epel-release
$ yum install root
```

MacOS package managers

Homebrew

On Mac, ROOT is also available as a homebrew formula. You can install it with

```
$ brew install root
```

Download a pre-compiled binary distribution

We distribute pre-compiled ROOT for several major Linux distributions as wel
binary are simple:

1. Install all required dependencies with the system package manager
2. Download the release for the desired platform and ROOT version
3. Unpack the archive
4. Add the ROOT libraries and executables to your environment by sourc
   ROOT binary release, in the `bin` directory.

Source distribution

Platform	Files	Size
source	root_v6.24.02.source.tar.gz	177M

Binary distributions

Platform	Files
CentOS 7	root_v6.24.02.Linux-centos7-x86_64-gcc4.8.tar.gz
Fedora 32	root_v6.24.02.Linux-fedora32-x86_64-gcc10.2.tar.gz
Ubuntu 16	root_v6.24.02.Linux-ubuntu16-x86_64-gcc5.4.tar.gz
Ubuntu 18	root_v6.24.02.Linux-ubuntu18-x86_64-gcc7.5.tar.gz
Ubuntu 20	root_v6.24.02.Linux-ubuntu20-x86_64-gcc9.3.tar.gz
macOS 10.14 x86_64 Xcode 11	root_v6.24.02.macos-10.14-x86_64-clang110.pkg
macOS 10.14 x86_64 Xcode 11	root_v6.24.02.macos-10.14-x86_64-clang110.tar.gz
macOS 10.15 x86_64 Xcode 12	root_v6.24.02.macos-10.15-x86_64-clang120.pkg

ROOT的安装 (Binary)

```
[weiyf@dampe root_v6_24_02]$ ls
root←root_v6.24.02.Linux-centos7-x86_64-gcc4.8.tar.gz
```

tar -xvf xxxx.tar.gz

```
[weiyf@dampe bin]$ pwd
/home/weiyf/lustre/softwareInstall/root_v6_24_02/root/bin
[weiyf@dampe bin]$ ls
g2root          memprobe            root          rootcp              rootmkdir   rootrm          setxrd.sh
genreflex       prepareHistFactory  rootbrowse    rootdrawtree        rootmv      roots           thisroot.csh
h2root          proofserv           rootcint      rooteventselector   rootnb.exe  roots.exe       thisroot.fish
hadd            proofserv.exe       rootcling     root.exe            rootn.exe   rootslimtree    thisroot.sh
hist2workspace  rmkdepend           root-config   rootls              rootprint   setxrd.csh      xpdtest
```

ROOT配置文件

在.bashrc中调用ROOT配置文件

```
[weiyf@dampe ~]$ ls
BackupWorkPlace/ buffer/          dmpwork/          .gnome2/          .mozilla/         PythonTest/
.bash_history    .cache/          .emacs            .gnome2_private/  newDataMonitor/   .qt/
.bash_logout     checkpeak.C      .esd_auth         HERD/             NucleiWorkPlace/  .root_hist
.bash_profile    .config/         .g4_hist          .lesshst          OrbitAnalysis/    .ssh/
.bashrc          data/            .gconf/           .local/           public_html/      .subversion/
.bashrc~         .dbus/           .gconfd/          log               .pulse/           .vim/
Bgo/             Desktop/         geant4_workdir/   lustre/           .pulse-cookie     .vimback/
```

```
#source /data/weiyf/softwareInstall/rootInstall/bin/thisroot.sh
source /home/weiyf/lustre/softwareInstall/root_v6_24_02/root/bin/thisroot.sh
#export PATH=$PATH:/export/software/cmake-3.7.2/bin
```

source .bashrc或者重启终端

```
[weiyf@dampe ~]$ echo $ROOTSYS
/home/weiyf/lustre/softwareInstall/root_v6_24_02/root
```

完成！

ROOT的安装 (Source)

```
root-6.24.02
rootbuild
root_v5.34.34.Linux-slc6-x86_64-gcc4.4.tar.gz
root_v5.34.34.source.tar.gz
root_v6.04.06.source.tar.gz
root_v6.24.02.source.tar.gz
```

建立一个build目录
cd rootbuild
ccmake ../root-xx.xx.xx　　cmake 版本要求高于3.9
按c键（configuration）
设置root安装目录
按c键（configuration）
按t键进入高级设置，可以配置可选项（非必要步骤）
按g键（generation）
按e键退出配置界面
make –j9（会从github下载文件，可能连不上）
make install
在 .bashrc中设置调用thisroot.sh　　**完成！**

ROOT的安装 (Source)

```
                                                    Page 1 of 6
CHROME_EXECUTABLE                    CHROME_EXECUTABLE-NOTFOUND
CMAKE_BUILD_TYPE                     Release
CMAKE_CXX_EXTENSIONS                 OFF
CMAKE_CXX_STANDARD                   11
CMAKE_INSTALL_JSROOTDIR
CMAKE_INSTALL_OPENUI5DIR
CMAKE_INSTALL_PREFIX                 /home/weiyf/lustre/softwareInstall/root_v6_24_02_source
CMAKE_INSTALL_PYTHONDIR              lib
CMAKE_INVOKE                         /lustre/weiyf/software/cmake/cmake-3.21.1-linux-x86_64/bin/cmake
CXX_STANDARD_STRING                  11
DL_LIBRARY_PATH                      /usr/lib64/libdl.so
ENABLE_EXPERIMENTAL_NEW_PASS_M       OFF
ENABLE_LINKER_BUILD_ID               OFF
ENABLE_X86_RELAX_RELOCATIONS         OFF
FIREFOX_EXECUTABLE                   /usr/bin/firefox
GLEW_DIR                             GLEW_DIR-NOTFOUND
GLEW_FOUND                           ON
GOLD_EXECUTABLE                      /usr/bin/ld.gold
GO_EXECUTABLE                        GO_EXECUTABLE-NOTFOUND
ICONV_LIBRARY_PATH                   /usr/lib64/libc.so
LLVM_BUILD_TYPE                      Release
LZ4_FOUND                            ON
PERL_EXECUTABLE                      /usr/bin/perl
PYTHIA6_pythia6_dummy_LIBRARY        PYTHIA6_pythia6_dummy_LIBRARY-NOTFOUND
PYTHIA8_DATA                         PYTHIA8_DATA-NOTFOUND
PY_PYGMENTS_FOUND                    OFF
PY_PYGMENTS_LEXERS_C_CPP_FOUND       OFF
PY_YAML_FOUND                        OFF
RT_LIBRARY                           /usr/lib64/librt.so

CHROME_EXECUTABLE: Path to a program.
Press [enter] to edit option Press [d] to delete an entry
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

安装路径

ROOT的运行

运行： root / root -l

```
[weiyf@dampe ~]$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.24/02                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Jun 28 2021, 09:28:51                 |
  | From tags/v6-24-02@v6-24-02                                      |
  | With c++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)                 |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0] .q
[weiyf@dampe ~]$ root -l
root [0] .q
[weiyf@dampe ~]$
```

进入ROOT环境

.q 退出ROOT环境

ROOT环境其它常用指令：

.L macro.C	Load文件macro.C
.x macro.C	执行文件macro.C
.ls	显示ROOT当前环境的所有信息
.! ls	显示Linux系统当前目录的所有信息

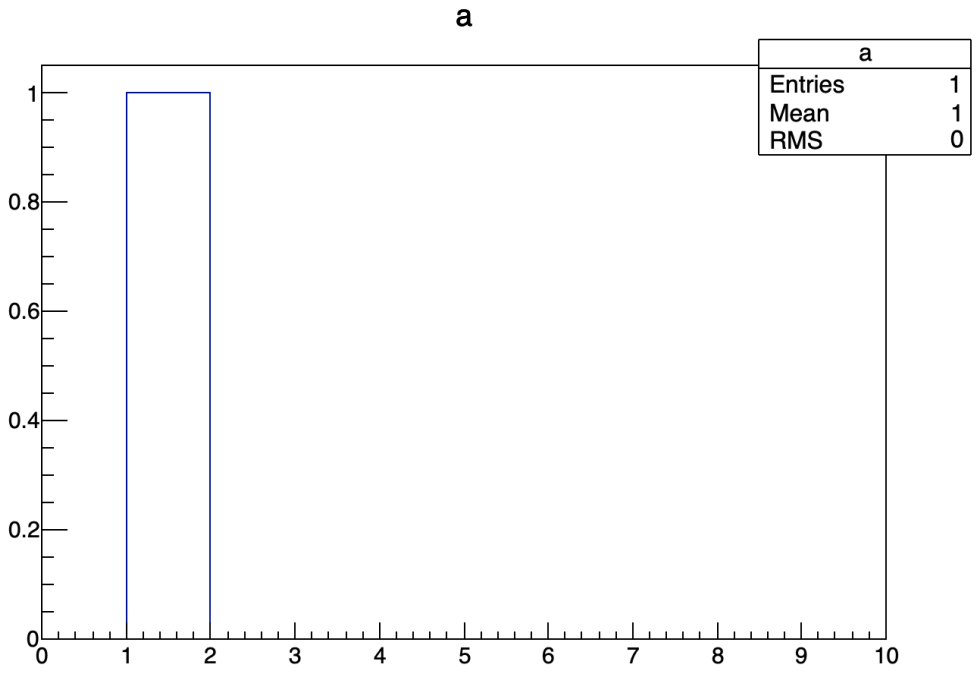
注：ROOT环境中，ROOT指令都以“.”开头

系统指令都以“.!”开头

ROOT的运行(命令行执行)

- ROOT使用C++语法
- C++语句可以在ROOT环境直接运行
- ROOT的类都以T开头：TFile，TH1D，TTree

```
[weiyf@dampe ~]$ root -l
root [0] TH1D *a = new TH1D("a","a",10,0,10);
root [1] a->Fill(1);
root [2] a->Draw();
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3]
```



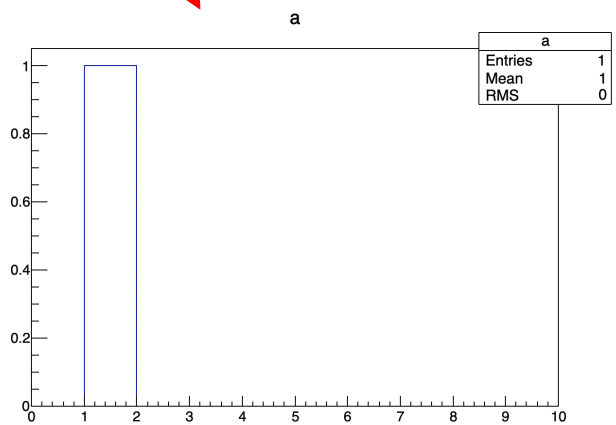
ROOT的运行(脚本执行/解释执行)

```
1 void aTest(){
2   TH1D *a = new TH1D("a","a",10,0,10);
3   a->Fill(1);
4   a->Draw();
5 }
```

脚本一般为后缀名 .C 的文件

```
[weiyf@dampe root]$ ls
aTest.C
[weiyf@dampe root]$ root -l aTest.C
root [0]
Processing aTest.C...
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [1]
```

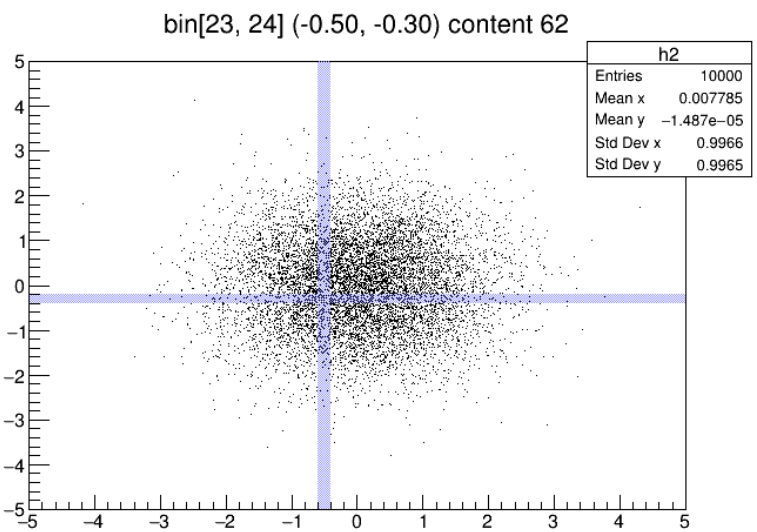
或者进入ROOT环境后使用 .x aTest.C



ROOT Tutorials

```
[weiyf@dampe root]$ ls $ROOTSYS/tutorials
CMakeLists.txt      demoshelp.C   fitsio          graphs          html
cocoa               doc           foam            gui             http
cont                eve           gallery.root    hist            image
CTestCustom.cmake   eve7          geom            histfactory     index.md
dataframe           fft           gl              hsimple.C       io
demos.C             fit           graphics        hsimple.root    launcher.py
```

Tips:
根据关键字“xxxx”从tutorials的例子中寻找线索
grep –sirn "xxxx" $ROOTSYS/tutorials
比如找随机数用法：grep –sirn "random" $ROOTSYS/tutorials



tutorials/hist/hlHisto1.C

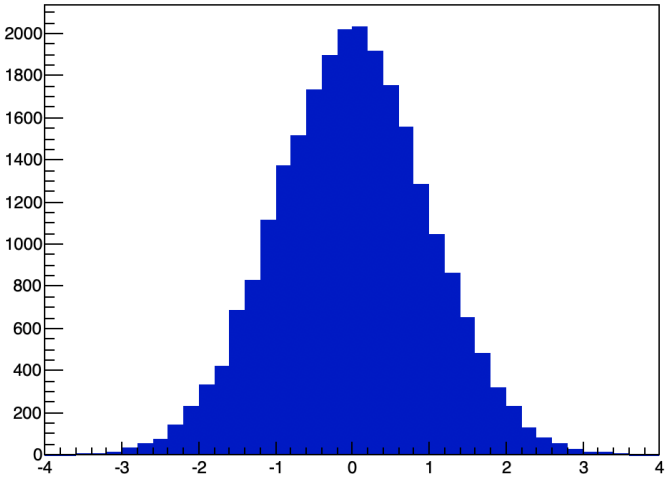
注意：
ROOT 5 与 ROOT 6 有些地方不同
例子可能不兼容

作图与拟合

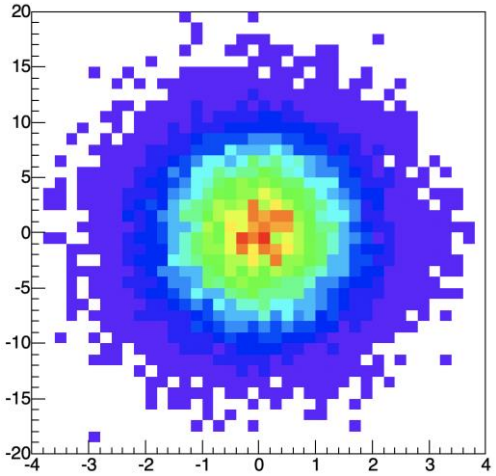
作图(常用分类)

- 直方图类：TH1F/TH1D, TH2, TH3 ..., TProfile
- 图形：TGraph, TGraphErrors ...
- 数学函数：TF1, TF2 ...
- 画布：TCanvas, TPad

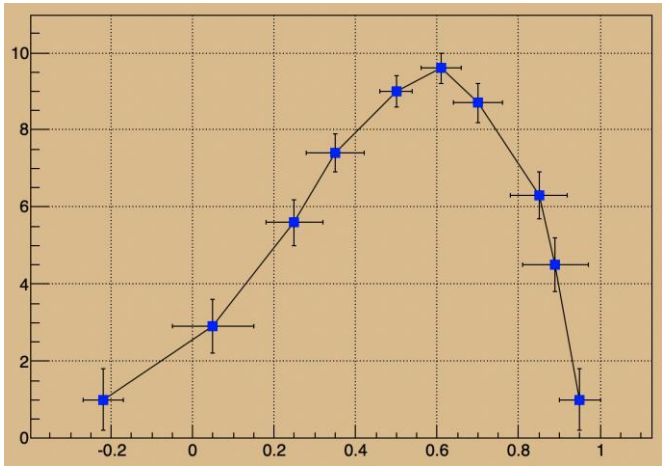
TH1



TH2



TGraphErrors



统计直方图

- 定制一维直方图

```
TH1F *hist_name = new TH1F("hist_name","hist_title",
num_bins,x_low,x_high);
```

- 定制二维图

```
TH2F *hist_name = new TH2F("hist_name","hist_title",
num_bins_x,x_low,x_high,num_bins_y,y_low,y_high);
```

- 定制三维图

```
TH3F *hist_name = new TH3F("hist_name","hist_title",
num_bins_x,x_low,x_high,num_bins_y,y_low,y_high,
num_bins_z,z_low,z_high);
```

- 填充统计图

```
hist_name.Fill(x);
hist_name.Fill(x,y);
Hist_name.Fill(x,y,z);
```

Copy from 杨振伟老师

绘图：

```
root[0]hist_name.Draw();
```

ROOT 类的用法查询

网站：https://root.cern.ch/doc/master/index.html

ROOT
Reference Guide
Version master
Search

善用搜索

- ROOT
  - ROOT Reference Documentation
  - Tutorials
  - Functional Parts
  - Namespaces
  - All Classes
  - Files
  - Release Notes

ROOT Reference Documentation

Introduction

Welcome to ROOT!

This is the reference manual of the **ROOT** software tooklit. You can find in the reference documentation page pointers to reference manuals for all ROOT versions.

Public Member Functions

TH1D ()
Constructor. More...

TH1D (const char *name, const char *title, **Int_t** nbinsx, const **Double_t** *xbins)
Create a 1-Dim histogram with variable bins of type double (see **TH1::TH1** for explanation of parameters) More...

TH1D (const char *name, const char *title, **Int_t** nbinsx, const **Float_t** *xbins)
Create a 1-Dim histogram with variable bins of type double (see **TH1::TH1** for explanation of parameters) More...

TH1D (const char *name, const char *title, **Int_t** nbinsx, **Double_t** xlow, **Double_t** xup)
Create a 1-Dim histogram with fix bins of type double (see **TH1::TH1** for explanation of parameters) More...

ROOT 类的用法查询

直接在ROOT环境中查询

```
[weiyf@dampe hist]$ root -l
root [0] TH1D *a = new TH1D(
TH1D TH1D()
TH1D TH1D(const TH1D& h1d)
TH1D TH1D(const TVectorD& v)
TH1D TH1D(const char* name, const char* title, Int_t nbinsx, Double_t xlow, Double_t xup)
TH1D TH1D(const char* name, const char* title, Int_t nbinsx, const Double_t* xbins)
TH1D TH1D(const char* name, const char* title, Int_t nbinsx, const Float_t* xbins)
```

打半个括号，按Tab

成员函数查询

```
root [0] TH1D *a = new TH1D()
(TH1D *) 0x360e5f0
root [1] a->
AbstractMethod
Add
AddAt
AddBinContent
AddDirectory
AddDirectoryStatus
Adopt
AndersonDarlingTest
AppendPad
At
```

按Tab

Example 01

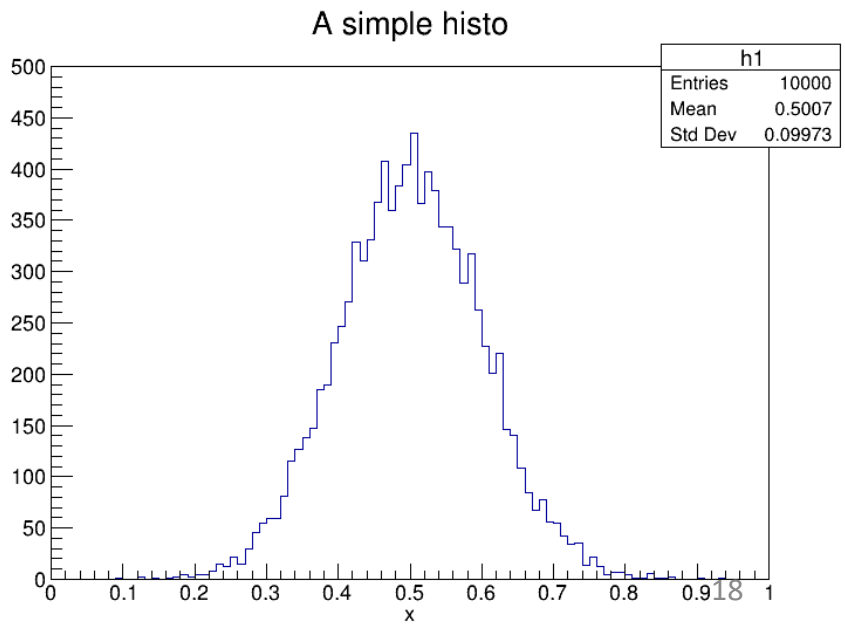
```
1  void Ex01_DrawHist(){
2    const Int_t NEntry= 10000 ;
3    TFile* file = new TFile("hist1.root","RECREATE");
4    TH1F*  h1 = new TH1F("h1","A simple histo",100,0,1);
5    double mean = 0.5;
6    double sigma = 0.1;
7    for (int i=0;i<NEntry;i++) h1->Fill( gRandom->Gaus(mean,sigma) );
8    h1->Draw();
9    h1->GetYaxis()->SetRangeUser(0,500);
10   h1->GetXaxis()->SetTitle("x");
11   h1->GetXaxis()->CenterTitle();
12   file->cd();
13   h1->Write();
14 }
```

建立输出文件

用高斯分布填充直方图

将直方图写入文件

```
[weiyf@dampe root]$ ls
aTest.C  Ex01_DrawHist.C  hist1.root
```



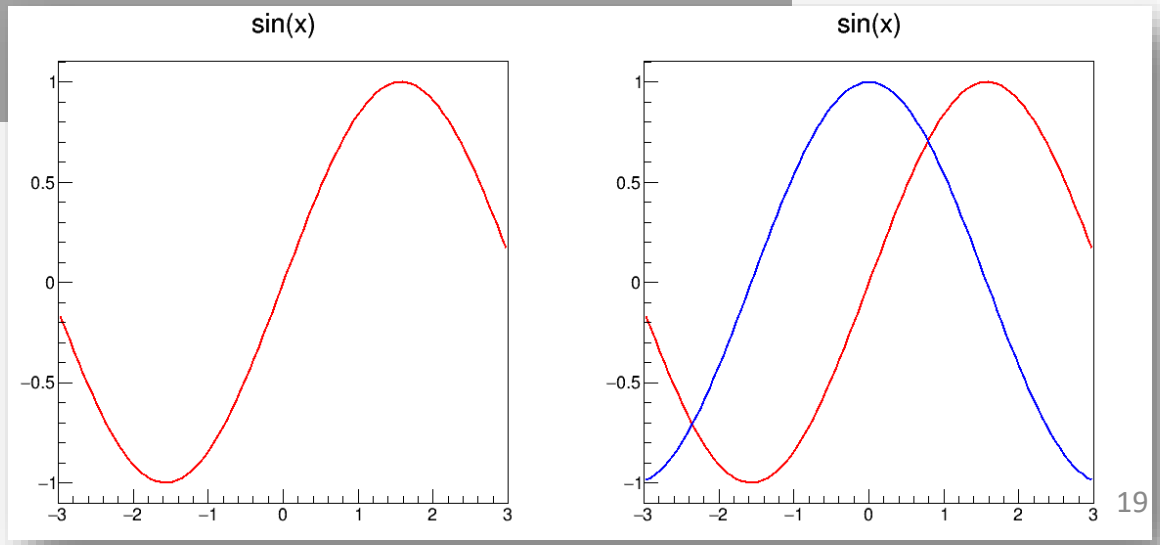
Example 02

```
1  void Ex02_DrawPad(){
2    TFile* file = new TFile("hist2.root","RECREATE");
3
4    TF1 *f1 = new TF1("func1","sin(x)",-3,3);
5    TF1 *f2 = new TF1("func2","cos(x)",-3,3);
6
7    TCanvas *aCvs = new TCanvas("aCvs","aCvs",1200,600);
8    aCvs->Divide(2,1);
9
10   aCvs->cd(1);
11   f1->Draw();
12
13   aCvs->cd(2);
14   f1->Draw();
15   f2->Draw("same");
16   f2->SetLineColor(kBlue);
17
18   aCvs->Write();
19   file->Close();
20 }
```

建立两个数学函数

建立画布，并分成两个Pad

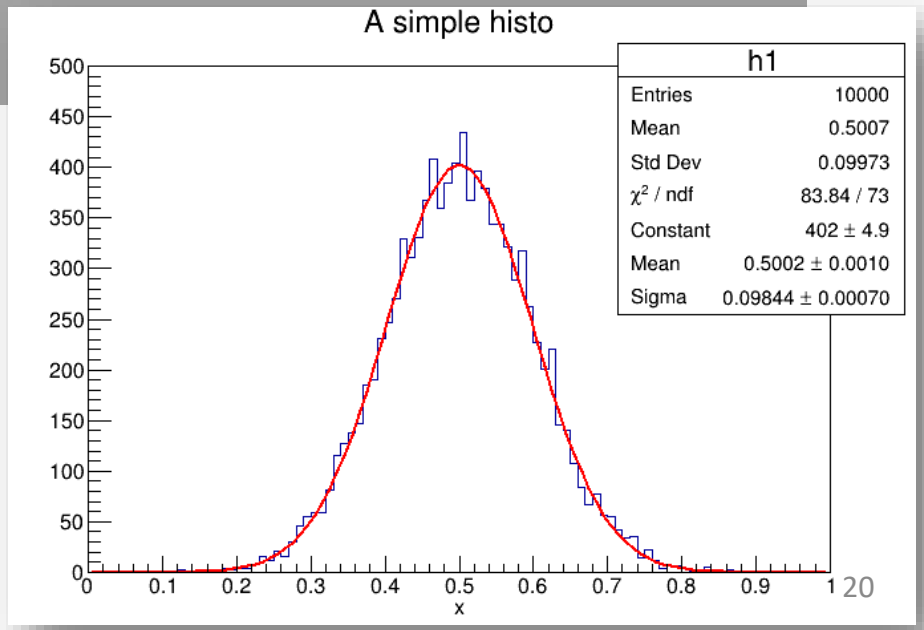
将数学函数分别写入两个Pad



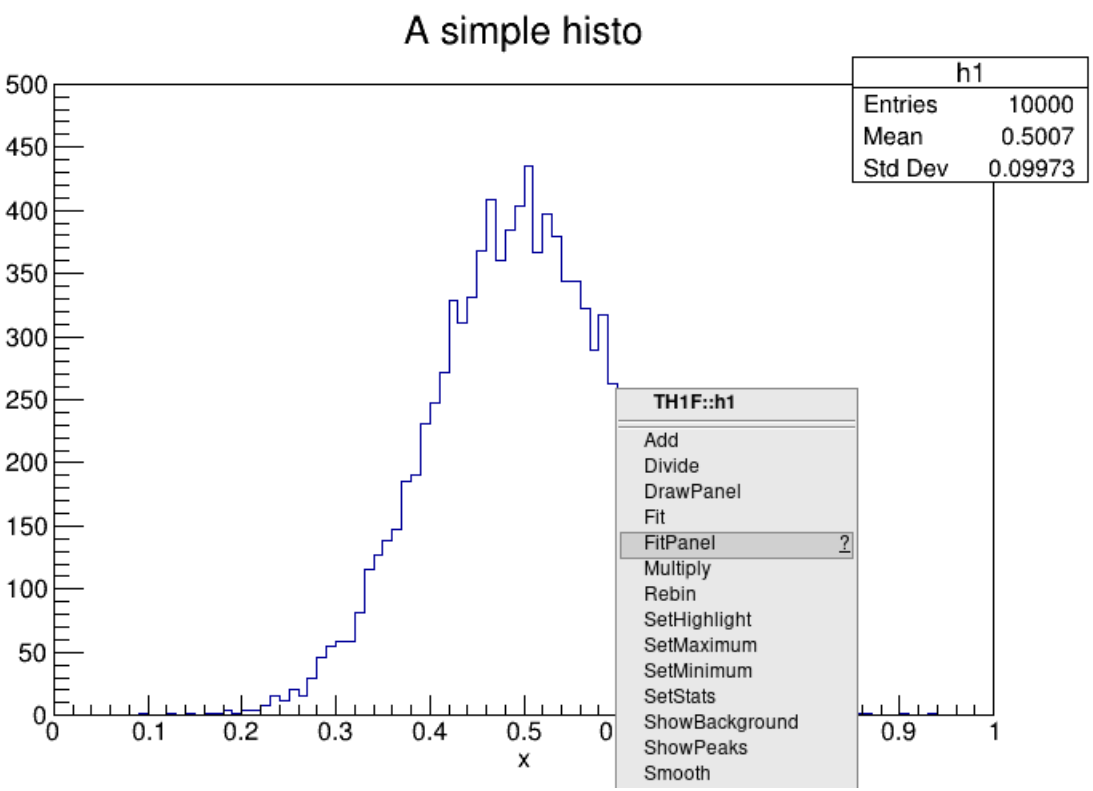
拟合 Example03

```
1  void Ex03_FitHist(){
2    const Int_t NEntry= 10000 ;
3    TFile* file = new TFile("hist1.root","RECREATE");
4    TH1F*  h1 = new TH1F("h1","A simple histo",100,0,1);
5    double mean = 0.5;
6    double sigma = 0.1;
7    for (int i=0;i<NEntry;i++) h1->Fill( gRandom->Gaus(mean,sigma) );
8    h1->Draw();
9    h1->GetYaxis()->SetRangeUser(0,500);
10   h1->GetXaxis()->SetTitle("x");
11   h1->GetXaxis()->CenterTitle();
12   h1->Fit("gaus","","",0,1);
13   file->cd();
14   h1->Write();
15 }
```

拟合

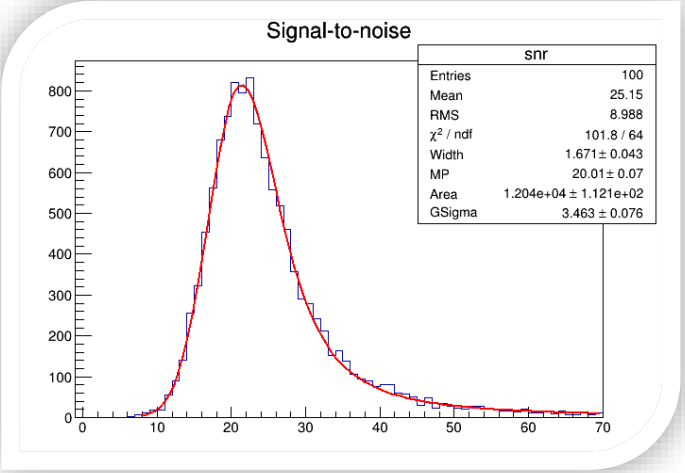


拟合 FitPanel


A simple histo
h1
Entries 10000
Mean 0.5007
Std Dev 0.09973
500
450
400
350
300
250
200
150
100
50
0
0
0.1
0.2
0.3
0.4
0.5
0
0.9
1
x
TH1F::h1
Add
Divide
DrawPanel
Fit
FitPanel
Multiply
Rebin
SetHighlight
SetMaximum
SetMinimum
SetStats
ShowBackground
ShowPeaks
Smooth


Fit Panel
Data Set: TH1F::h1
拟合函数
Fit Function
Type: Predef-1D
gaus
Operation
Nop
Add
NormAdd
Conv
gaus
Selected:
gaus
拟合参数
Set Parameters...
General
Minimization
Fit Settings
Method
Chi-square
User-Defined...
Linear fit
Robust: 0.95
Fit Options
Integral
Use range
Best errors
Improve fit results
All weights = 1
Add to list
Empty bins, weights=1
Use Gradient
Draw Options
SAME
No drawing
Do not store/draw
拟合范围
X 0.00
1.00
Update
Fit
Reset
Close
TH1F::h1
LIB Minuit
MIGRAD
Itr: 0
Prn: DEF

拟合 自定义函数



tutorials/fit/langaus.C

```
Double_t langaufun(Double_t *x, Double_t *par) {
```

自定义函数

```
   //Fit parameters:
   //par[0]=Width (scale) parameter of Landau density
   //par[1]=Most Probable (MP, location) parameter of Landau density
   //par[2]=Total area (integral -inf to inf, normalization constant)
   //par[3]=Width (sigma) of convoluted Gaussian function
   //
   //In the Landau distribution (represented by the CERNLIB approximation),
   //the maximum is located at x=-0.22278298 with the location parameter=0.
   //This shift is corrected within this function, so that the actual
   //maximum is identical to the MP parameter.
```

```
TF1 *ffit = new TF1(FunName,langaufun,fitrange[0],fitrange[1],4);
ffit->SetParameters(startvalues);
ffit->SetParNames("Width","MP","Area","GSigma");

for (i=0; i<4; i++) {
   ffit->SetParLimits(i, parlimitslo[i], parlimitshi[i]);
}

his->Fit(FunName,"RB0");   // fit within specified range, use ParLimits, do not plot
```

自定义TF1

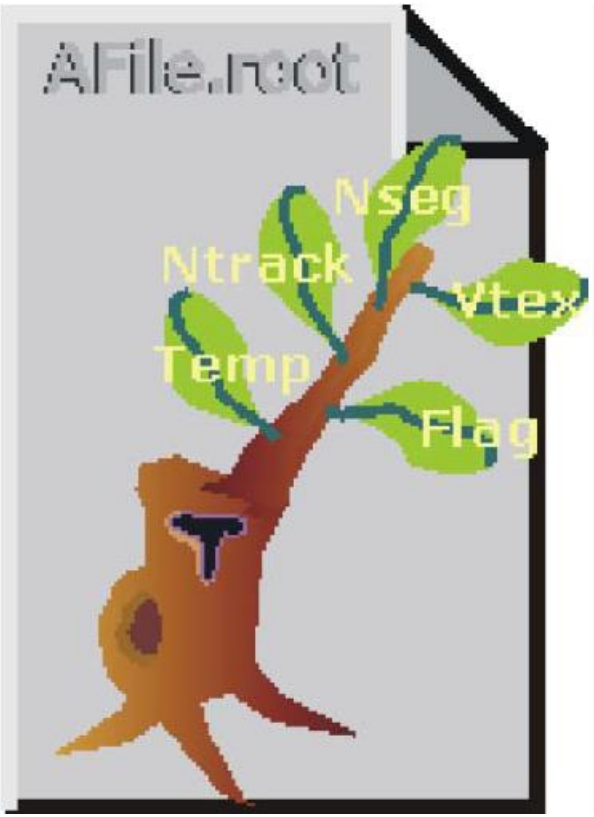
ROOT文件格式

粒子物理数据存储需求

- 海量数据
- 每个事例都需要存储多种信息
  - 能量
  - 径迹
  - 电荷
  - …
- 每个事例都要经过筛选逻辑处理

TTree

- 适用于大量的类型相同的对象
- 可以存储包括类的对象、数组等各种类型数据
- 一般情况下，tree的Branch，Leaf信息就是一个事例的完整信息，有了tree之后，可以很方便地对事例进行循环处理
- 占用空间少，读取速度快



Ttree & TBranch

TTree构造函数

```
TTree TTree(const char* name, const char* title,
Int_t splitlevel = 99)
```

创建Ttree，并设置Branch，比如：

```
Int_t RunID;
TTree *t1 = new TTree("t1","test tree");
TBranch *br = t1->Branch("RunID",&RunID,"RunID/I");
```

存入Tree　Example 04

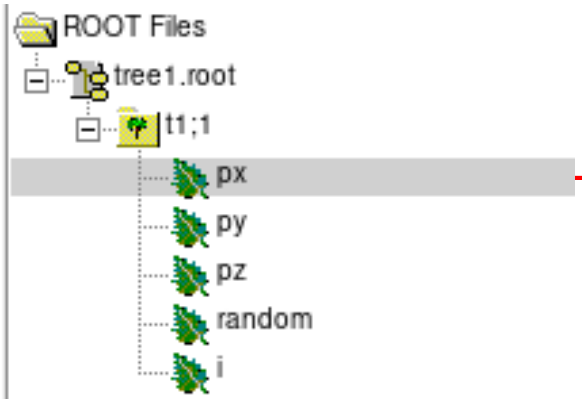
```
1  void Ex04_SaveTree() {
2
3    TFile *f = new TFile("tree1.root","recreate");
4    TTree *t1 = new TTree("t1","test tree");
5    gRandom->SetSeed(0);
6    Float_t px,py,pz;
7    Double_t random;
8    Int_t i;
9    //Set the Branches of tree
10   t1->Branch("px",&px,"px/F");
11   t1->Branch("py",&py,"py/F");
12   t1->Branch("pz",&pz,"pz/F");
13   t1->Branch("random",&random,"random/D");
14   t1->Branch("i",&i,"i/I");
15   for (i=0;i<5000;i++) {
16     gRandom->Rannor(px,py);
17     pz = px*px + py*py;
18     random = gRandom->Rndm();
19     t1->Fill();//Fill tree
20   }
21   t1->Write();
22 }
```

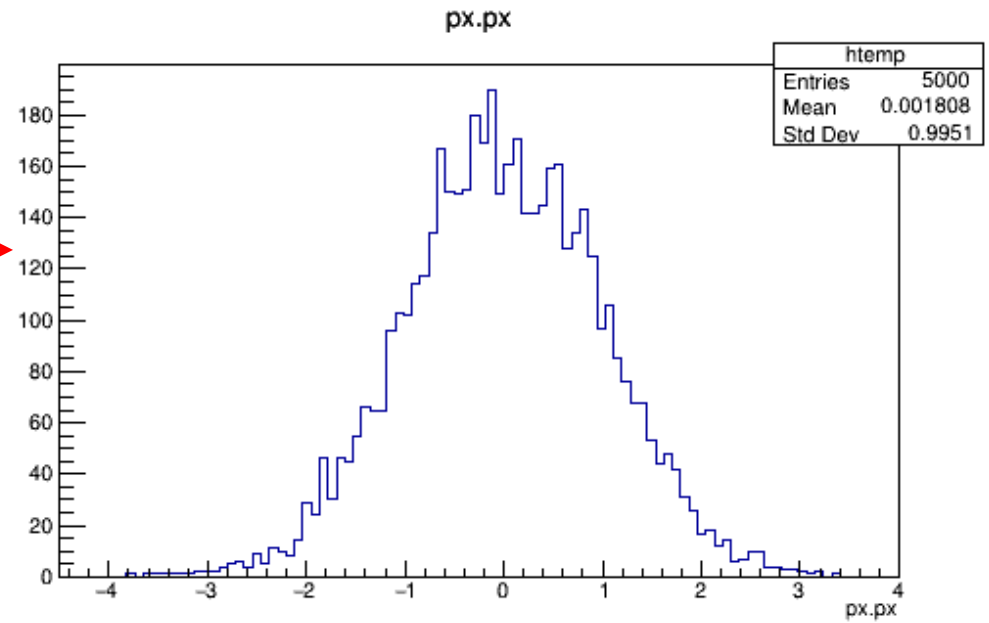
定义一个Tree

设置Branch，参数分别为
“名称”，“地址”，“leaf列表和类型”

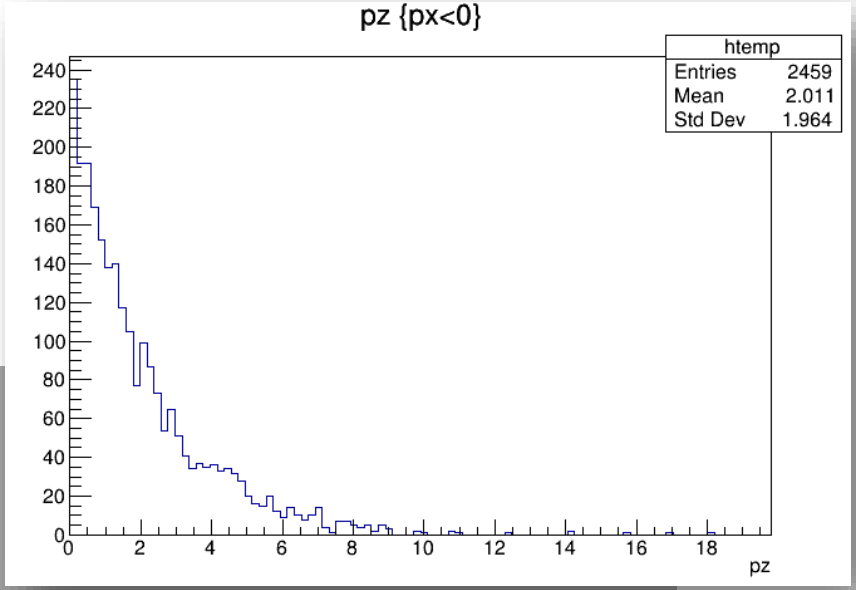
查看Tree的信息

```
[weiyf@dampe root]$ root -l tree1.root
root [0]
Attaching file tree1.root as _file0...
(TFile *) 0x35a5ec0
root [1] TBrowser a
```





查看Tree的信息




```
[weiyf@dampe root]$ root -l tree1.root
root [0]
Attaching file tree1.root as _file0...
(TFile *) 0x3a9d020
root [1] .ls
TFile**         tree1.root
 TFile*         tree1.root
  KEY: TTree    t1;1    test tree
root [2] t1->Show(0)
======> EVENT:0
 px              = 0.154407
 py              = 0.531087
 pz              = 0.305895
 random          = 0.714362
 i               = 0
root [3] t1->GetEntries()
(long long) 5000
root [4] t1->Draw("px")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [5] t1->Draw("pz","px<0")
(long long) 2459
root [6]
```

Show第0个事例的信息

做一些简单的判选

查看Tree的信息

也可以

```
>root -l                    进入root
root[0] TFile *f1=new TFile("tree1.root");
root[1] t1->Draw( "sqrt(px*px+py*py)" );
root[2] TH1F *h1;
root[3] t1->Draw("px>>h1");
root[4] t1->Draw("py","px>0","sames");
root[5] t1->Draw("py","","sames");
```

读取Tree的信息 Example 05

```
1  void Ex05_ReadTree() {
2
3    TBranch        *b_px;    //!
4
5    Float_t px,py,pz;
6    Double_t random;
7    Int_t i;
8
9    TFile *f = new TFile("tree1.root");
10   TTree *t1 = (TTree*) f->Get("t1");
11
12   t1->SetBranchAddress("px", &px, &b_px);
13   t1->SetBranchAddress("py",&py);
14   t1->SetBranchAddress("pz",&pz);
15   t1->SetBranchAddress("random",&random);
16   t1->SetBranchAddress("i",&i);
17
18   TH1D *h1d = new TH1D("test11","test11",100,-2,2);
19   int nentries = t1->GetEntries();
20   for (int ii=0;ii<nentries;ii++){
21     t1->GetEntry(ii);
22     h1d->Fill(px);
23   }
24
25   h1d->Draw();
26 }
```

初始化

读取数据

逐事例循环进行分析

给出分析结果

TChain读取多个文件 Example 06

TChain读取包含相同Tree的ROOT文件列表

```
1 void Ex06_ReadChain(){
2   TChain *fChain = new TChain("t1");
3   fChain->Add("./tree1.root");
4   fChain->Add("./tree2.root");
5   //fChain->Add("./tree*.root");
6
7   fChain->Draw("px");
8 }
```

注意TChain的名字与ROOT文件中Tree的名字相同

使用MakeSelector进行分析

MakeSelector/MakeClass

MakeSelector/MakeClass可以自动产生分析文件和头文件

```
[weiyf@dampe MakeSelector]$ root -l tree1.root
root [0]
Attaching file tree1.root as _file0...
(TFile *) 0x22e1d60
root [1] .ls
TFile**         tree1.root
 TFile*         tree1.root
  KEY: TTree    t1;1    test tree
root [2] t1->MakeSelector("MySelector")
(int) 0
```

```
[weiyf@dampe MakeSelector]$ ls
MySelector.C  MySelector.h  tree1.root
```

MakeClass的产生方法与MakeSelector类似

.h 文件

```
 1 //////////////////////////////////////////////////////////
 2 // This class has been automatically generated on
 3 // Tue Aug 10 20:37:51 2021 by ROOT version 6.24/02
 4 // from TTree t1/test tree
 5 // found on file: tree1.root
 6 //////////////////////////////////////////////////////////
 7
 8 #ifndef MySelector_h
 9 #define MySelector_h
10
11 #include <TROOT.h>
12 #include <TChain.h>
13 #include <TFile.h>
14 #include <TSelector.h>
15 #include <TTreeReader.h>
16 #include <TTreeReaderValue.h>
17 #include <TTreeReaderArray.h>
18
19 // Headers needed by this particular selector
20
21
22 +-- 34 lines: class MySelector : public TSelector {-------
56
57 #endif
58
59 #ifdef MySelector_cxx
60 void MySelector::Init(TTree *tree)
61 +-- 10 lines: {-----------------------------------------
71
72 Bool_t MySelector::Notify()
73 +--  9 lines: {-----------------------------------------
82
83
84 #endif // #ifdef MySelector_cxx
```

Selector

读入Tree

声明接口

TSelector::Init(Ttree *tree)

```
class MySelector : public TSelector {
public :
   TTreeReader     fReader;  //!the tree reader
   TTree          *fChain = 0;   //!pointer to the analyze
d TTree or TChain

   // Readers to access the data (delete the ones you do n
ot need).
   TTreeReaderValue<Float_t> px = {fReader, "px"};
   TTreeReaderValue<Float_t> py = {fReader, "py"};
   TTreeReaderValue<Float_t> pz = {fReader, "pz"};
   TTreeReaderValue<Double_t> random = {fReader, "random"}
;
   TTreeReaderValue<Int_t> i = {fReader, "i"};
```

ROOT 6

```
#ifdef MySelector_cxx
void MySelector::Init(TTree *tree)
{
   // The Init() function is called wh
s to initialize
   // a new tree or chain. Typically I
itialized.
   // It is normally not necessary to
generated
   // code, but the routine can be ext
 needed.
   // Init() will be called many times
OF
   // (once per file to be processed)

   fReader.SetTree(tree);
}
```

ROOT 5

```
   // Set branch addresses and branch pointers
   if (!tree) return;
   fChain = tree;
   fChain->SetMakeClass(1);

   fChain->SetBranchAddress("px", &px, &b_px);
   fChain->SetBranchAddress("py", &py, &b_py);
   fChain->SetBranchAddress("pz", &pz, &b_pz);
   fChain->SetBranchAddress("random", &random, &b_random);
   fChain->SetBranchAddress("i", &i, &b_i);
```

.C文件

```
28 #include "MySelector.h"
29 #include <TH2.h>
30 #include <TStyle.h>
31
32 void MySelector::Begin(TTree * /*tree*/)
33 +--  7 lines: {--------------------------------
40
41 void MySelector::SlaveBegin(TTree * /*tree*/)
42 +--  8 lines: {--------------------------------
50
51 Bool_t MySelector::Process(Long64_t entry)
52 +-- 21 lines: {--------------------------------
73
74 void MySelector::SlaveTerminate()
75 +--  6 lines: {--------------------------------
81
82 void MySelector::Terminate()
83 +--  6 lines: {--------------------------------
```

分析初始化（创建Hist）

Event loop & process

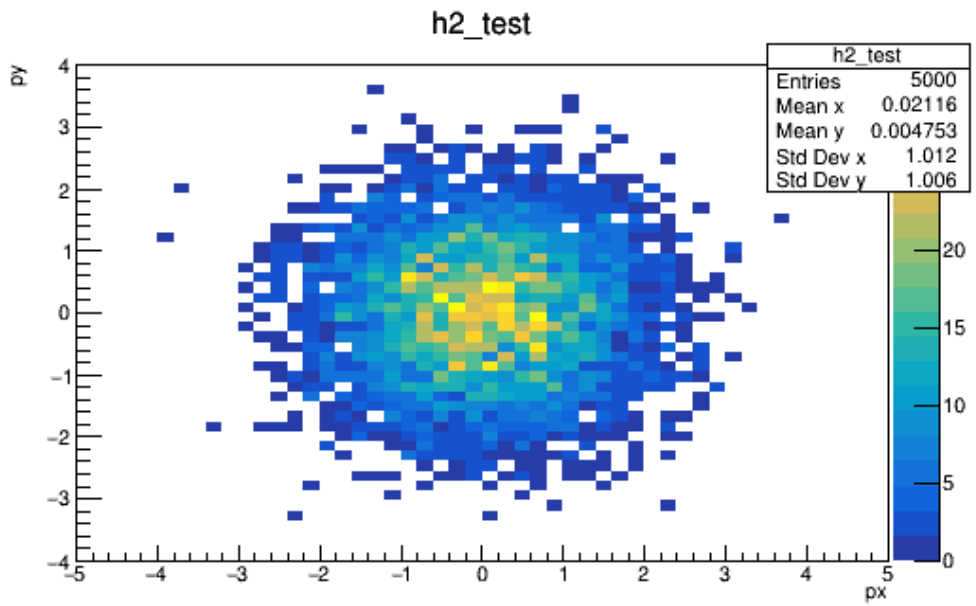
分析结束 （画图）

编写你的分析逻辑

```
32 void MySelector::Begin(TTree * /*tree*/)
33 {
34
35    TString option = GetOption();
36
37    h2_test = new TH2D("h2_test","h2_test;px;py",50,-5,5,50,-4,4);
38 }
39
40 void MySelector::SlaveBegin(TTree * /*tree*/)
41 +--  8 lines: {-------------------------------------------------
49
50 Bool_t MySelector::Process(Long64_t entry)
51 {
52
53    fReader.SetLocalEntry(entry);
54
55    h2_test->Fill(*px,*py);
56
57    return kTRUE;
58 }
59
60 void MySelector::SlaveTerminate()
61 +--  6 lines: {-------------------------------------------------
67
68 void MySelector::Terminate()
69 {
70
71    TFile *afile = new TFile("fResult.root","RECREATE");
72    h2_test->Write();
73    afile->Close();
74 }
```

Process & Result

```
[weiyf@dampe MakeSelector]$ root -l tree1.root
root [0]
Attaching file tree1.root as _file0...
(TFile *) 0x2e97550
root [1] t1->Process("MySelector.C+")
```



- 在这个过程中，ROOT自动进行了Event loop
- 用户可以不用写主程序
- 专注于自己的分析算法

Summary

- 什么是ROOT
  - 大数据处理，统计分析，可视化，数据存储
- ROOT安装与运行
  - Package manager安装
  - Binary安装
  - 编译安装
- 作图与拟合
  - 直方图，数学函数
  - 在一块画布上的多个pad作图
  - 拟合（内置函数，自定义函数）

Summary

- ROOT文件格式
  - TTree
  - Branch的存储和读取
  - TChain读取多个文件
- 使用MakeSelector进行分析
  - MakeSelector自动产生头文件和源文件
  - 自动读取复杂数据格式
  - 自动进行事例循环
  - 用户专注于自己的分析算法