RooFit

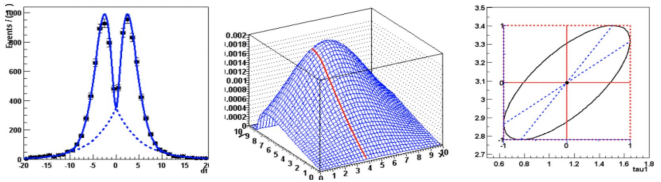
Aula 04

Sumário

Baseado no material do Wouter Verkerke (autor do RooFit).

- Introdução ao RooFit
  - Funcionalidades básicas
    - Construindo modelos (“workspace”)
    - Gerar amostras
    - Modelos Compostos e Ajustes aos dados
    - Validação

…



- Exercícios com o RooFit:
  - Construir, ajustar modelos aos dados e validar

Documentação do RooFit

Ponto inicial: https://root.cern/manual/roofit/

– Manual: https://root.cern/download/doc/RooFit_Users_Manual_2.91-33.pdf

– Quick Start Guide (20 pages, recent):

– Tutorial-macros (also in $ROOTSYS/tutorials/roofit):

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

– Mais de 200 slides do **Wouter Verkerke** documentando todos os recursos do RooFit, eles estão disponíveis na “French School of Statistics 2008”:

http://indico.in2p3.fr/getFile.py/access?contribId=15&resId=0&materialId=slides&confId=750

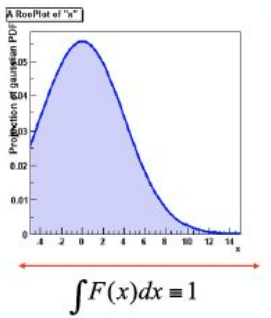
-

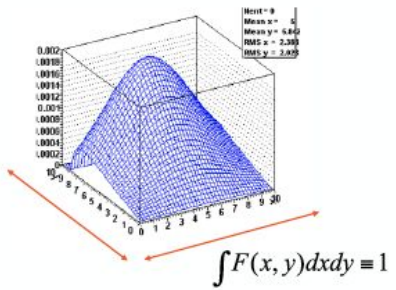
O que é o RooFit ?

É um pacote de ferramentas distribuído com o ROOT para modelagem de dados.

- Uma coleção de classes que ampliam o ambiente ROOT para modelar dados
  - É usado para modelar distribuições que são usadas para ajustes e análise estatística de dados.
    - distribuição do modelo da observável **x** em termos do parâmetro **p**
  - Função Densidade de Probabilidade (P.D.F.): P(x;p)
  - A PDF é normalizada sobre o intervalo permitido das observáveis **x** em relação ao parâmetro **p** (sobre o qual a PDF depende).

$$\int_{\Omega} P(\vec{x};\vec{p})d\vec{x} = 1$$





Por que o RooFit foi desenvolvido?

O Experimento **BaBar** no **SLAC** queria extrair o sin(2β) da dependencia temporal da violação CP de decaimentos de B:

$$e^+e^- \rightarrow \Upsilon(4S) \rightarrow B\bar{B}$$

- Reconstruir ambos os Bs, medir a diferença de tempo de decaimento;
- A física de interesse está na oscilação dependente do tempo do decaimento;

$$f_{sig} \cdot \left[\mathrm{SigSel}(m;\bar{p}_{sig}) \cdot \left(\mathrm{SigDecay}(t;\vec{q}_{sig},\sin(2\beta)) \otimes \mathrm{SigResol}(t \mid dt;\vec{r}_{sig})\right)\right] + (1-f_{sig})\left[\mathrm{BkgSel}(m;\bar{p}_{bkg}) \cdot \left(\mathrm{BkgDecay}(t;\vec{q}_{bkg}) \otimes \mathrm{BkgResol}(t \mid dt;\vec{r}_{bkg})\right)\right]$$

SLAC (Stanford Linear Accelerator Center)

https://www.slac.stanford.edu/pubs/slacpubs/9000/slac-pub-9040.pdf

Muitas questões surgiram:

- Estrutura padrão da função ROOT era claramente insuficiente para lidar com tais funções complicadas;
- Normalização de pdf nem sempre é trivial calcular;
- Ajuste unbinned, >2 dimensões, muitos eventos -> desempenho computacional é importante;

Estimativa de Parâmetros

As funções usadas em probabilidades devem ser funções de densidade de probabilidade:

$$\int F(\vec{x};\vec{p})d\vec{x} \equiv 1 \ , \ F(\vec{x};\vec{p}) > 0$$

- O modelo dos dados observados é expresso usando a Função Densidade de Probabilidade(PDF)
  - –A PDF é uma probabilidade diferencial $p(x|\theta)$

ex.: a probabilidade de observar um evento em um bin de um histograma $P_{bin} = \int_{bin} p(x|\theta)dx$

  - –A PDF é normalizada a 1 quando integrada em todo o espaço da amostra Ω $\int_{\Omega} p(x|\theta)dx = 1$

- Para estimar os parâmetros usamos **M. Likelihood Function (Função de Verossimilhança)**

$$\mathcal{L}(x_1, x_2, ...x_N; \theta) = \prod_{i=1}^{N} p(x_i|\theta)$$

- Aplica-se o log(ln) da likelihood-function. Por conveniência usa-se o negativo log-likelihood function encontramos o minimum global

$$-\ln \mathcal{L}(x_1, x_2, ...x_N; \theta) = -\sum_{i=1}^{N} \ln p(x_i|\theta) \qquad \frac{\partial \ln \mathcal{L}}{\partial \theta} = 0$$

ML estendida:

$$\mathcal{L}_{ext}(\theta, N_{\text{exp}}) = \frac{e^{-N_{\text{exp}}} \cdot N_{\text{exp}}^{N_{\text{obs}}}}{N_{\text{obs}}!} \cdot \prod_{i=1}^{N_{obs}} P(x_i|\theta)$$

Por que o RooFit ?

–ROOT pode lidar com funções complicadas, porém pode exigir a escrita de uma grande quantidade de código

  –Normalização de PDF nem sempre trivial

    •RooFit faz isso automaticamente

–No caso de ajuste complexo, o desempenho de computação é importante

  • necessidade de otimizar o código para um desempenho aceitável

    • otimização integrada disponível no RooFit

–Fornece uma descrição completa do modelo para uso posterior

RooFit

- RooFit fornece funcionalidades para construir as PDFs
  - –construção de modelo complexo
  - –composição com produto, adição, …
- Todos os modelos fornecem a funcionalidade para:
  - –ajuste ***maximum(minimum) likelihood***
  - –gerador de amostras a partir de uma PDF
  - –visualização

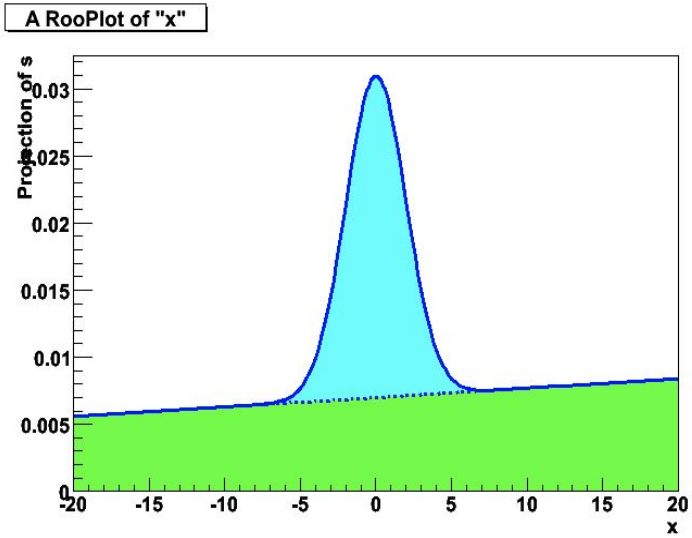
Funções vs Funções Densidades de Probabilidades

• Por que usar PDFs em vez de funções “simples” para modelar os dados?

–Fácil de interpretar os modelos.

-Se tivermos a garantia que as pdfs em Azul e em Verde sejam normalizadas a 1

-Então as frações de Azul e Verde podem ser interpretadas como #eventos



–Muitas técnicas estatísticas só funcionam corretamente com p.d.f. (ex.: maximum likelihood fits).

• O que é difícil com p.d.f ?

–A normalização pode ser difícil de calcular

(ex.: isso pode ser diferente para cada conjunto de valores de parâmetro p)

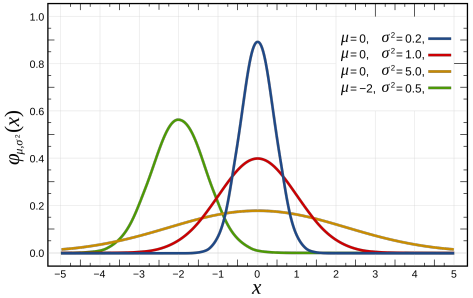
• Para dimensões >1 a integração é difícil

–O RooFit visa simplificar essas tarefas

Algumas funções de ajuste

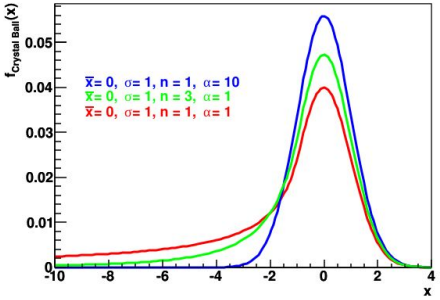
Funções de densidade de probabilidade.

Gaussiana



$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right)$$

Crystal Ball

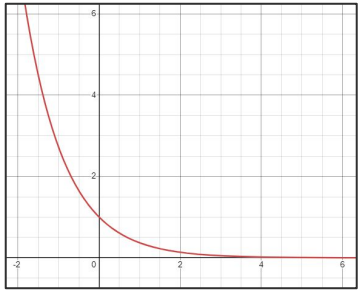


$$f(x) = N \cdot \begin{cases} \exp\left(\frac{-(x-\bar{x})^2}{2\sigma^2}\right), & \text{para } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot \left(B - \frac{x-\bar{x}}{\sigma}\right)^{-n}, & \text{para } \frac{x-\bar{x}}{\sigma} \leq -\alpha \end{cases}$$

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right)$$

$$B = \frac{n}{|\alpha|} - |\alpha|$$

Exponencial



$$f(x) = Ae^{-\lambda x}$$

O RooFit está instalado?

- Verifique se o roofit está instalado na sua versão do root: `root-config --has-roofit`

```
eliza@eae53bb4620f:~/Aula_IntroducaoAnalaiseDados_2024_02/RooFit$ root-config --has-roofit
yes
```

O **RooFit** está instalado e habilitado corretamente no seu ambiente do ROOT.

Princípios centrais de Design do RooFit

Conceitos matemáticos são representados como objetos em C++

Conceitos matematicos	**Classe no RooFit**
Varável x	**Roo**RealVar
Função $f(x)$	**Roo**AbsReal
PDF $p(x)$	**Roo**AbsPdf
Ponto espacial $\vec{x}$	**Roo**ArgSet
Integral $\int_{x_{min}}^{x_{max}} f(x)\,dx$	**Roo**RealIntegral
Listas de pontos espaciais	**Roo**AbsData

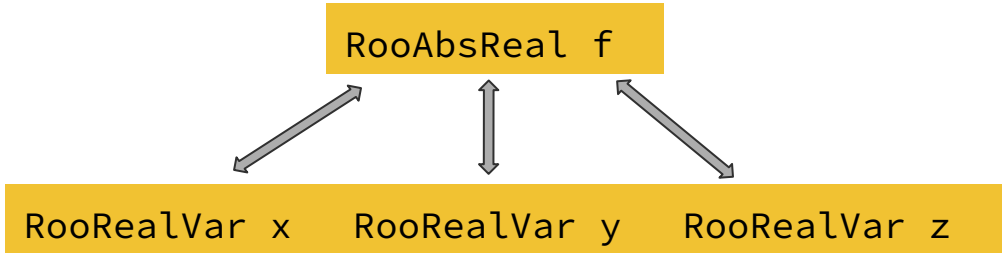
Princípios centrais de Design do RooFit

Relações entre variáveis e funções :

matemática:

f(x,y,z)

Diagrama do RooFit:



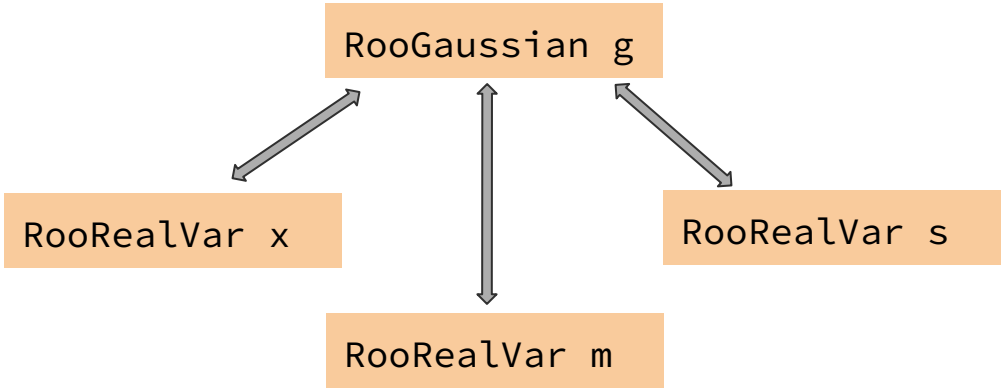
código RooFit:

```
RooRealVar x(“x”, “x”, 2);
RooRealVar y(“y”, “y”, 3);
RooRealVar z(“z”, “z”, 4);
RooBogusFunction f(“f”, “f”, x,y,z);
```

Modelagem com o RooFit

Exemplo: pdf Gaussianna

Gauss(x,m,s)



RooFit code:

```
RooRealVar x("x", "x", 2,-10,10);
RooRealVar s("s", "s", 3);
RooRealVar m("m", "m", 0);
RooGaussian g("g", "g", x,m,s);
```

Como definir as variáveis no RooFit?

As variáveis são definidas como:

RooRealVar var(“nome”, “título”, valor); ou
RooRealVar var(“nome”, “título”, valor, minValor, maxValor, “unidade”);

construir com um valor fixo / ou um intervalo / ou valor inicial + intervalo

os observáveis (ex.: x, y, energia, tempo, massa) e os parâmetros da PDF (ex.: mean, sigma, slope) ambos são variáveis.

→ o conjunto de dados “diz” a PDF quais são os observáveis
→ todas as outras varáveis são parâmetros

ao ajustar o modelo PDF aos dados: todos os parâmetros livres não fixos são ajustados, pode-se posteriormente fixar e excluir um parâmetro de ser ajustado, pelo método:

```
RooRealVar var("varName", "description", initialValue);
var.setVal(value);
var.setConstant(kTRUE);
```

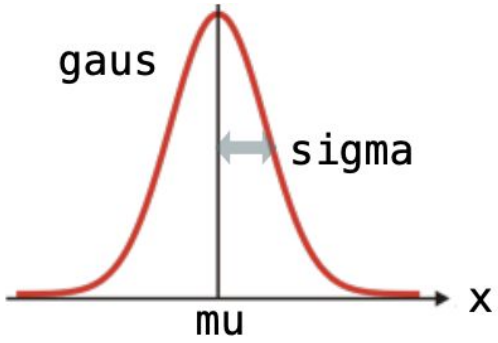
Como definir as variáveis no RooFit?

construir variável flexivel:

C++

```
RooRealVar mean("mean", "mean", 5.0);
RooRealVar shift("shift", "shift", 2.0);

RooFormulaVar mean_shifted("mean_shifted", "@0+@1", RooArgList(mean, shift));
```

Um exemplo simples (PDF Gaussiana)



Nome do objeto (no seu código)

Nome do objeto

Título do objeto

range inicial(min, máx)

objetos representando um valor "real"

```
RooRealVar x("x", "Observable",-10,10);
RooRealVar mean("mean", "B0 mass", 5.2794, "GeV");
RooRealVar sigma("sigma", "B0 mass width", 0.00027);

RooGaussian model("model", "signal pdf", x,mean,sigma);
```

unidade opcional

initial value

Objeto PDF

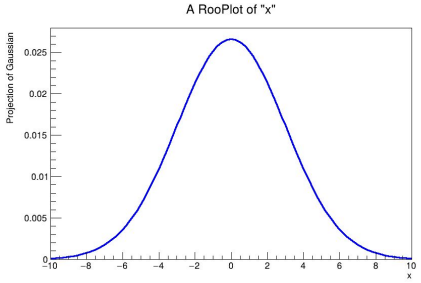
Referências as variáveis(objetos)

Um exemplo simples (PDF Gaussiana)

Macro no ROOT (C++): plotGaussian.C

definições de classes, funções e bibliotecas que o seu código precisa usar.

```
// Inclua os cabeçalhos necessários
#include "TCanvas.h"
#include "RooRealVar.h"
#include "RooGaussian.h"
#include "RooPlot.h"
#include "RooFit.h"

void plotGaussian() {
    // Definir as variáveis x, mean e width com seus intervalos
    RooRealVar x("x", "x", 0, -10, 10);

    RooRealVar mean("mean", "Mean of Gaussian", 0, -10, 10);

    RooRealVar width("width", "Width of Gaussian", 3, 0.1, 10);

    // Criar a função gaussiana
    RooGaussian g("g", "Gaussian", x, mean, width);

    // Criar um objeto RooPlot que serve como um "frame" para o gráfico
    RooPlot* frame = x.frame();

    // Plotar a PDF no frame (a função gaussiana g)
    g.plotOn(frame);

    TCanvas* c1 = new TCanvas("c1", "Gaussian Plot", 800, 600);


    frame->Draw();

    c1->Draw();

    c1->SaveAs("gauss.pdf");
}
```



RooPlot é a classe responsável por criar um "quadro" (frame) onde você pode desenhar a PDF, o histograma, e ajustar gráficos.

Gerando eventos a partir de uma pdf

Gerar 10000 eventos em x a partir de uma p.d.f Gaussiana e mostrar a distribuição

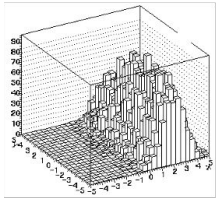
```
// Generate an unbinned toy MC set
RooDataSet* toyData = gauss.generate(x,10000);

// Generate an binned toy MC set
RooDataHist* toyData = gauss.generateBinned(x,10000);
```

Unbinned

x	y	z
1	3	5
2	4	6
1	3	5
2	4	6

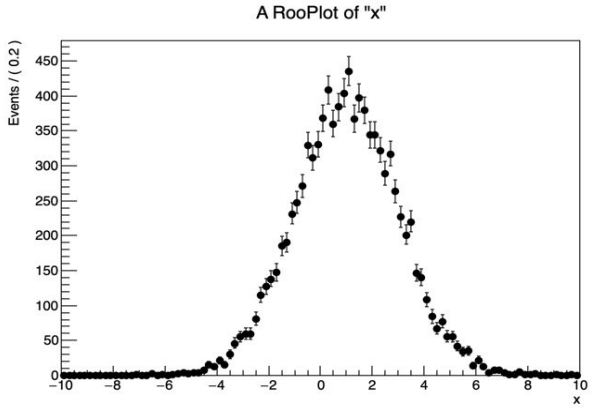
RooDataSet

Binned



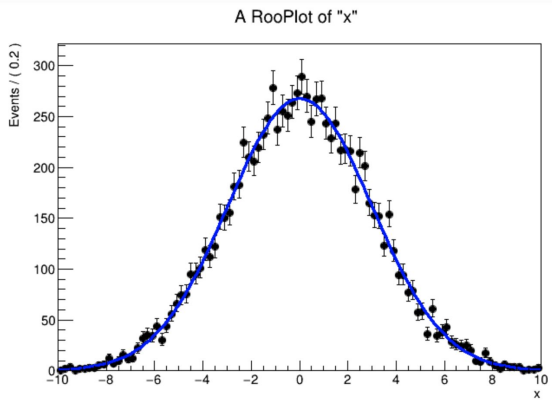
RooDataHist

RooAbsData



Gerando eventos a partir de uma pdf

```
// generate unbnined dataset of 10k events
RooDataSet* toyData = g.generate(x,10000) ;

// Perform unbinned ML fit to toy data
g.fitTo(*toyData) ;

// Plot toy data and pdf in observable x
RooPlot* frame = x.frame() ;
toyData->plotOn(frame) ;
g.plotOn(frame) ;
frame->Draw() ;
```



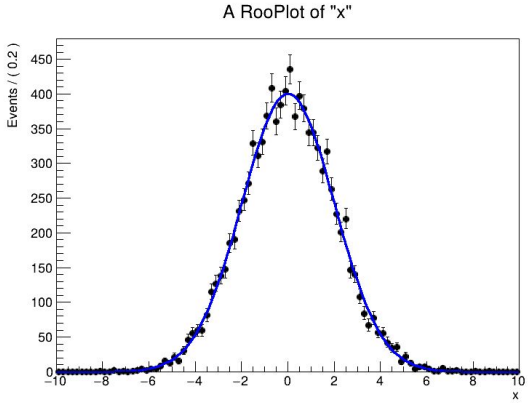
As PDF são automaticamente normalizadas ao dataset

Quando é feito o **pdf.fitTo(*data)**, construímos um objeto que representa **-log(L)** da likelihood que é minimizada usando um algoritmo (ex.: MINUIT).

Gerando eventos a partir de uma pdf

Exemplo de uma macro

```
// Inclua os cabeçalhos necessários
#include "TCanvas.h"
#include "RooRealVar.h"
#include "RooGaussian.h"
#include "RooDataSet.h"
#include "RooPlot.h"
#include "RooArgSet.h"

void generateGaussianEvents() {

    RooRealVar x("x", "x", -10, 10);

    RooRealVar mean("mean", "Mean of Gaussian", 0, -10, 10);

    RooRealVar width("width", "Width of Gaussian", 2, 0.1, 10);

    RooGaussian gaussian("gaussian", "Gaussian PDF", x, mean, width);

    // Gerar 10.000 eventos a partir da PDF
    RooDataSet* data = gaussian.generate(RooArgSet(x), 10000);

    gaussian.fitTo(*data);
    RooPlot* frame = x.frame();
    data->plotOn(frame);

    // Plotar a função PDF Gaussiana ajustada no mesmo frame
    gaussian.plotOn(frame);

    TCanvas* c1 = new TCanvas("c1", "Gaussian Event Generation", 800, 600);

    frame->Draw();

    c1->Draw();
    c1->SaveAs("gaussianEventos.png");
}
```

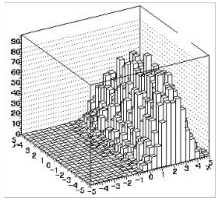


Importando dados

Unbinned

x	y	z
1	3	5
2	4	6
1	3	5
2	4	6

Binned



RooDataSet RooDataHist

RooAbsData

- Importar dados unbinned de ROOT **TTrees**

```
// Import unbinned data
RooDataSet data("data","data",x,Import(*myTree));
```

– Importa o **TTree** branch chamado "**x**".

– Tipos possíveis: Double_t, Float_t ou Int_t.

-Todo dado é convertido internamente em Double_t.

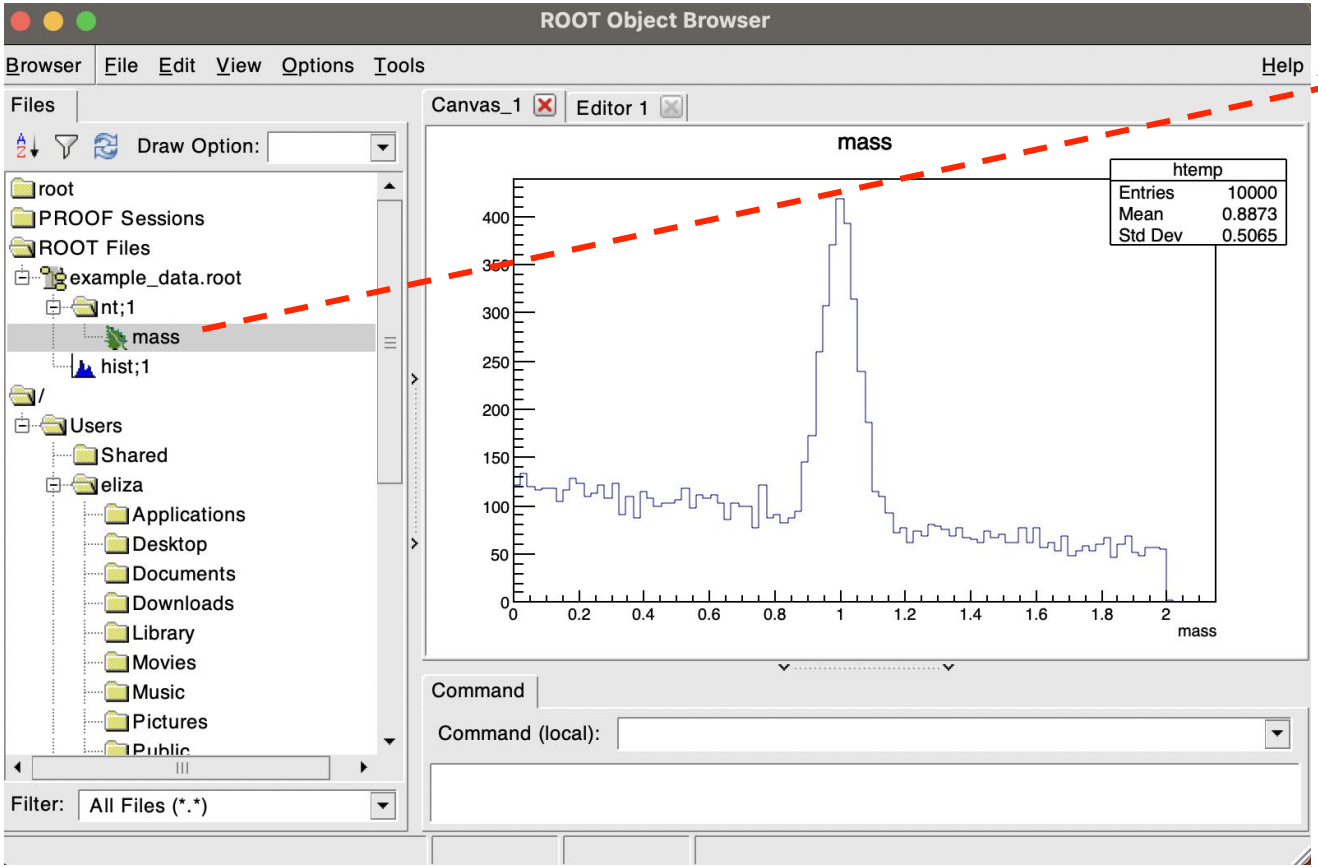
– Especifique um RooArgSet de vários observáveis para importar múltiplos observáveis (conjunto sem ordem específica).

- Importar dados de histogramas ROOT THX.

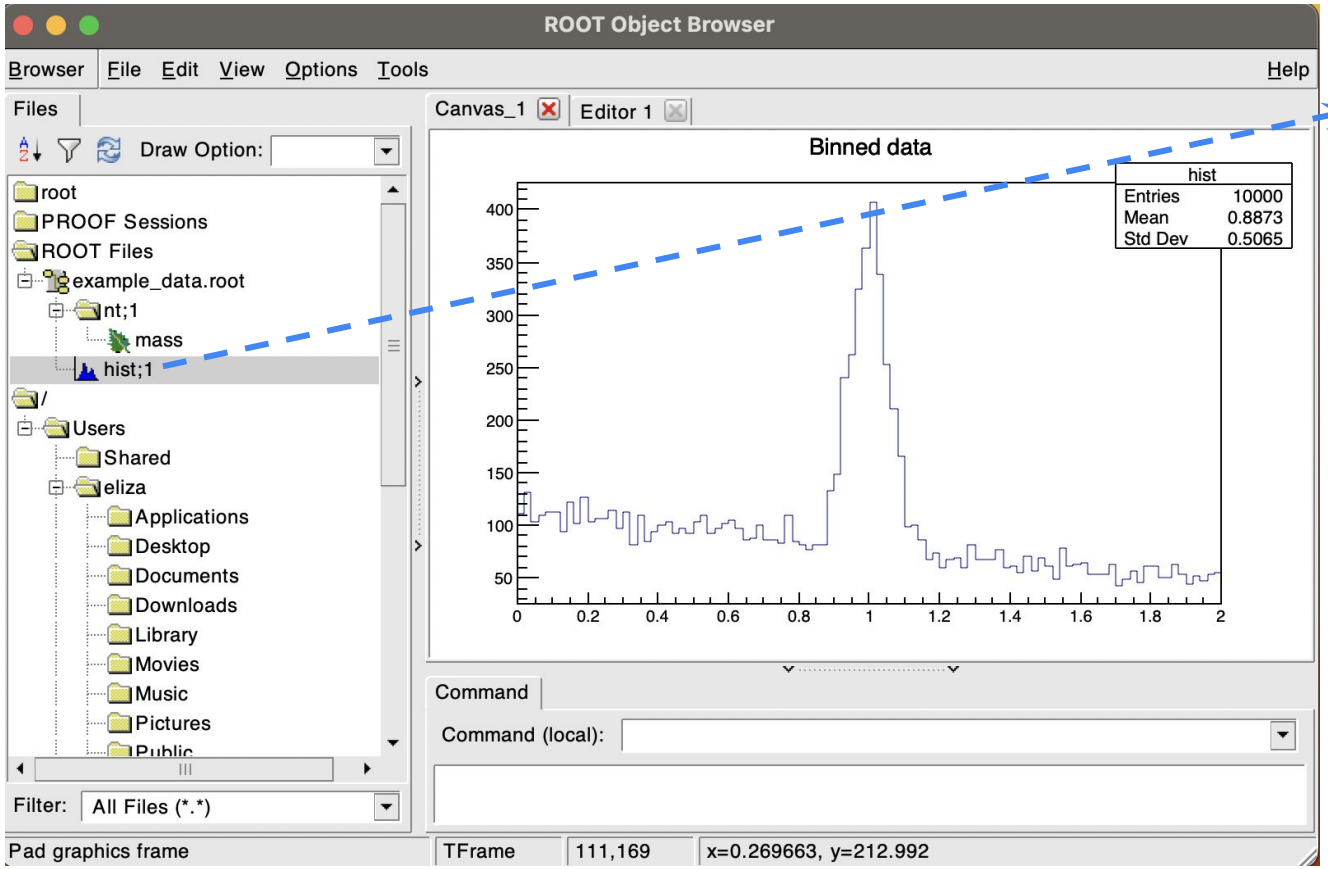
```
// Import binned data
RooDataHist data("data","data",x,Import(*myTH1));
```

– Importa valores, define binning e erros (se definido).

– Especifica uma lista de observáveis RooArgList quando importa um TH2/3.



Unbinned data


ROOT Object Browser
Browser
File
Edit
View
Options
Tools
Help
Files
Draw Option:
root
PROOF Sessions
ROOT Files
example_data.root
nt;1
mass
hist;1
/
Users
Shared
eliza
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
Filter:
All Files (*.*)
Canvas_1
Editor 1
Binned data
hist
Entries 10000
Mean 0.8873
Std Dev 0.5065
400
350
300
250
200
150
100
50
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Command
Command (local):
Pad graphics frame
TFrame
111,169
x=0.269663, y=212.992

binned data

Importando dados

```
RooRealVar x("x", "x", -10.0, 10.0);
RooRealVar c("c", "c", 0.0, 30.0);
// Import unbinned data
RooDataSet data("data","data",RooArgSet(x,c),Import(*myTree));
```

Exemplo:

1. Abrir o arquivo no terminal e verificar os observáveis : $ ls /opendata/ eos **example_data.root**

```
eliza@93f99f4d0fb7:~/Aula_IntroducaoAnalaiseDados_2024_02/testes/RooFit$ root -l
root [0] TFile *file = TFile::Open("/opendata/example_data.root");
root [1] .ls
TFile**         /opendata/example_data.root
 TFile*         /opendata/example_data.root
  KEY: TNtupleD nt;1    Unbinned data
  KEY: TH1D     hist;1  Binned data
root [2] TTree *tree = (TTree *)file->Get("nt");
root [3] tree->Print();
******************************************************************************
*Tree    :nt        : Unbinned data                                          *
*Entries :    10000 : Total =           81050 bytes  File  Size =      76111 *
*        :          : Tree compression factor =   1.06                       *
******************************************************************************
*Br    0 :mass      : mass/D                                                 *
*Entries :    10000 : Total  Size=      80696 bytes  File Size  =      75719 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   1.06     *
*............................................................................*
root [4]
```

o nome correto da branch do observável

Importando dados

2. Importe o observável *unbinned*

```
#include "TTree.h"
#include "TH1.h"
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooArgSet.h"
#include "RooPlot.h"
#include "RooFit.h"
#include "TCanvas.h"
#include <iostream>

void importarDados() {

    // Abrir o arquivo ROOT
    TFile *file = TFile::Open("example_data.root");


    TTree *myTree = (TTree *)file->Get("nt");

    // Definir a variável observável com o nome 'mass'
    RooRealVar x("mass", "mass", 0.0 , 2.0);

    // Importar dados unbinned com RooDataSet
    RooDataSet dataUnbinned("dataUnbinned", "Unbinned data", RooArgSet(x), RooFit::Import(*myTree));
    std::cout << "Dados importados: " << std::endl;
    dataUnbinned.Print("v");

    TCanvas *c = new TCanvas("c", "Imported Data", 800, 600);

    RooPlot* frame = x.frame();

    dataUnbinned.plotOn(frame, RooFit::MarkerColor(kBlue), RooFit::Name("Unbinned"));

    frame->Draw();

    c->Draw();

    c->SaveAs("ImportarDadosUnbinned.png");

    file->Close();
}
```