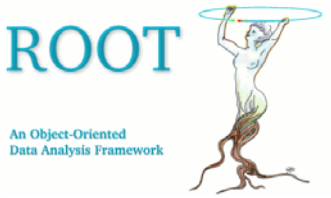
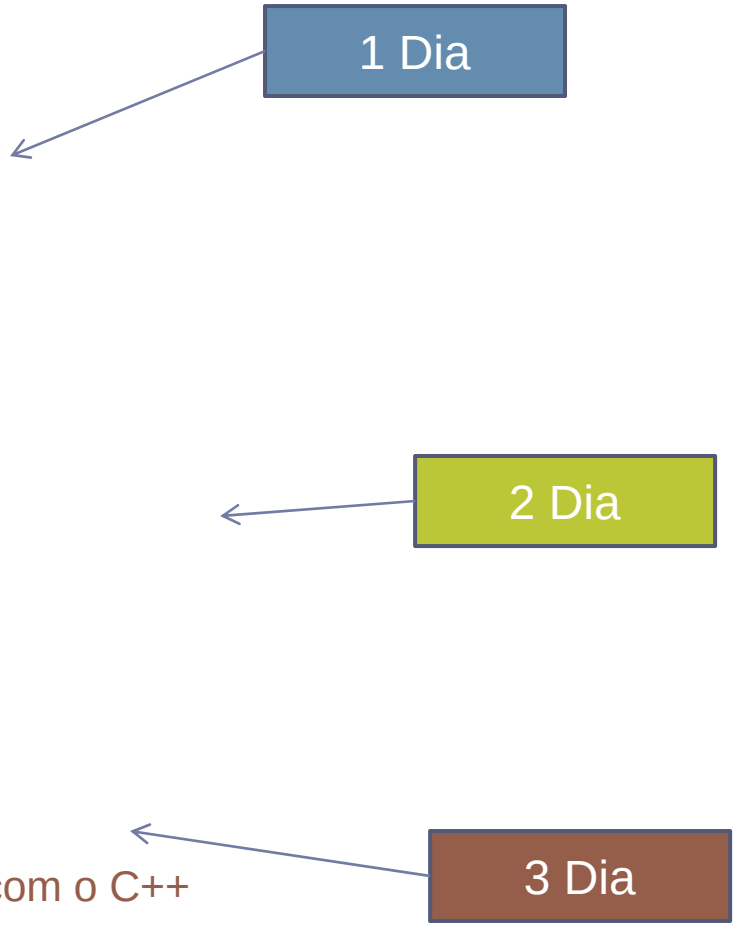


C++ e ROOT via *ssh*

d0server/access

programa

1. o *ssh*
   1.1 *ssh* e o conceito de *grid computing*
   1.2 acessando a d0server
   1.3 comandos básicos de navegação
   1.4 *up* e *down*load de arquivos
2. C++ e ROOT via ssh
   2.1 abrindo o ROOT na access
   2.2 funções
   2.3 gráficos
   2.4 histogramas
3. C++ e ROOT via ssh
   3.1 um pouco de C++
   3.2 rodando uma rotina C++ no ROOT
   3.3 abrindo e lendo arquivos de texto com o C++
   3.4 automatizando a analise de dados



abrindo o ROOT

- o root é um poderoso programa de simulação e analise de dados, feito por físicos e para físicos; http://root.cern.ch
- a seguir veremos como abrir o root via ssh; para quem quiser instalar o root no seu próprio computador (altamente recomendado!), o download esta disponível no site root.cern.ch (qualquer problema mail me!)

abrindo o ROOT (windows)

- baixe o putty e instale o xming (usaremos estes programas agora porque o OpenSSH não suporta abrir as telas gráficas do root)
- configure o putty como mostrado ao lado
- execute o xming 

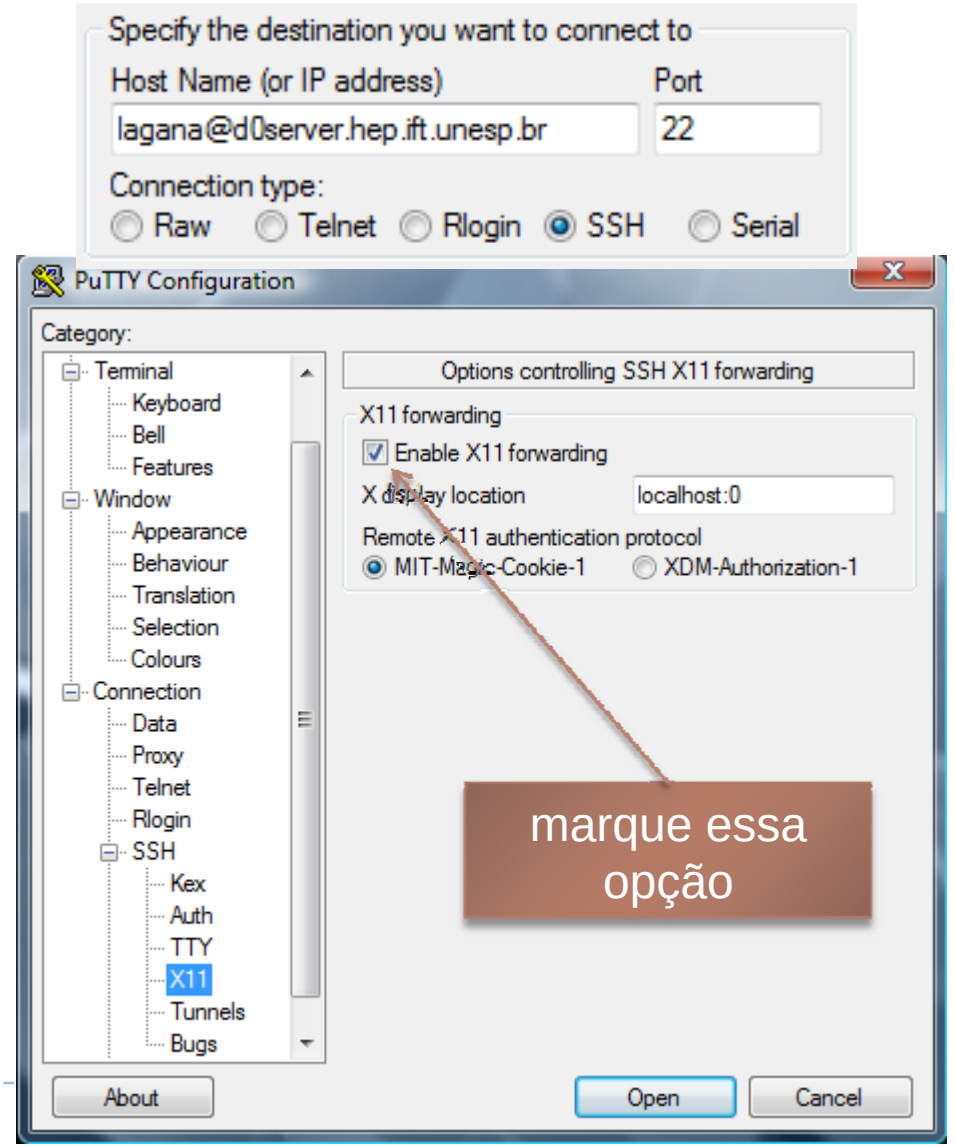
- entre na sua conta d0 pelo putty
- copie o arquivo .srt_envrc para seu diretorio home

```
cp /home/lagana/.srt_envrc /home/seunome
```

- digite o comando

```
setup D0RunII p18.10.00
```

- digite o comando root
- *Vualá!*



abrindo o ROOT (linux/mac)

- no linux as coisas são bem mais fáceis. o ssh do linux já suporta abrir telas gráficas, basta usar a opção –X para logar

  ssh –X usuario@d0server.hep.ift.unesp.br

- copiar o tal arquivo .srt_envrc:

  cp /home/lagana/.srt_envrc /home/seunome

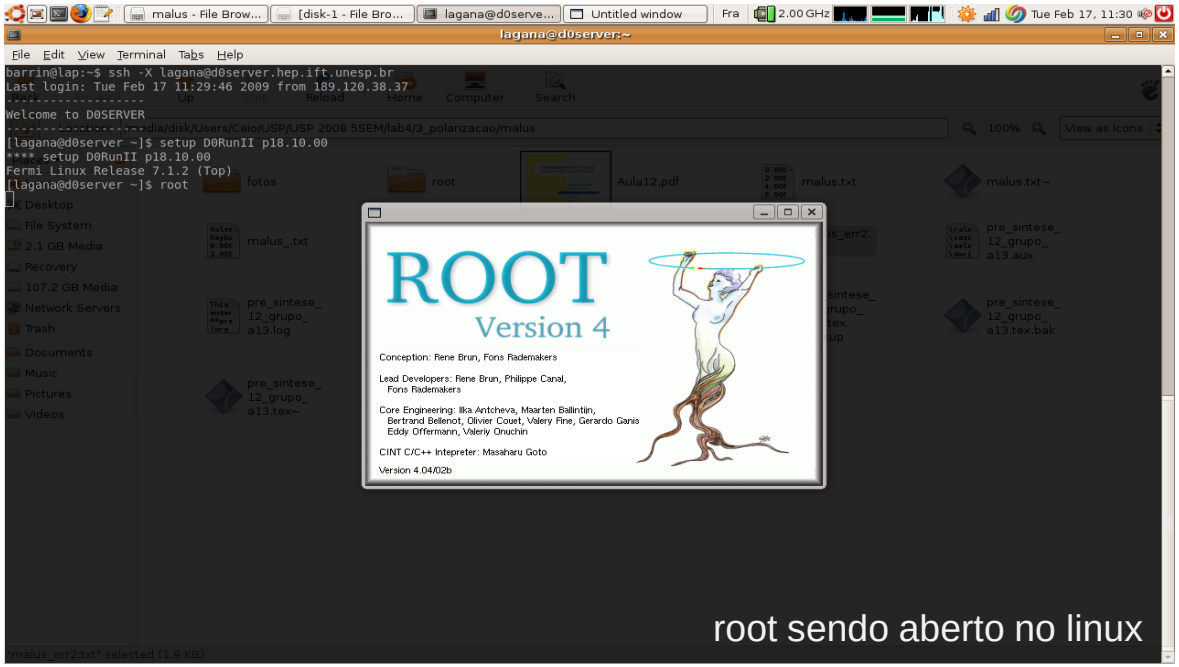
- dar o tal comando:

  setup D0RunII p18.10.00

- e abrir o root:

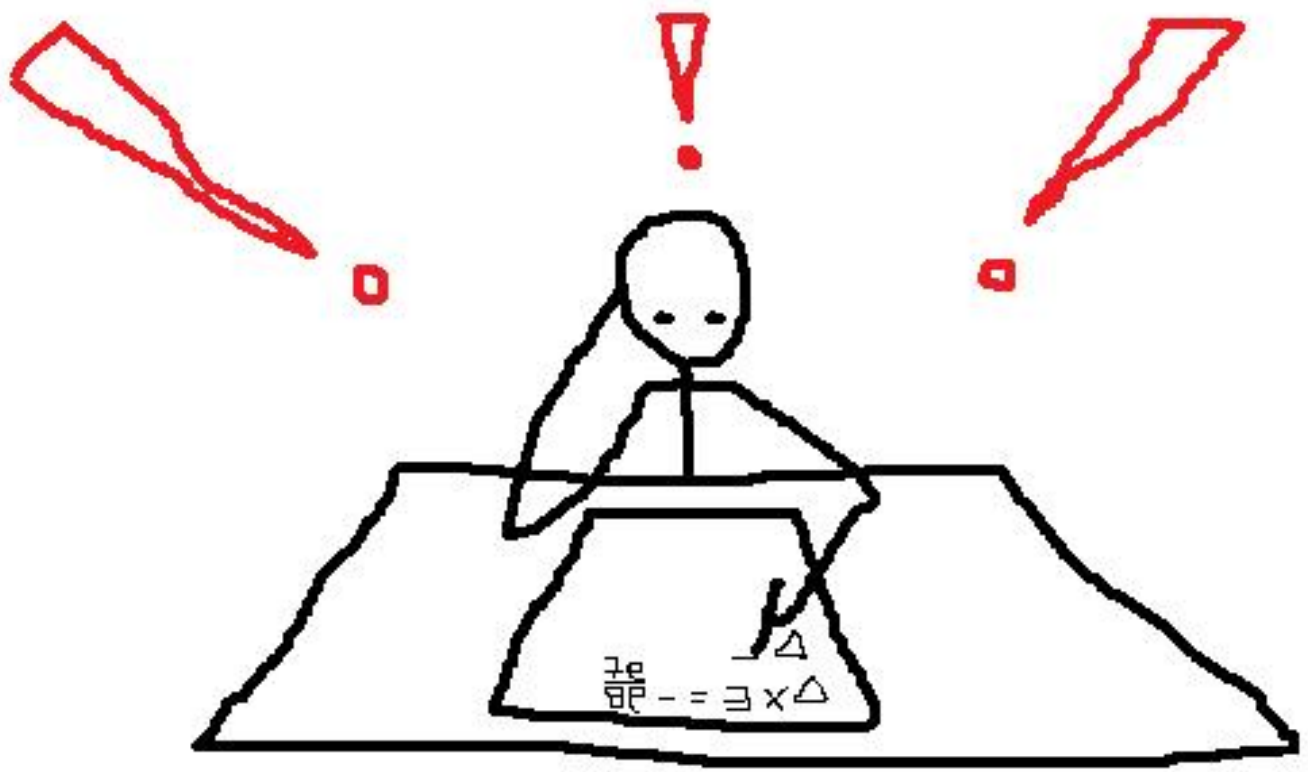
  root

- e agora que tudo esta funcionando, vamos fazer um pouco de física!



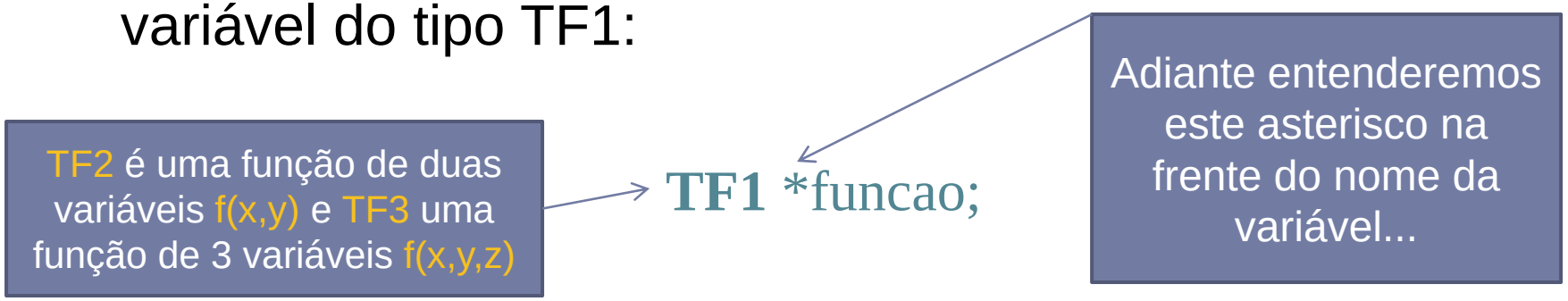
root sendo aberto no linux

exercício: abrir o root na sua conta da d0



funções

- no root, uma função *f(x)* é uma variável do tipo TF1 (da mesma forma que um numero é uma variável float no C)
- assim, para criar uma função basta declararmos uma variável do tipo TF1:



- analogamente como se cria uma variável no C:

```
float numero;
```

funções

repare que aqui não vai o asterisco

TF1 *funcao;

- para configurar os atributos da função fazemos assim:

funcao = **new** TF1("nome-da-funcao", "sin(x)", 0, 5)

aqui é o nome da funçao; este nome é irrelevante, o sin(x) não precisa chamar "seno"

este argumento é a formula propriamente dita. a letra "x" é reservada para a variável *x*

aqui vai o intervalo que queremos que a função seja criada

funções

```
TF1 *funcao;
funcao = new TF1("nome-da-funcao", "sin(x)", 0, 5)
```

- tendo criado a nossa função, podemos alterar uma série de atributos dela usando a flechinha ->

```
funcao->SetLineColor(2);
funcao->SetLineWidth(1);
```

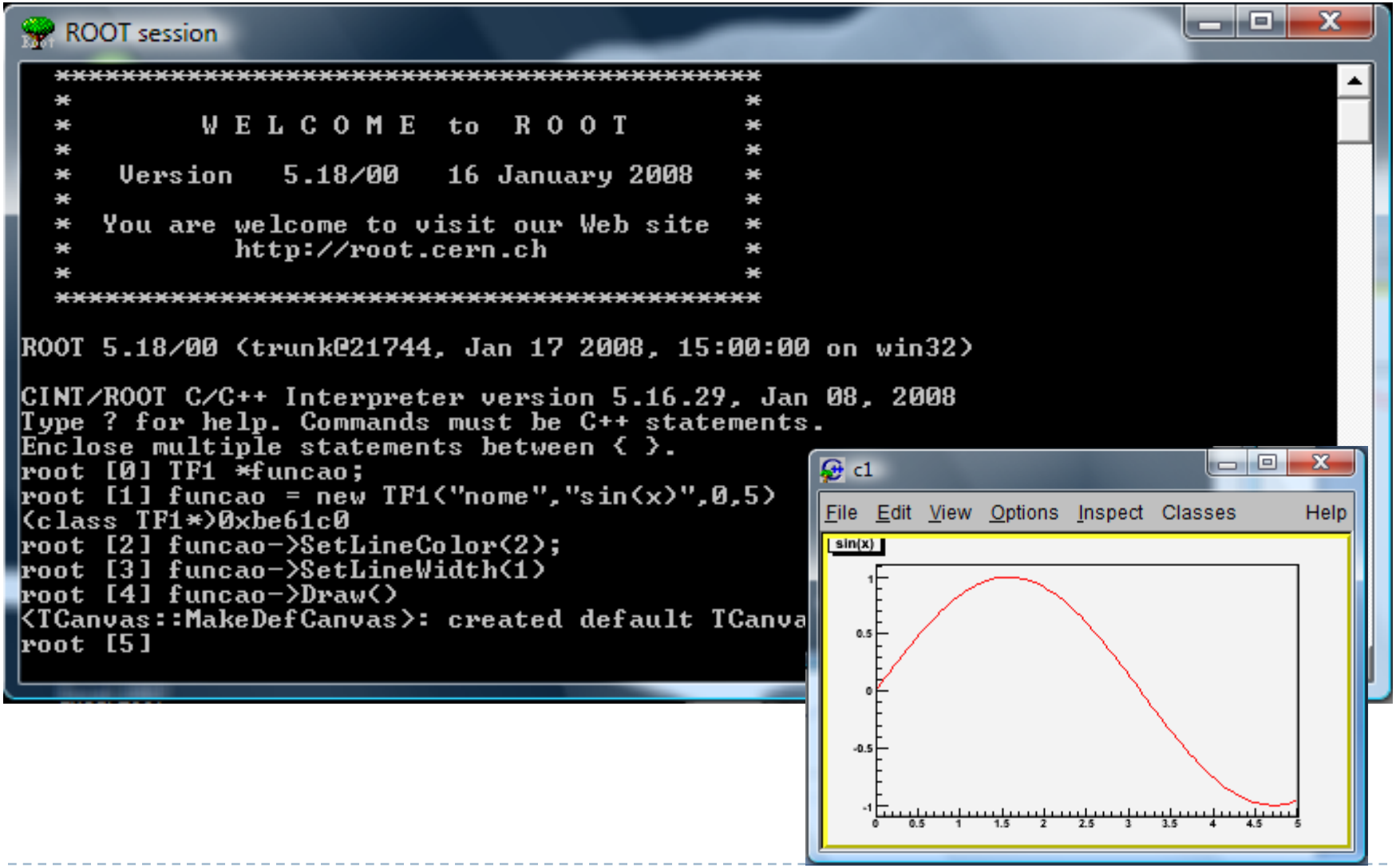
altera a cor da linha do gráfico

altera a espessura da linha do gráfico

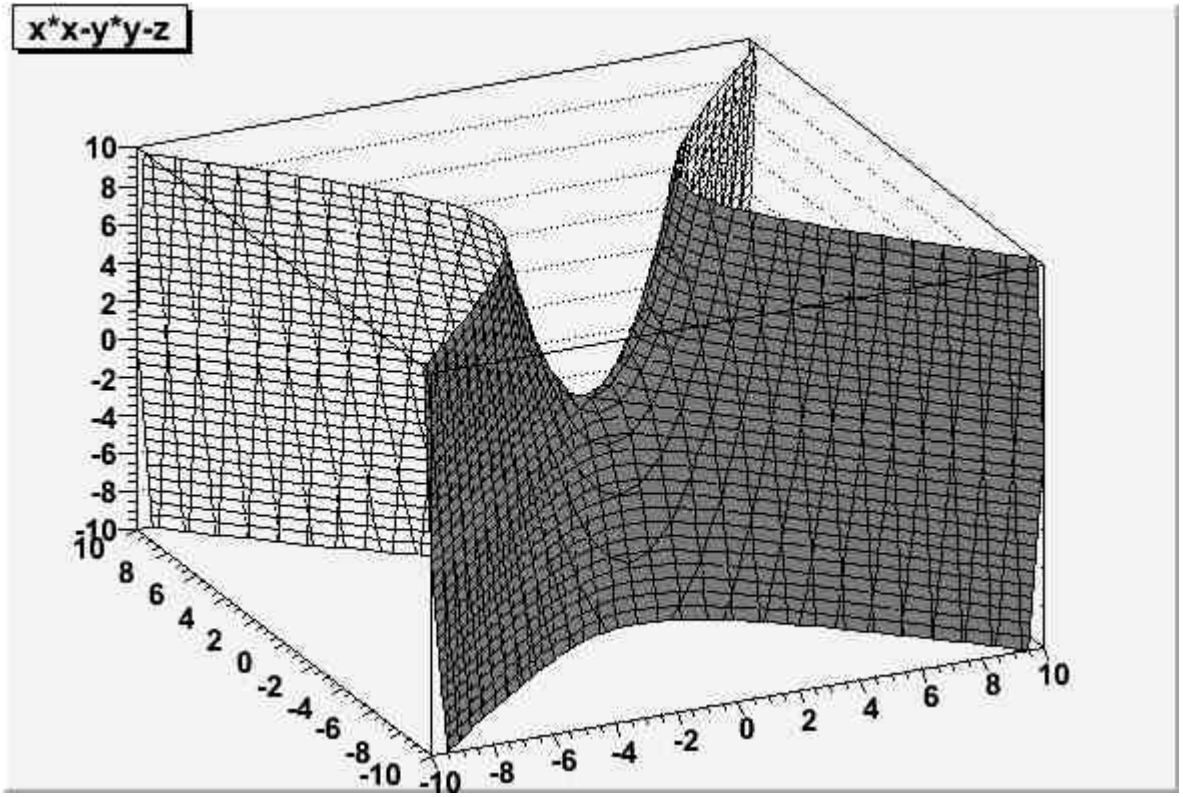
- e finalmente desenhar o gráfico!

```
funcao->Draw();
```

funções

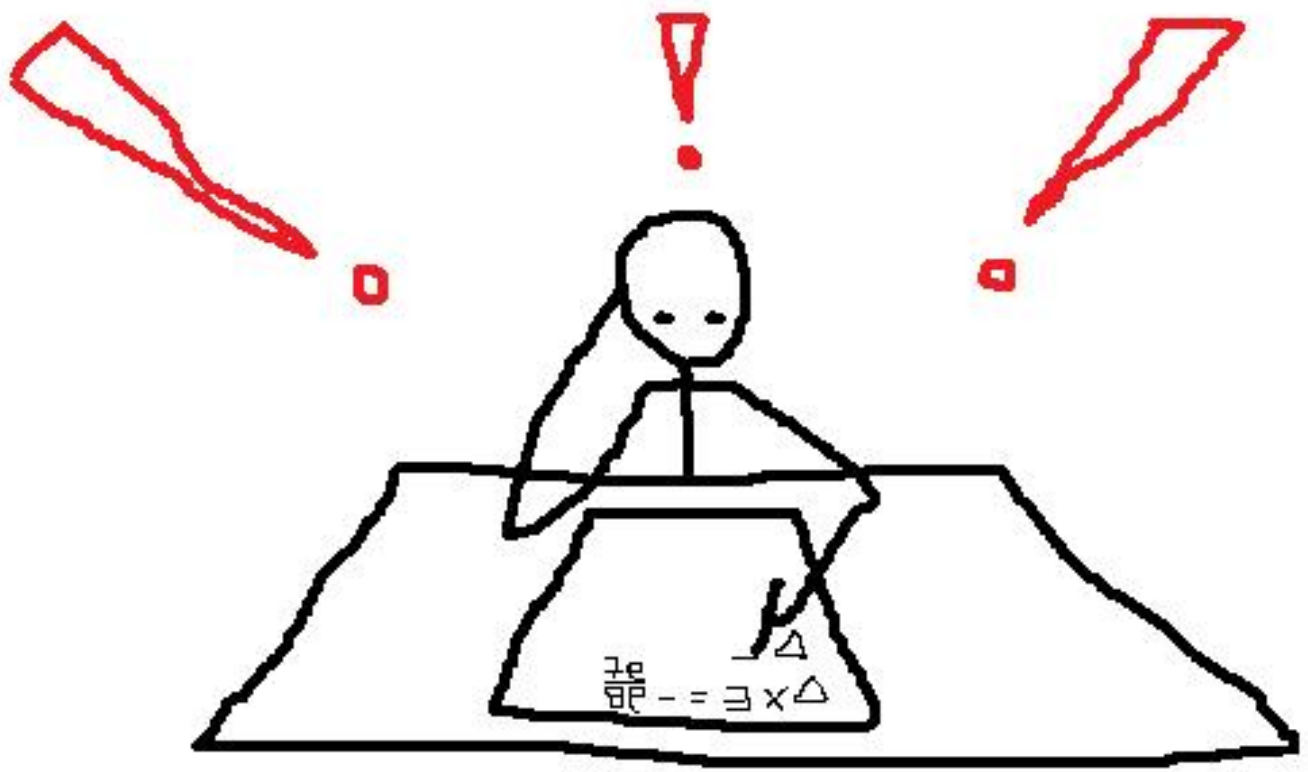

ROOT session
****
*   *
*   W E L C O M E to R O O T   *
*   *
*   Version   5.18/00   16 January 2008   *
*   *
* You are welcome to visit our Web site   *
*   http://root.cern.ch   *
*   *
****
ROOT 5.18/00 (trunk@21744, Jan 17 2008, 15:00:00 on win32)
CINT/ROOT C/C++ Interpreter version 5.16.29, Jan 08, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TF1 *funcao;
root [1] funcao = new TF1("nome","sin(x)",0,5)
(class TF1*)0xbe61c0
root [2] funcao->SetLineColor(2);
root [3] funcao->SetLineWidth(1)
root [4] funcao->Draw()
<TCanvas::MakeDefCanvas>: created default TCanva
root [5]
c1
File Edit View Options Inspect Classes Help
sin(x)
1
0.5
0
-0.5
-1
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

funções



e a titulo de curiosidade, um plot de $z = x^2 - y^2$

exercício: criar e desenhar uma função



gráficos

- podemos declarar dois tipos de gráficos no root:
- gráfico sem barra de erros

```
TGraph *grafico_sem_erros;
```

- e gráfico com barra de erros

```
TGraphErrors *grafico_com_erros;
```

```
root [8] TGraph *grafico_sem_erros;
root [9] TGraphErrors *grafico_com_erros;
root [10] _
```

gráficos

TGraphErrors *grafico_com_erros;

- suponhamos agora que queremos preencher a variável grafico_com_erros com dados de um arquivo dados.txt

```
grafico_com_erros = new TGraphErrors("/home/dados.txt","%lg %lg %lg")
```

aqui vai o endereço completo do arquivo de dados

aqui colocamos a formato em que os dados devem ser lidos no arquivo (como se fosse um scanf(%lf) no C); neste caso os dados estão dispostos da seguinte forma no arquivo:

x	y	y_err
1	2.3	0.7
1.2	4.7	0.5
...		

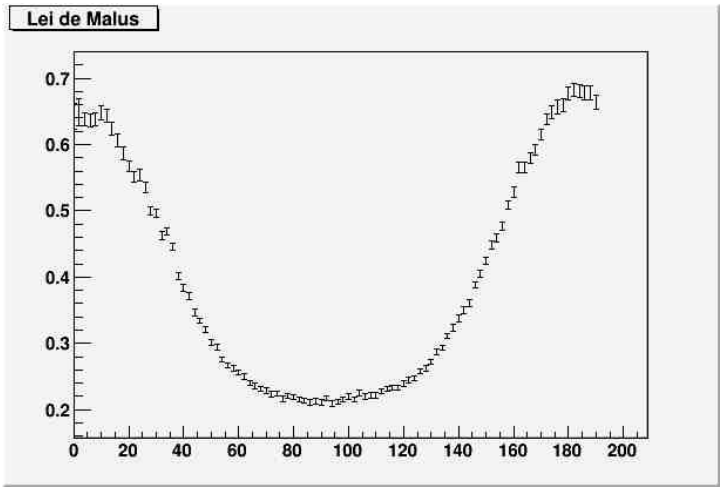
gráficos

- desenhamos o gráfico

```
grafico_sem_erros->Draw("ap");
```

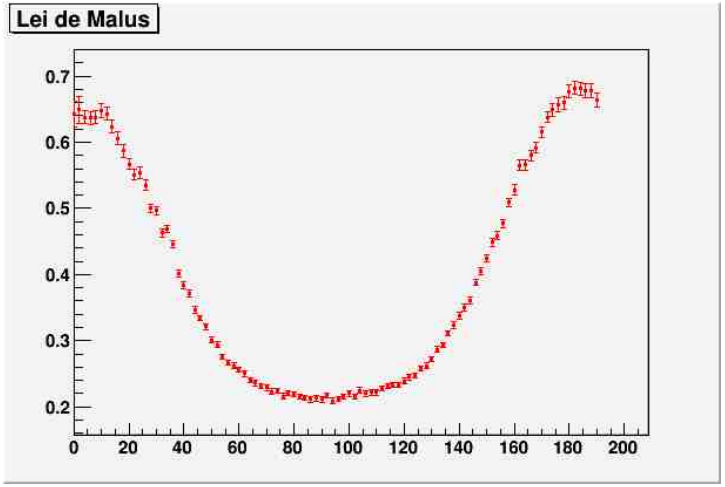
- podemos modificar alguns atributos

```
grafico_sem_erros->SetMarkerStyle(7);
grafico_sem_erros->SetMarkerColor(4);
grafico_sem_erros->Draw("ap");
```



antes...

e depois!



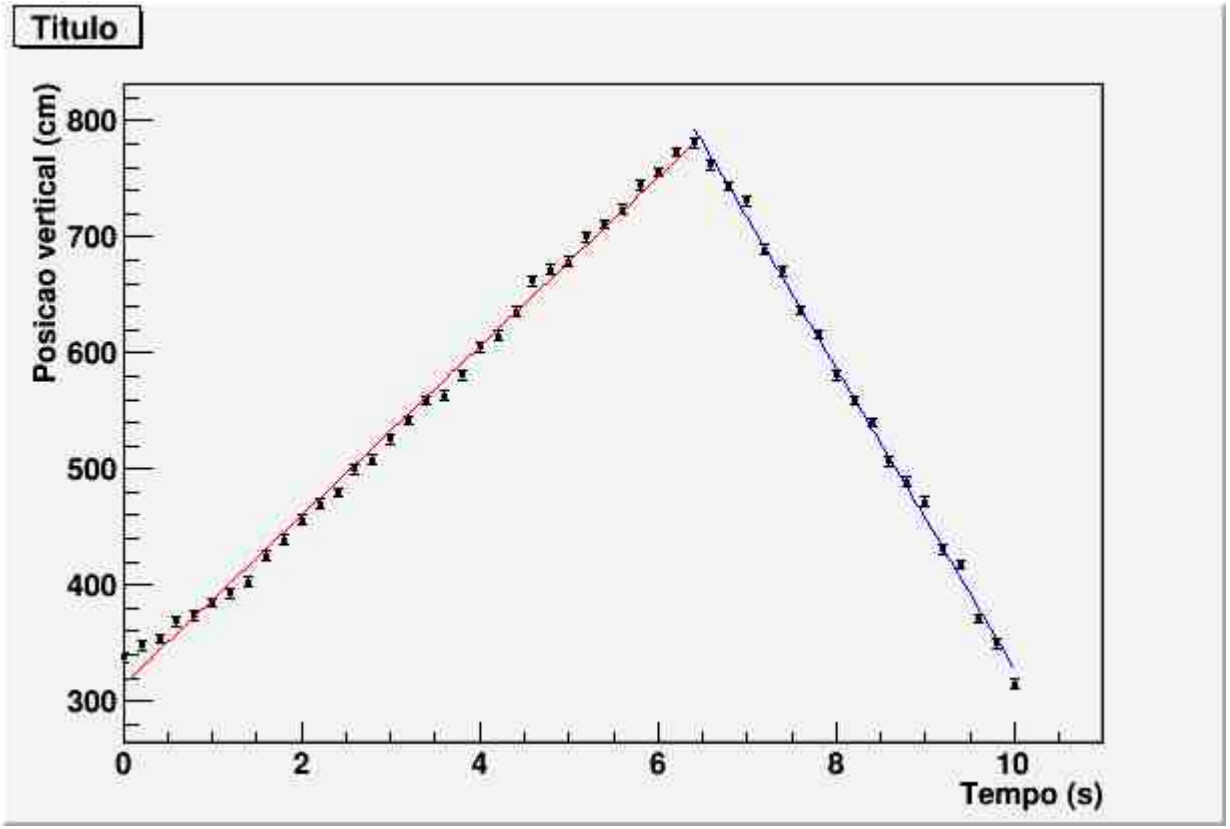
gráficos

- e como a primeira coisa que um físico pode querer de um gráfico é ajustar uma função a ele, aqui vai a receita:

[0], [1], [2]... são parâmetros a serem ajustador pelo root; neste caso, definimos uma reta y = Ax + B

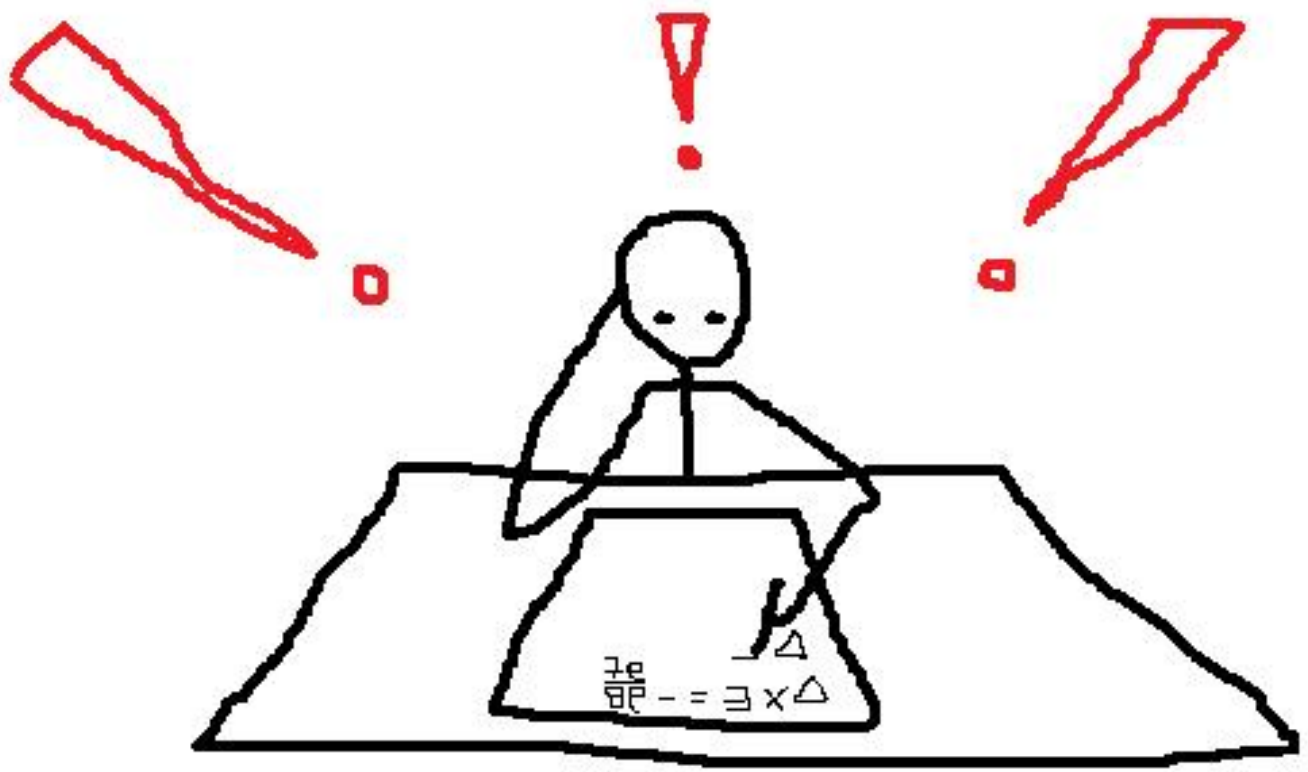
```
TF1 *funcao; //cria a variável da função
funcao = new TF1("fit","[0]*x + [1]", 0, 10); //configura a função
TGraphErrors *grafico_com_erros; //cria a variável do gráfico
grafico_com_erros = new TGraphErrors("/home/dados.txt","%lg %lg %lg")
grafico_com_erros->Fit(funcao); //fita a função no gráfico
grafico_com_erros->Draw("ap"); //desenha o gráfico fitado
```

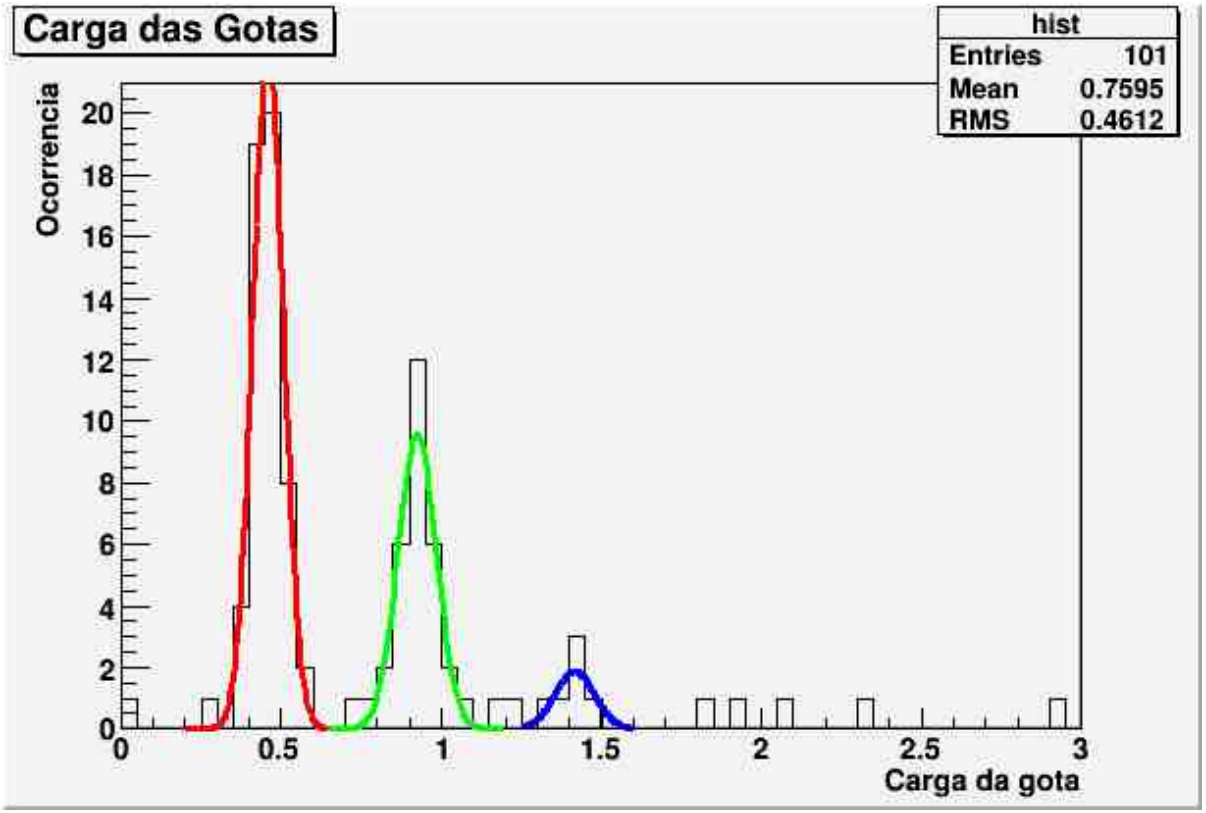
gráficos



exemplo de um gráfico ajustado por duas funções

exercício: criar um gráfico e ajustar uma função a ele; tem arquivos com dados em lagana/dados/





histogramas

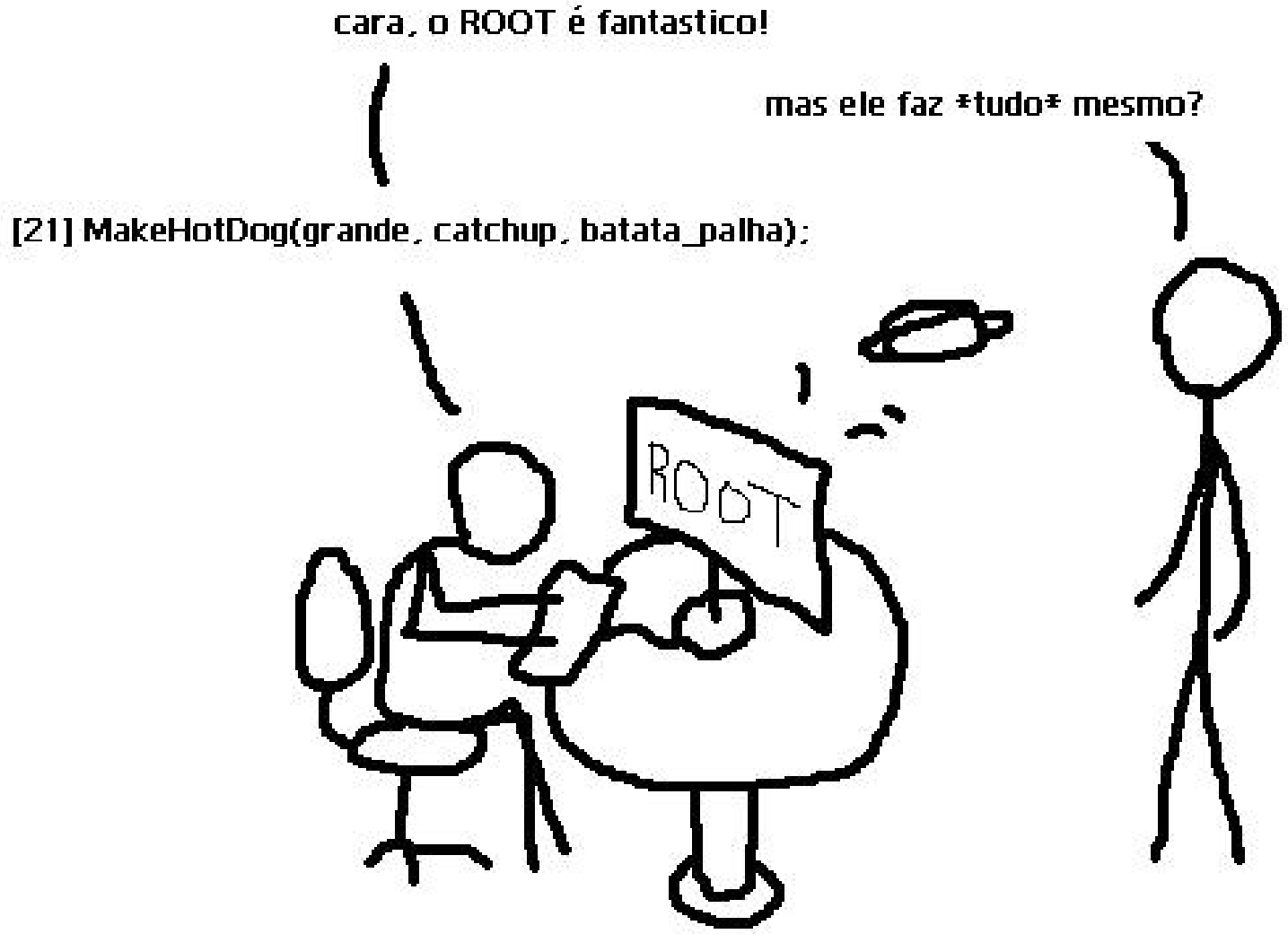
é possível ajustar curvas a histogramas como se ajusta a dados (x,y)

para montar um histograma, basicamente criamos a variável

```
TH1F *histograma;
histograma =  new TH1F("nome", "titulo", n_bins, x_min, x_max);
```

e enchemos com dados

```
histograma->Fill(dado);
```


cara, o ROOT é fantastico!
mas ele faz *tudo* mesmo?
[21] MakeHotDog(grande, catchup, batata_palha);
ROOT