

Temas avanzados en física computacional Análisis de datos

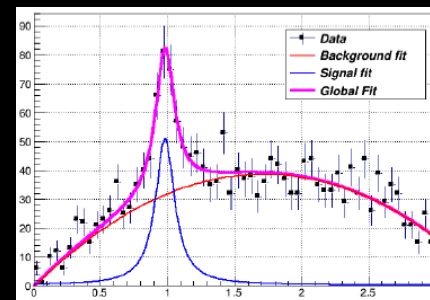
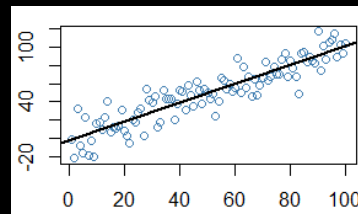
Semestre

2016-I

Clase-5

José Bazo

jbazo@pucp.edu.pe



estadística
decision
learning
minimizaciones
redes
ajustes
trees
lenguaje
datos
modelamiento
visualización
machine
analisis
neuronales
regresión
multivariate
programación
funciones
probabilidad
manipulación
pruebas
framework
modelos
ROOT
R
TMVA
distribución
ciencia
estimación

- ✓ Introducción al análisis de datos y data science
- ✓ Lenguaje de programación R
- ✓ ROOT Data Analysis Framework
- ✓ Manipulación y visualización de datos

5. Modelamiento estadístico

6. Machine Learning

7. TMVA (Toolkit for Multivariate Data Analysis)

5. Ajustes y minimizaciones

F. James. [MINUIT Reference Manual](#). 1994. CERN

ROOT [Users Guide](#). 2014. CERN

ROOT [Fitting Tutorials](#).

Ajustes con ROOT se basan en clase: **TMinuit**
Fred James @CERN '70s



Conversión a C++ del paquete original en Fortran

Ajustar una función multiparamétrica mediante:

- χ^2
- Likelihood: NLL

Clase:

```
ROOT::Fit::Fitter fitter;  
fitter.Config().MinimizerOptions()
```

Librerías de minimizadores: **Minuit, Minuit2, Fumili, GSL, Genetics**

```
ROOT::Math::MinimizerOptions::SetDefaultMinimizer("Minuit2");
```

ROOT::Fit::Chi2FCN

- Mínimos cuadrados usando errores observados (Neyman chi-squared)
- Mínimos cuadrados usando errores esperados de la función (Pearson chi-squared)
- Binned likelihood fit (**ROOT::Fit::PoissonLikelihoodFCN**)
- Unbinned likelihood fit, si se tienen en el histograma los datos originales usados para rellenarlo (**ROOT::Fit::LogLikelihoodFCN**)

Pasos para realizar un ajuste:

1. Crear objeto con datos para ajustar
2. Crear función modelo
3. Configurar el ajuste
4. Realizar el ajuste a los datos
5. Examinar el resultado

H1FitChisquare:

Calcula χ^2 entre función a ajustar (modelo, valor esperado) y datos (observaciones, histograma) y encuentra parámetros que den el mínimo χ^2

$$\chi^2 = \sum_{i,j} (x_i - y_i(a)) V_{ij} (x_j - y_j(a))$$

$$\chi^2(\alpha) = \sum_{i=1}^n \frac{f(x_i, \alpha) - e_i)^2}{\sigma_i^2}$$

V inversa de matriz de errores (covarianza) de observaciones (si son independientes (e.g. bins en histograma) se convierte en diagonal)

α vector de parámetros a ajustar

Para e_i = número enteros de eventos (bins en histograma sin pesar) $\rightarrow \sigma_i^2 = e_i$

Si no se conocen los σ_i , errores de mediciones, los valores de errores en parámetros no tienen significado.

Si se sobrestiman por un factor β , sucede lo mismo para los errores de los parámetros

- **Probabilidad**: se usa antes de obtener los datos para describir resultados posibles dados valores fijos de parámetros
- **Likelihood o verosimilitud**: se usa después de obtener los datos para describir una función de los parámetros dado un resultado

Likelihood de un conjunto de valores de parámetros, θ , dados los resultados x , es igual a la probabilidad de esos resultados observados dados los valores de los parámetros:

$$\mathcal{L}(\theta|x) = P(x|\theta) = f(x|\theta) \text{ prob. density. func. (pdf)}$$

El logaritmo de likelihood, log-likelihood, es más fácil de usar ya que:

- Logaritmo es una función que crece monótonicamente
- Alcanza su máximo en el mismo punto que la original

Para hallar el máximo se requiere derivar y suele ser más sencillo con log-likelihood: e.g. Es más fácil derivar una suma ($\log(ab)=\log(a)+\log(b)$) que un producto.

H1FitLikelihood :

Negative log-likelihood function que se puede minimizar

$$F = - \sum_i \ln f(x_i, a)$$

$$F = -2 \log(\text{likelihood})$$

x: vector de observaciones

a: parámetros para ajustar

f: hipótesis, función normalizada:

$$\int f(x_i, a) dx_1 dx_2 \dots dx_n = \text{constant}$$

Independiente de parámetros ajustados

Valor de likelihood en el mínimo no tiene significado

MINUIT ofrece diferentes algoritmos de minimización:

- **MIGRAD**: el mejor para casi todas las funciones. Debilidad: depende del conocimiento de primeras derivadas, de lo contrario falla.
- **SIMPLEX**: robusta pero lenta, no da errores confiables y puede no converger a tiempo.
- **MINOS**: funciona si se ha hallado un buen mínimo y matriz de error. Da errores asimétricos de parámetros que incluyen correlaciones y no-linealidades.
- **HESSE**: error calculado de invertir la matriz completa de segundas derivadas por diferencias finitas

Métodos dan los mismos errores en parámetros ajustados si:

1. Modelo para ajustar (χ^2 o NLL) es función lineal de los parámetros
2. Infinitos datos observados

Da errores (1σ) simétricos para parámetros calculados de matriz de covarianza.
Si no hay límites en parámetros, el error es $\text{sqrt}(\text{diag}(\text{matriz}))$

$$\Sigma_{ij} = \text{cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)]$$

$$\text{Covar}(x_i, x_j) = \text{Covar}(x_j, x_i)$$

$$\text{cov}(X, X) = \text{Var}(X) \equiv \sigma^2(X)$$

Confianza en errores estimados. Indicios de problemas:

- Mensajes de alerta
- Falla al encontrar nuevo mínimo
- Valor de EDM (distancia estimada al mínimo muy grande)
- Coeficientes de correlación $=0$ o >0.99
- Parámetro cercano a límite

Internamente MINUIT pasa de rango finito en límites de parámetros a cualquier valor con transformación no lineal. Valores cerca de límites pueden ser indistinguibles.

$$P_{\text{int}} = \arcsin \left(2 \frac{P_{\text{ext}} - a}{b - a} - 1 \right) P_{\text{ext}}$$

Si se usan limites en parámetros tener en cuenta:

- Solo si necesario para prevenir valores no físicos
- Si parámetro está cerca de límites en el mínimo:
 - tal vez minimizador está bloqueado
 - no tiene error en una dirección y matriz de error (solo da errores simétricos) pierde sentido
- Si se halló un mínimo usando límites, rehacer el análisis de errores sin límites
- Si el parámetro está cerca del

Ajustando histograma en 1D: TH1::Fit

Funciones predefinidas:

gaus: Gaussiana con 3 parámetros: $f(x) = p_0 \cdot \exp(-0.5 \cdot ((x-p_1)/p_2)^2)$.

expo: Exponencial con 2 parámetros: $f(x) = \exp(p_0 + p_1 \cdot x)$.

polN: Polinomial con N grados: $f(x) = p_0 + p_1 \cdot x + p_2 \cdot x^2 + \dots$

landau: distribución de Landau con media y σ .

No es necesario dar valores iniciales para funciones predefinidas:

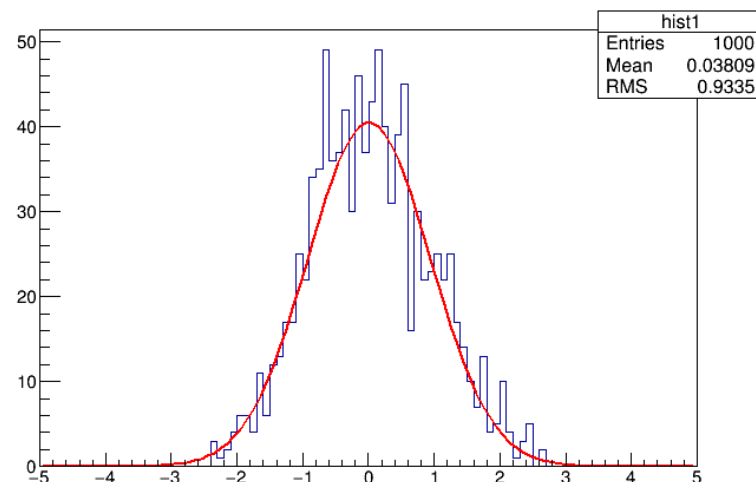
Ejemplo:

```
TH1F *hist=new TH1F("hist1","",100,-5,5);
```

```
hist->FillRandom("gaus",1000);
```

```
hist->Fit("gaus");
```

```
hist->Draw();
```



```
FCN=63.558 FROM MIGRAD      STATUS=CONVERGED      66 CALLS      67 TOTAL
                        EDM=1.40315e-012      STRATEGY= 1      ERROR MATRIX ACCURATE
```

EXT NO.	PARAMETER NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	Constant	4.05063e+001	1.67376e+000	5.19314e-003	9.59791e-007
2	Mean	9.25953e-003	3.17403e-002	1.22929e-004	-2.88124e-005
3	Sigma	9.29411e-001	2.49772e-002	2.75252e-005	1.93897e-004

hist->FillRandom("gaus",10000);

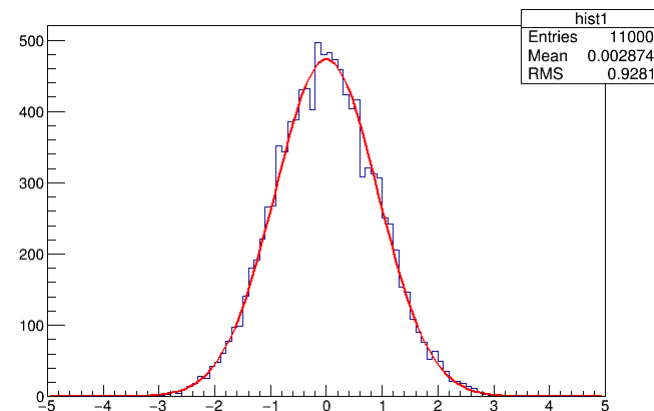
```
FCN=69.5361 FROM MIGRAD      STATUS=CONVERGED      50 CALLS      51 TOTAL
                        EDM=6.84313e-008      STRATEGY= 1      ERROR MATRIX ACCURATE
```

EXT NO.	PARAMETER NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	Constant	4.73099e+002	5.50099e+000	1.85658e-002	6.06980e-005
2	Mean	1.29169e-003	8.84365e-003	3.62169e-005	2.59699e-002
3	Sigma	9.21885e-001	6.11091e-003	7.42863e-006	3.76073e-002

Valor FCN:

χ^2 : suma de cuadrados de residuos de cada bin
normalizados por el error del bin

NLL: log - likelihood construida usando probabilidades
de Poisson para cada bin y no depende de los
parámetros de la función



Con función definida por usuario:

```
TF1 *myfunc= new TF1("myGaus", "[0]*exp(-0.5*pow((x-[1])/[2],2))",-5,5);
```

Definir valores de parámetros:

```
myfunc->SetParameter(0, 450);
```

```
myfunc->SetParameter(1, 0);
```

```
myfunc->SetParameter(2, 1);
```

Nombres

```
myfunc->SetParName(0,"Norm");
```

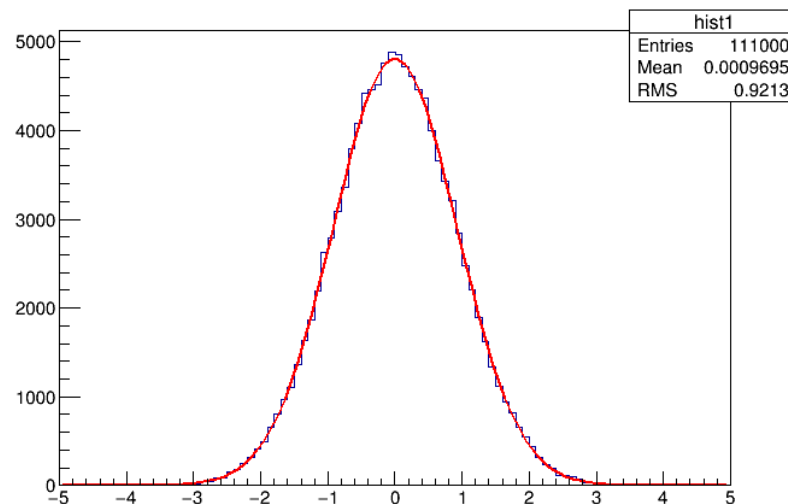
```
myfunc->SetParName(1,"mean");
```

```
myfunc->SetParName(2,"sd");
```

En alternativa:

```
myfunc->SetParameters(450,0,1)
```

```
myfunc->SetParNames("Norm","mean","sd")
```



hist->Fit("myGaus")

```
FCN=85.4134 FROM MIGRAD      STATUS=CONVERGED      144 CALLS      145 TOTAL
                        EDM=3.86758e-008      STRATEGY= 1      ERROR MATRIX ACCURATE
```

EXT NO.	PARAMETER NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	Norm	4.81187e+003	1.77552e+001	6.55814e-002	1.34732e-005
2	mean	8.63400e-004	2.76292e-003	1.25394e-005	9.11869e-003
3	sd	9.19618e-001	1.97384e-003	7.29016e-006	1.70037e-001

hist->Fit("gaus")

```
FCN=85.4134 FROM MIGRAD      STATUS=CONVERGED      60 CALLS      61 TOTAL
                        EDM=4.35402e-013      STRATEGY= 1      ERROR MATRIX ACCURATE
```

EXT NO.	PARAMETER NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	Constant	4.81186e+003	1.77552e+001	6.55811e-002	5.94016e-008
2	Mean	8.63361e-004	2.76293e-003	1.25390e-005	1.27670e-005
3	Sigma	9.19618e-001	1.97384e-003	2.64051e-006	1.36652e-003

hist->Fit("myGaus","v")

Opción Verbose

```
NOW USING STRATEGY 1: TRY TO BALANCE SPEED AGAINST RELIABILITY
*****
** 223 **MIGRAD          1345          0.01
*****
FIRST CALL TO USER FUNCTION AT NEW START POINT, WITH IFLAG=4.
START MIGRAD MINIMIZATION. STRATEGY 1. CONVERGENCE WHEN EDM .LT.1.00e-005
FCN=9827.08 FROM MIGRAD STATUS=INITIATE      8 CALLS      9 TOTAL
EDM= unknown STRATEGY= 1 NO ERROR MATRIX

EXT PARAMETER
NO. NAME VALUE ERROR STEP SIZE FIRST DERIVATIVE
1 Norm 3.67179e+003 1.15437e+001 1.15437e+001 -4.25855e-005
2 mean 1.04562e-002 2.31673e-003 2.31673e-003 1.19226e-002
3 sd 1.10000e+000 8.79216e-006 8.83645e-006 1.13167e+005
NO ERROR MATRIX
FCN=4561.06 FROM MIGRAD STATUS=PROGRESS      17 CALLS      18 TOTAL
EDM=8248.61 STRATEGY= 1 NO ERROR MATRIX

EXT PARAMETER
NO. NAME VALUE ERROR STEP SIZE FIRST DERIVATIVE
1 Norm 3.67179e+003 1.15437e+001 3.73219e-003 -7.85161e+000
2 mean 1.04562e-002 2.31673e-003 -4.20853e-008 1.47249e+003
3 sd 1.00004e+000 8.79216e-006 -1.00128e-001 4.03728e+003
MIGRAD MINIMIZATION HAS CONVERGED.
MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
-START COVARIANCE MATRIX CALCULATION
EIGENVALUES OF SECOND-DERIVATIVE MATRIX:
4.1868e-001 1.0000e+000 1.5813e+000
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=85.4134 FROM MIGRAD STATUS=CONVERGED      72 CALLS      73 TOTAL
EDM=2.61101e-007 STRATEGY= 1 ERROR MATRIX ACCURATE

EXT PARAMETER
NO. NAME VALUE ERROR STEP SIZE FIRST DERIVATIVE
1 Norm 4.81186e+003 1.77552e+001 6.55810e-002 -4.89547e-005
2 mean 8.63446e-004 2.76293e-003 1.25390e-005 2.32051e-002
3 sd 9.19618e-001 1.97384e-003 7.31357e-006 -3.23051e-001
EXTERNAL ERROR MATRIX. NDIM= 25 NPAR= 3 ERR DEF=1
3.152e+002 -5.019e-004 -2.037e-002
-5.019e-004 7.634e-006 9.609e-008
-2.037e-002 9.609e-008 3.896e-006
PARAMETER CORRELATION COEFFICIENTS
NO. GLOBAL 1 2 3
1 0.58121 1.000 -0.010 -0.581
2 0.01762 -0.010 1.000 0.018
3 0.58132 -0.581 0.018 1.000
nfo in <TMinuitMinimizer::Minimize>: Finished to run MIGRAD - status 0
```

Acceder a parámetros y resultados:

```
Double_t chi2 = myfunc->GetChisquare();  
Double_t ndf = myfunc->GetNDF();  
Double_t par0 = myfunc->GetParameter(0);  
Double_t err0 = myfunc->GetParError(0);
```

ndf: número de puntos usados en ajuste
menos número de parámetros libres

χ^2 **reducido** = $\text{chi2}/\text{ndf}$ $=85.4/73=1.17$

ndf = número de bins $\neq 0$ – parámetros

Otras funciones:

FixParameter (Int_t ipar, Double_t value)

Integral (Double_t a, Double_t b)

SetRange (Double_t xmin, Double_t xmax)

SetParLimits (Int_t ipar, Double_t parmin, Double_t parmax)

GetParLimits (Int_t ipar, Double_t &parmin, Double_t &parmax)

myfunc->SetParLimits (2, 1.1, 1.5)

```
FCN=9827.08 FROM MIGRAD      STATUS=CONVERGED      87 CALLS      88 TOTAL
                        EDM=7.35778e-013    STRATEGY= 1    ERROR MATRIX UNCERTAINTY
2.4 per cent
EXT  PARAMETER
NO.  NAME      VALUE      ERROR      STEP      FIRST
      SIZE      DERIVATIVE
  1  Norm      3.67179e+003  1.15457e+001  5.60575e-003 -1.26999e-007
  2  mean      1.04562e-002  2.31374e-003 -1.73722e-007  2.80719e-004
  3  sd        1.10000e+000  8.79147e-006 -1.28206e-005 ** at limit **
```

χ^2 reducido = χ^2/ndf =134.6

Definir función pasar el nombre al constructor TF1.

Debe tener la forma:

Double_t fitfunc(Double_t *x, Double_t *par)

x : puntero al arreglo de dimensiones. Cada elemento contiene una dimensión. (e.g. 1D solo x[0], 3D x[0], x[1], y x[2].

par: puntero al arreglo de parámetros.

Ejemplo:

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


```
Double_t arg = 0;
```

```
if (par[2] != 0) arg = (x[0] - par[1])/par[2];
```

```
Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
```

```
return fitval; }
```

```
TF1 *func = new TF1("fit",fitfunc,-3,3,3); //último valor es el número de parámetros
```

void Fit(const char *fname, Option_t *option, Option_t *goption,  opción gráfica
Axis_t xxmin, Axis_t xxmax) ->intervalo de ajuste

Opciones sobre el ajuste:

- "W" todos los pesos a 1 para bins no vacíos, ignorar barras de error
- "WW" todos los pesos a 1 incluyendo bins vacíos, ignorar barras de error
- "I" usar integral de función en bin en vez de valor central
- "L" usar método log likelihood (default χ^2)
- "Q" Modo silencioso
- "V" Verbose (default entre Q y V)
- "E" Hallar mejores estimados de errores con técnica Minos
- "M" Mejorar resultados del ajuste con Hesse
- "R" Usar rango de la función
- "N" no guardar la función gráfica, no dibujar
- "O" No dibujar resultado del ajuste
- "+" añadir función ajustada a lista de funciones
- "LL" log-likelihood mejorada para baja estadística (bin content < 100 o no entero)
- "F" si se ajusta con polN usar fitter (default linear fitter)

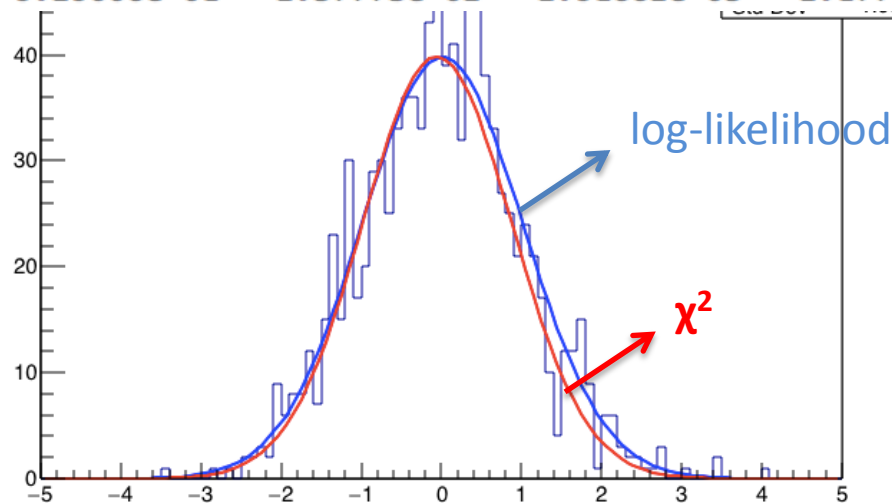
```
root [15] hist->Fit("gaus","L");
FCN=45.3593 FROM MIGRAD STATUS=CONVERGED 58 CALLS 59 TOTAL
EDM=4.25064e-09 STRATEGY= 1 ERROR MATRIX ACCURATE
```

EXT	PARAMETER				
NO.	NAME	VALUE	ERROR	STEP	FIRST
				SIZE	DERIVATIVE
1	Constant	3.97761e+01	1.54058e+00	5.88348e-03	2.77235e-05
2	Mean	1.18002e-02	3.17170e-02	1.48311e-04	7.85708e-08
3	Sigma	1.00297e+00	2.24294e-02	2.84588e-05	-4.08171e-03

ERR DEF= 0.5

```
root [16] hist->Fit("gaus","+");
FCN=75.7525 FROM MIGRAD STATUS=CONVERGED 67 CALLS 68 TOTAL
EDM=4.32161e-09 STRATEGY= 1 ERROR MATRIX ACCURATE
```

EXT	PARAMETER				
NO.	NAME	VALUE	ERROR	STEP	FIRST
				SIZE	DERIVATIVE
1	Constant	3.98366e+01	1.69178e+00	5.60562e-03	3.37534e-05
2	Mean	-5.23123e-02	3.13950e-02	1.32816e-04	-2.62145e-03
3	Sigma	9.29006e-01	2.57773e-02	2.91662e-05	2.17769e-03



```
[root [9] hist->Fit("gaus");
FCN=75.7525 FROM MIGRAD      STATUS=CONVERGED      67 CALLS      68 TOTAL
                        EDM=4.3216e-09      STRATEGY= 1      ERROR MATRIX ACCURATE

EXT PARAMETER
NO.   NAME      VALUE      ERROR      STEP      FIRST
1   Constant    3.98366e+01    1.69178e+00    5.60562e-03    3.37533e-05
2   Mean        -5.23123e-02    3.13950e-02    1.32816e-04    -2.62145e-03
3   Sigma       9.29006e-01    2.57773e-02    2.91662e-05    2.17769e-03

[root [10] hist->Fit("gaus","E");
FCN=75.7525 FROM MINOS      STATUS=SUCCESSFUL      21 CALLS      172 TOTAL
                        EDM=4.32155e-09      STRATEGY= 1      ERROR MATRIX ACCURATE

EXT PARAMETER
NO.   NAME      VALUE      ERROR      STEP      FIRST
1   Constant    3.98366e+01    1.69145e+00    -1.08789e-02    4.57758e-04
2   Mean        -5.23123e-02    3.13946e-02    2.72794e-04    1.81114e-02
3   Sigma       9.29006e-01    2.57719e-02    2.57719e-02    -1.84599e-02

[root [11] hist->Fit("gaus","M");
FCN=75.7525 FROM HESSE      STATUS=OK      16 CALLS      92 TOTAL
                        EDM=4.32152e-09      STRATEGY= 1      ERROR MATRIX ACCURATE

EXT PARAMETER
NO.   NAME      VALUE      ERROR      STEP      FIRST
1   Constant    3.98366e+01    1.69145e+00    1.12112e-03    3.37533e-05
2   Mean        -5.23123e-02    3.13946e-02    2.65631e-05    -2.62155e-03
3   Sigma       9.29006e-01    2.57719e-02    5.83323e-06    2.17484e-03
```

Generar histograma a partir de función del usuario:

```
TF1 *sqroot = new TF1("sqroot","x*gaus(0) + [3]*abs(sin(x)/x)",0,10);
sqroot->SetParameters(5,4,1,30);
TH1F *h1 = new TH1F("h1","",200,0,10);
h1->FillRandom("sqroot",100000);
h1->Draw("PEO");
h1->Fit("sqroot");
```

Visualizar resultados:

```
gStyle->SetOptFit();
gStyle->SetOptFit(mode)
```

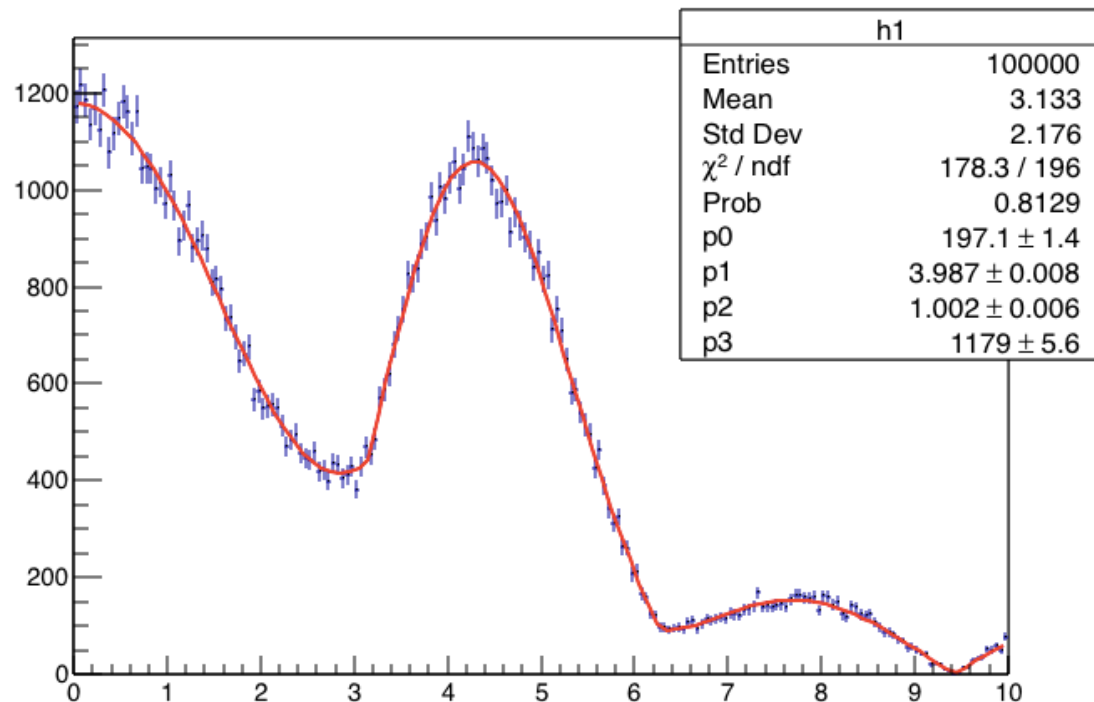
mode = pcev (default = 0111)

p=probabilidad

c=chi2/ndf

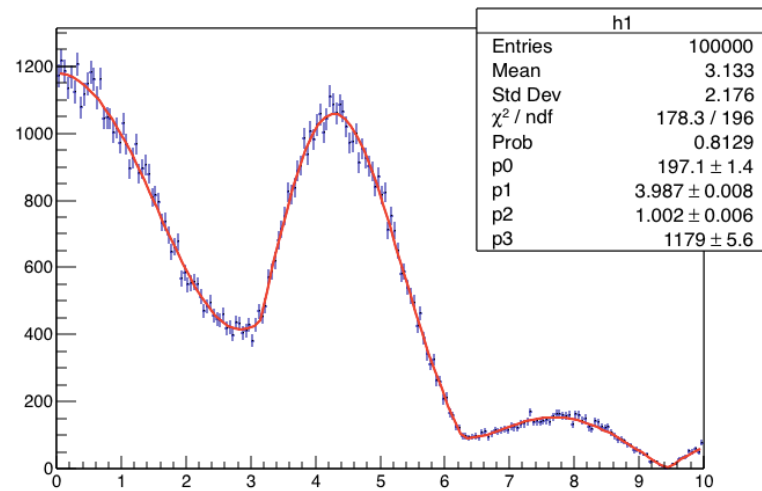
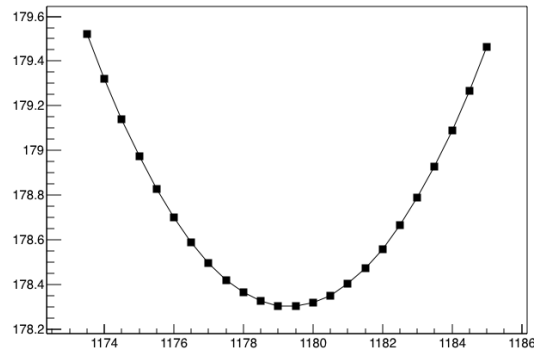
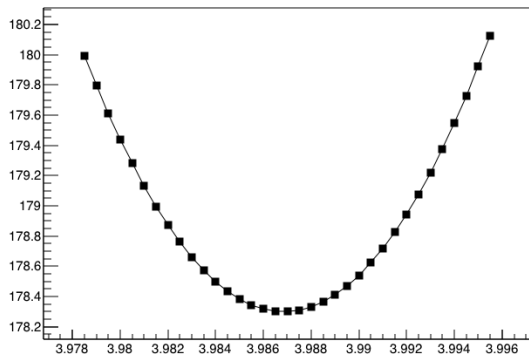
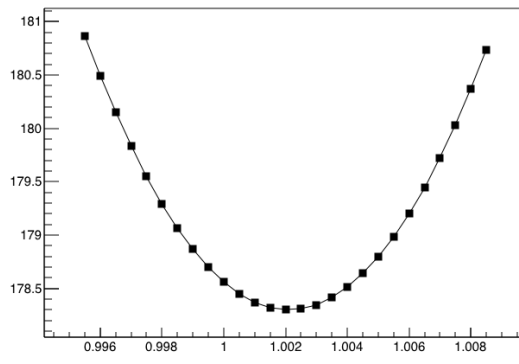
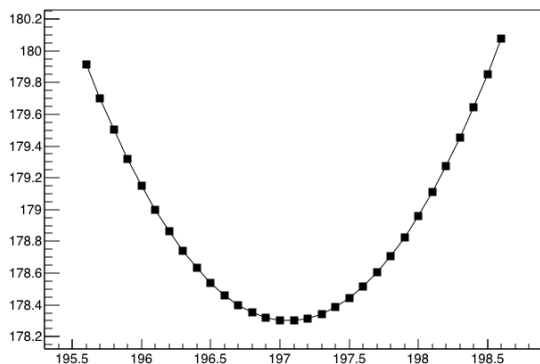
e=errores

v=nombres de parámetros



Interactuando con Minuit

```
gMinuit->Command("SCAN 1");  
TGraph *gr = (TGraph*) gMinuit->GetPlot();  
gr->SetMarkerStyle(21);  
gr->Draw("ALP");
```

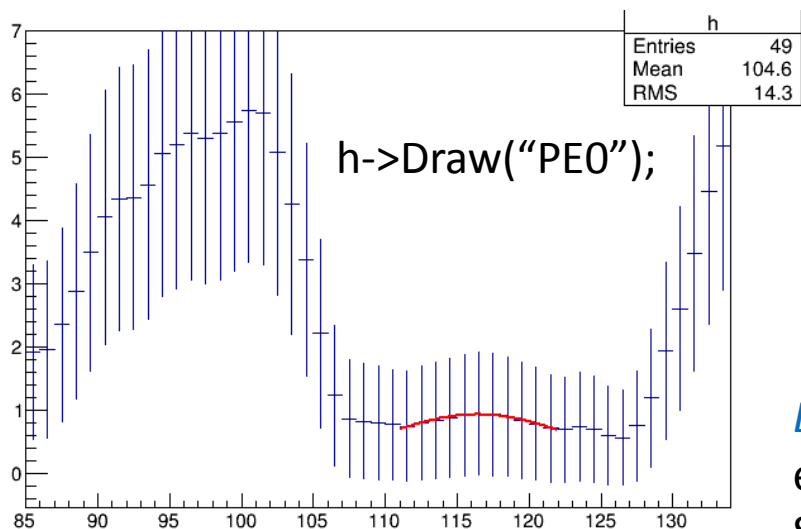
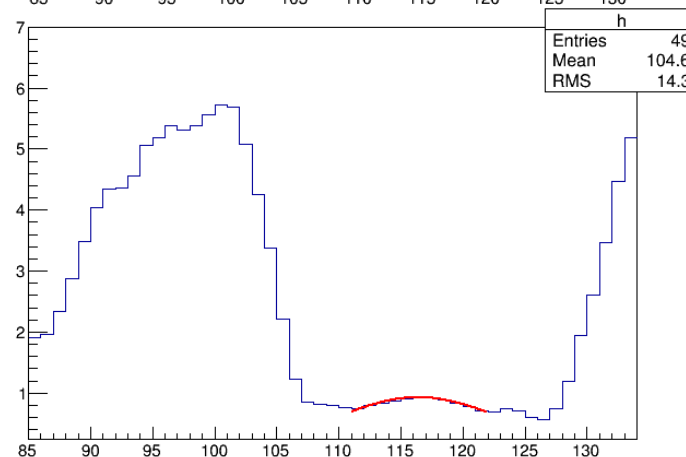
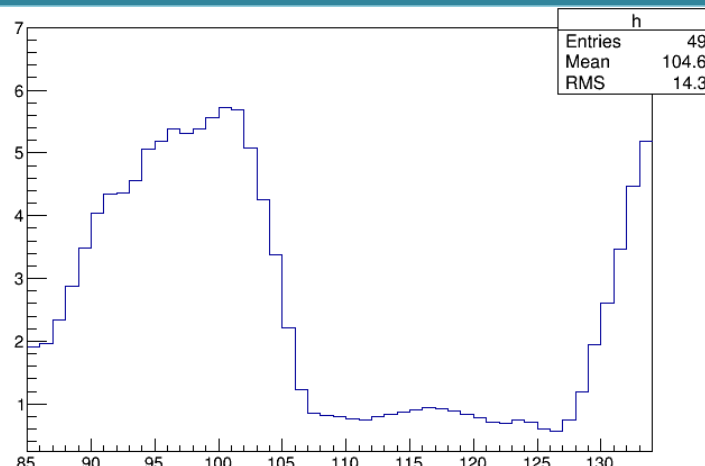


Ajustar intervalos del histograma:

Opción "R"

```
TF1 *g3 = new TF1("g3","gaus",111,122);
h->Fit(g3,"R");
```

```
FCN=0.0018845 FROM MIGRAD STATUS=CONVERGED 80 CALLS 81 TOTAL
EDM=5.367e-010 STRATEGY= 1 ERROR MATRIX UNCERTAINTY 4
.5 per cent
EXT PARAMETER
NO. NAME VALUE ERROR STEP SIZE FIRST DERIVATIVE
1 Constant 9.27737e-001 4.49328e-001 -1.80943e-004 4.55409e-005
2 Mean 1.16396e+002 2.04833e+000 -4.97279e-004 1.44065e-005
3 Sigma 2.11445e+000 1.87500e+001 1.34320e-003 -1.03929e-005
```



Errores de bins:

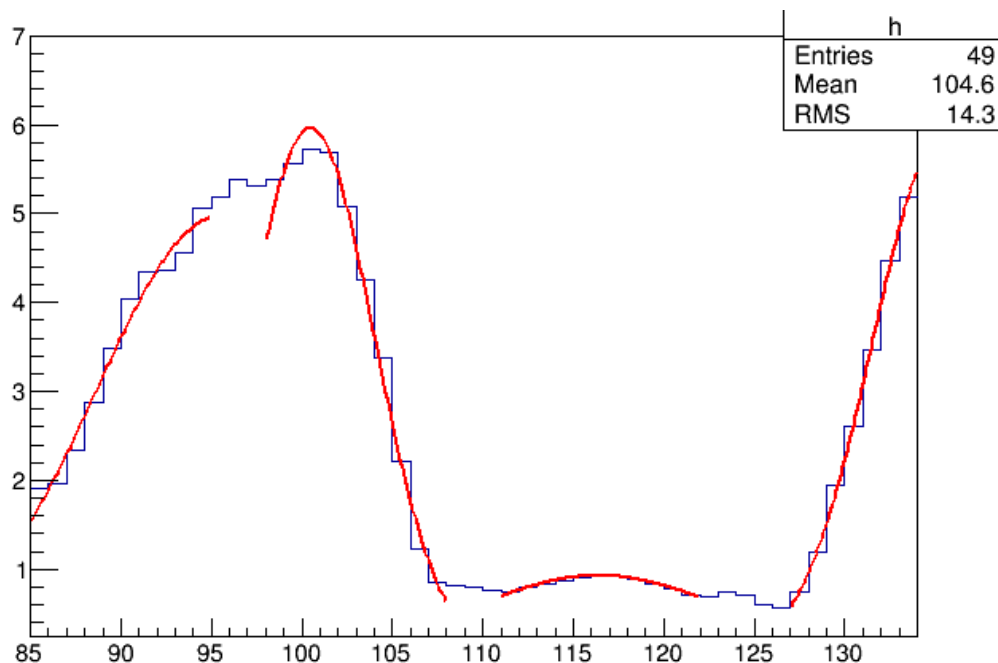
$\text{error} = \sqrt{\text{bin_content}}$

Si TH1::Sumw2 $\text{error} = \sqrt{\text{sum peso_bin}^2}$

Ajustar intervalos del histograma:

```
TF1 *g1 = new TF1("g1","gaus",85,95);
TF1 *g2 = new TF1("g2","gaus",98,108);
TF1 *g3 = new TF1("g3","gaus",111,122);
TF1 *g4 = new TF1("g4","gaus",127,135);
```

```
h->Fit(g1,"R");
h->Fit(g2,"R+");
h->Fit(g3,"R+");
h->Fit(g4,"R+");
```

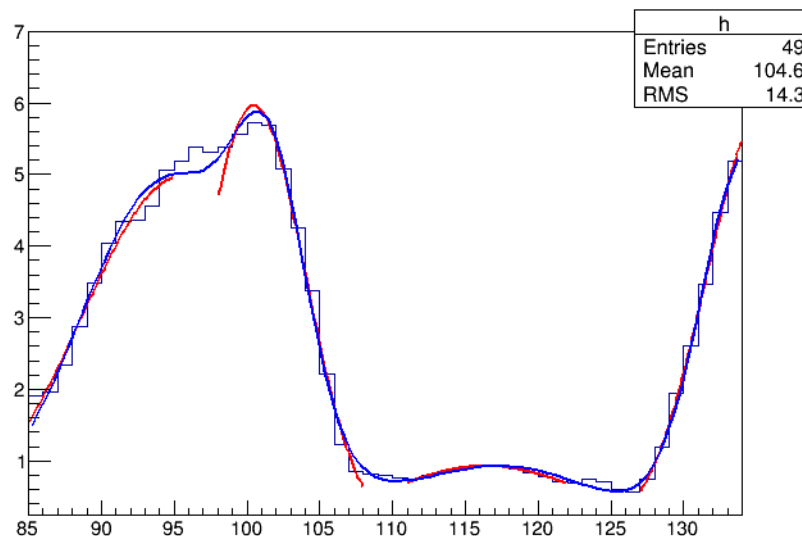
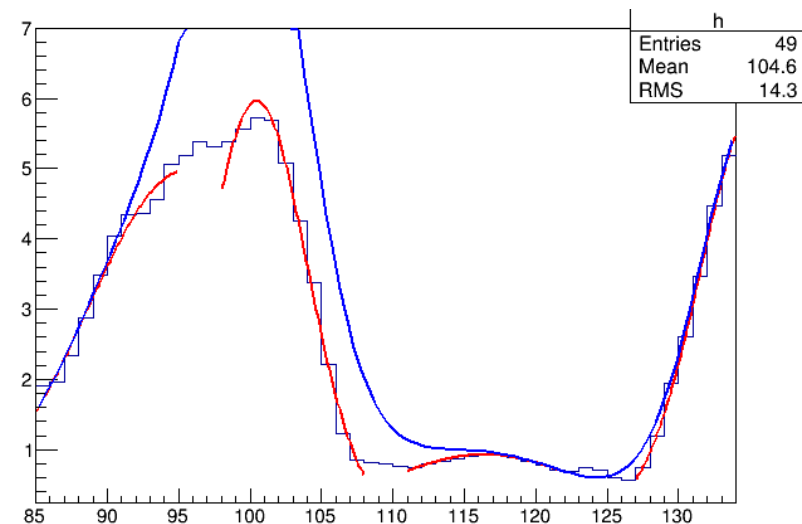


```
Double_t par[12];
g1->GetParameters(&par[0]);
g2->GetParameters(&par[3]);
g3->GetParameters(&par[6]);
g4->GetParameters(&par[9]);
TF1 *total = new
TF1("total","gaus(0)+gaus(3)+gaus(6)+gaus(9)",8
5,135);
total->SetLineColor(kBlue);
total->SetParameters(par); //es necesario dar
parámetros iniciales cuando no es una función
simple
total->Draw("same");
```

```
h->Fit(total,"R+");
```

χ^2 reducido = $\text{chi2}/\text{ndf} = 0.334/37 = 0.009$

$\text{ndf} = 49 \text{ bins} - 12 \text{ parámetros} = 37$



Ajuste de gráfico múltiple con traslape

```
TGraphErrors *gr1 = new TGraphErrors(n, x1, y1, 0, e1);
TGraphErrors *gr2 = new TGraphErrors(n, x2, y2, 0, e2);
TGraphErrors *gr3 = new TGraphErrors(n, x3, y3, 0, e3);
```

```
TMultiGraph *mg=new TMultiGraph("mg","")
```

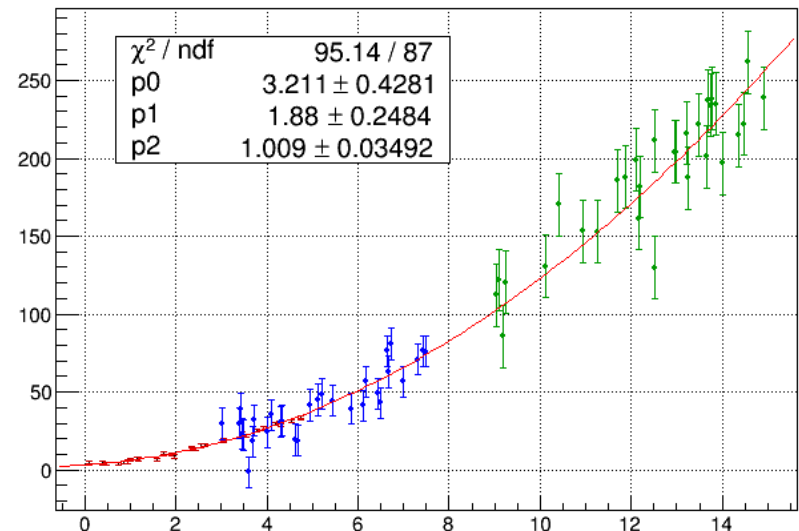
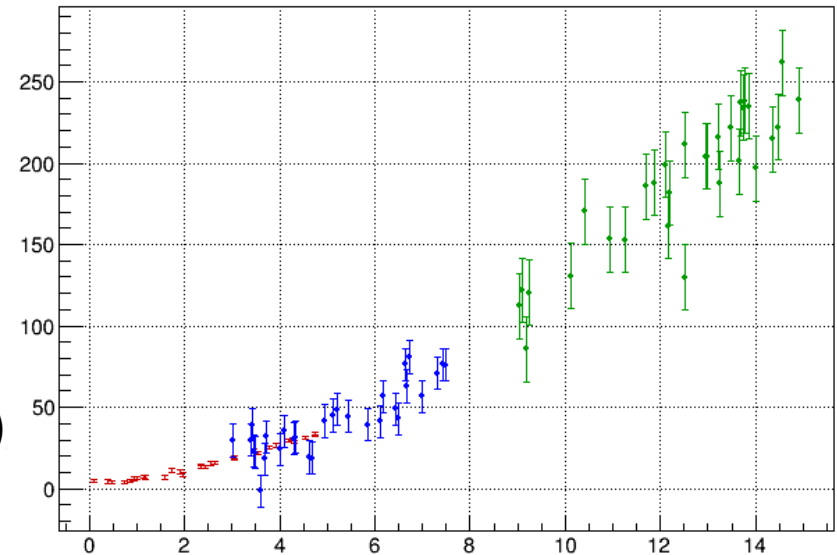
```
mg->Add(gr1);
```

```
mg->Add(gr2);
```

```
mg->Add(gr3);
```

```
mg->Draw("AP");
```

```
mg->Fit("pol2", "F");
```



```
FCN=95.1374 FROM MIGRAD      STATUS=CONVERGED      54 CALLS      55 TOTAL
                    EDM=4.06527e-020      STRATEGY= 1      ERROR MATRIX ACCURATE
```

EXT NO.	PARAMETER NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	p0	3.21134e+000	4.28055e-001	8.68677e-004	4.08980e-011
2	p1	1.87952e+000	2.48404e-001	2.95405e-004	-8.41861e-010
3	p2	1.00914e+000	3.49228e-002	6.26446e-005	-1.30438e-008

Para acceder al resultado:

TF1 *fpol = mg->**GetFunction**("pol2");

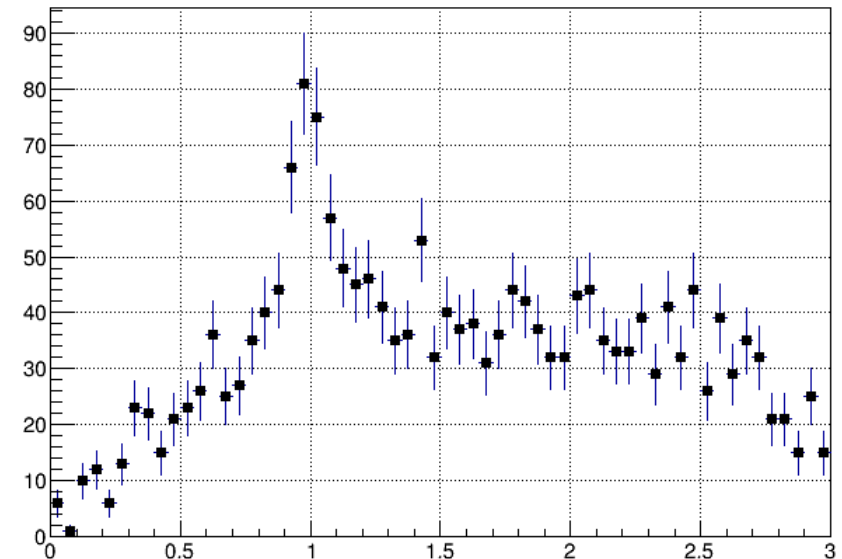
fpol->GetChisquare()/fpol->GetNDF() // =1.09

ndf = número de puntos – número de parámetros= 90 – 3 = 87

Ajuste para [señal y fondo](#)

Graficar Datos:

```
const int nBins = 60;
Double_t data[nBins] = { ... }
TH1F *histo = new TH1F("histo", "", 60, 0, 3);
for(int i=0; i < nBins; i++) histo-
>SetBinContent(i+1, data[i]);
histo->Draw("PE0");
```



Ajuste para [señal y fondo](#)

Definir funciones para fondo, señal, y total:

Fondo: función cuadrática:

```
Double_t background(Double_t *x, Double_t *par) {  
    return par[0] + par[1]*x[0] + par[2]*x[0]*x[0];  
}
```

Señal: función pico Lorentziano

```
Double_t lorentzianPeak(Double_t *x, Double_t *par) {  
    return (0.5*par[0]*par[1]/TMath::Pi()) / TMath::Max( 1.e-10,(x[0]-par[2])*(x[0]-par[2])  
    + .25*par[1]*par[1]);  
}
```

Total= Señal + Fondo

```
Double_t fitFunction(Double_t *x, Double_t *par) {  
    return background(x,par) + lorentzianPeak(x,&par[3]);  
}
```


Crear función en intervalo 0-3 con 6 parámetros:

```
TF1 *fitFcn = new TF1("fitFcn",fitFunction,0,3,6);
```

```
fitFcn->SetParameters(1,1,1,1,1,1);
```

```
histo->Fit("fitFcn","0");
```

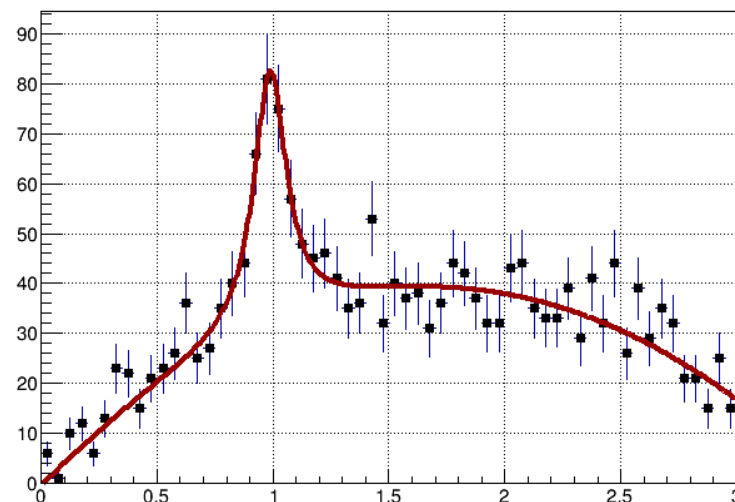
$$\chi^2_{\text{red}}=1.09$$

Al fijar parámetros mejoran errores:

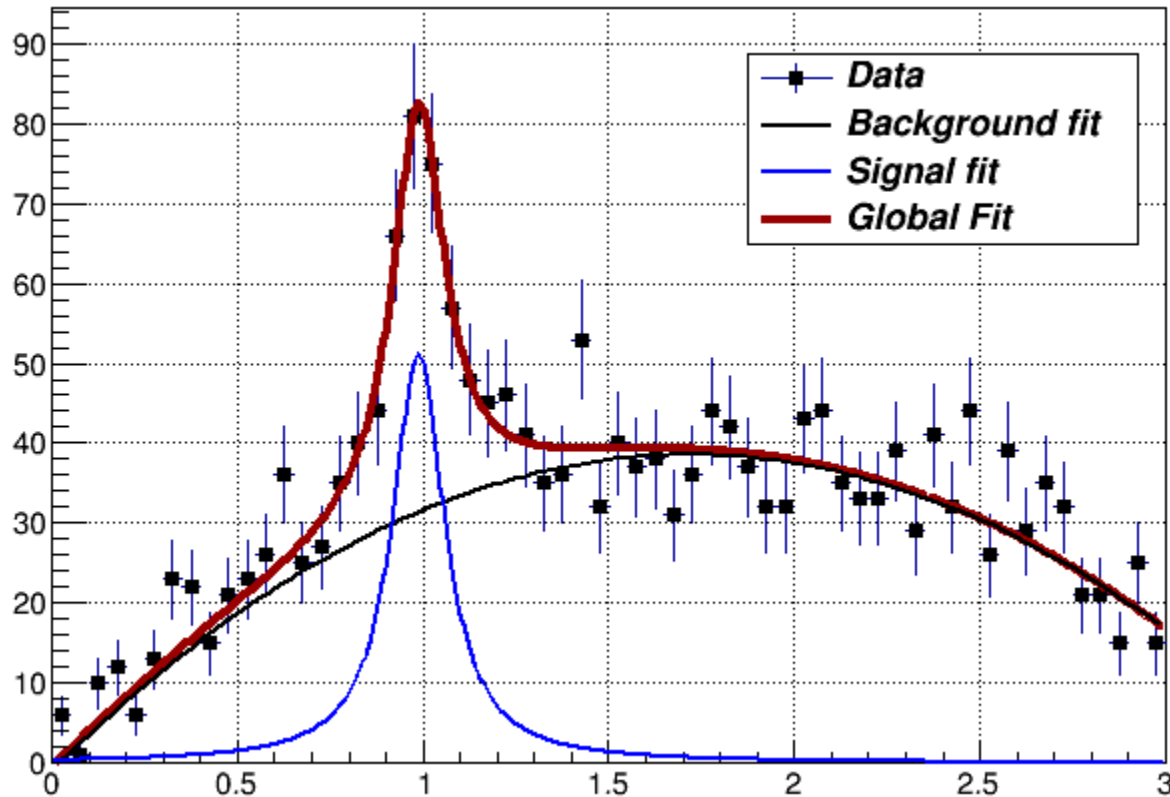
```
fitFcn->SetParameter(4,1.7); // width
```

```
fitFcn->SetParameter(5,1); // peak
```

```
histo->Fit("fitFcn","+","ep");
```



FCN=58.9284 FROM MIGRAD STATUS=CONVERGED 618 CALLS 619 TOTAL					
EDM=1.54329e-009 STRATEGY= 1 ERROR MATRIX UNCERTAINTY					
1.2 per cent					
EXT	PARAMETER				
NO.	NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	p0	-8.64715e-001	8.87889e-001	3.02210e-005	-3.15277e-006
2	p1	4.58434e+001	2.64076e+000	6.35729e-004	1.78463e-005
3	p2	-1.33214e+001	9.77307e-001	-1.31737e-004	3.73302e-005
4	p3	1.38074e+001	2.20785e+000	-1.29864e-003	-9.22424e-006
5	p4	1.72300e-001	3.72077e-002	-5.22394e-006	-1.45631e-003
6	p5	9.87281e-001	1.13098e-002	2.92804e-006	-3.44378e-004
FCN=58.9284 FROM MIGRAD STATUS=CONVERGED 224 CALLS 225 TOTAL					
EDM=4.83662e-010 STRATEGY= 1 ERROR MATRIX ACCURATE					
EXT	PARAMETER				
NO.	NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	p0	-8.64704e-001	8.91777e-001	2.08428e-003	-3.07864e-006
2	p1	4.58434e+001	2.64184e+000	1.52471e-003	-4.57181e-005
3	p2	-1.33214e+001	9.76813e-001	6.23584e-004	-1.05448e-004
4	p3	1.38074e+001	2.17653e+000	5.05350e-003	-7.36202e-006
5	p4	1.72309e-001	3.58106e-002	9.43011e-005	5.07442e-005
6	p5	9.87281e-001	1.12682e-002	4.24023e-005	1.26816e-003



```
TF1 *backFcn = new TF1("backFcn",background,0,3,3);
TF1 *signalFcn = new TF1("signalFcn",lorentzianPeak,0,3,3);
Double_t par[6];
fitFcn->GetParameters(par);
backFcn->SetParameters(par);
signalFcn->SetParameters(&par[3]);
```

TFractionFitter: Ajustar fracciones de histogramas de MC a histograma de datos

Toma en cuenta errores estadísticos tanto de los datos como del MC

- Usa un ajuste de likelihood con estadística de Poisson
 - Predicciones del MC (template) se varían dentro de la estadística con contribuciones adicionales a la likelihood.
- > más parámetros que ajustar: uno por bin por template, pero minimización es hecha analíticamente no siendo verdaderos parámetros que ajustar.

Suposiciones:

- Número total de eventos en cada template no es muy pequeño (se puede omitir su error poissoniano)
- Número de eventos en cada bin es mucho menor que el número total de eventos en cada template (errores multinomiales pueden ser reemplazados por poissonianos)

```
{
    TH1F *data;           //data histogram
    TH1F *mc0;            // first MC histogram
    TH1F *mc1;            // second MC histogram
    TH1F *mc2;            // third MC histogram
    ....                 // retrieve histograms
    TObjArray *mc = new TObjArray(3); // MC histograms are put in this array
    mc->Add(mc0);
    mc->Add(mc1);
    mc->Add(mc2);
    TFractionFitter* fit = new TFractionFitter(data, mc); // initialise
    fit->Constrain(1,0.0,1.0); // constrain fraction 1 to be between 0 and 1
    fit->SetRangeX(1,15); // use only the first 15 bins in the fit
    Int_t status = fit->Fit(); // perform the fit
    std::cout << "fit status: " << status << std::endl;
    if (status == 0) { // check on fit status
        TH1F* result = (TH1F*) fit->GetPlot();
        data->Draw("Ep");
        result->Draw("same");
    }
}
```

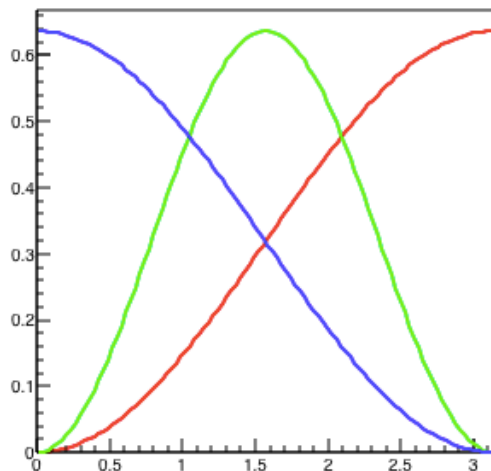
Histogramas con pesos del MC se pueden incluir:

fit->**SetWeight**(parameter #, pointer to weights histogram);

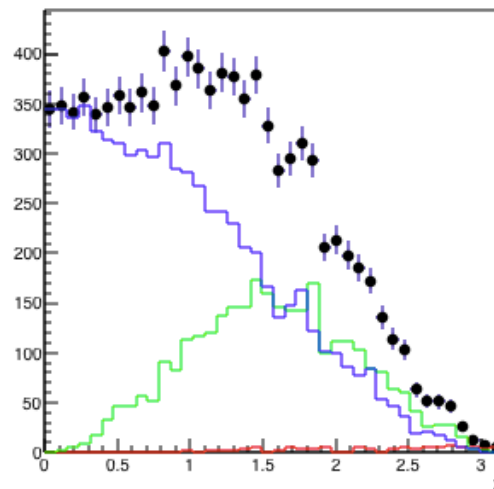
Acceder a resultados del ajuste:

fit->**GetResult**(parameter #, value, error);

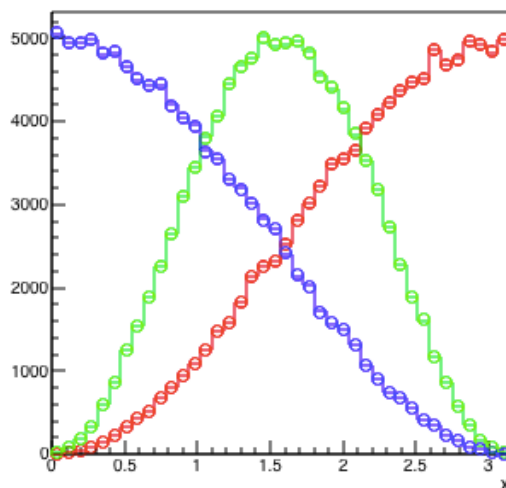
$[0] \cdot (1 - \cos(x)) / \text{TMath::Pi}()$



Data distribution with true contributions



MC generated samples with fit predictions



Data distribution with fitted contributions

