HANDS-ON SESSION

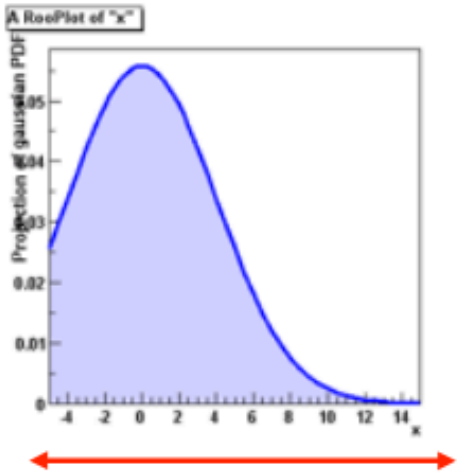
Introduction to ROOT/RooFit

Credits go to V.Werkerke, L.Lista & L.Moneta

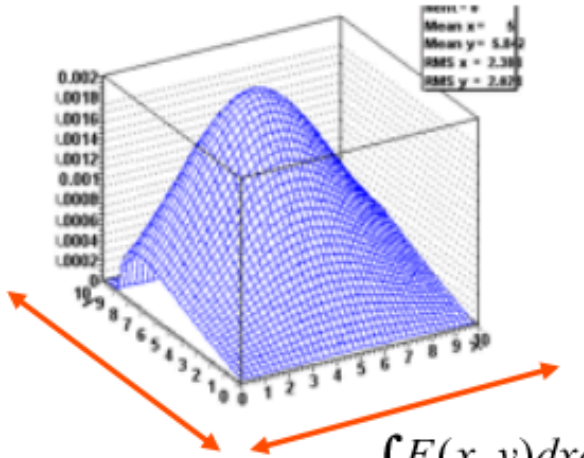
- Toolkit for data modeling
  - developed by *W. Verkerke and D. Kirkby*
- model distribution of observable x in terms of parameters p
  - probability density function (pdf): $\mathcal{P}(x;p)$
- pdf are normalized over allowed range of observables x with respect to the parameters p

- Probability Density Functions describe probabilities, thus
  - All values most be >0
  - The total probability must be 1 *for each p*, i.e.
  - Can have any number of dimensions

$$\int_{\vec{x}_{\min}}^{\vec{x}_{\max}} g(\vec{x}, \vec{p}) d\vec{x} \equiv 1$$



$$\int F(x)dx \equiv 1$$



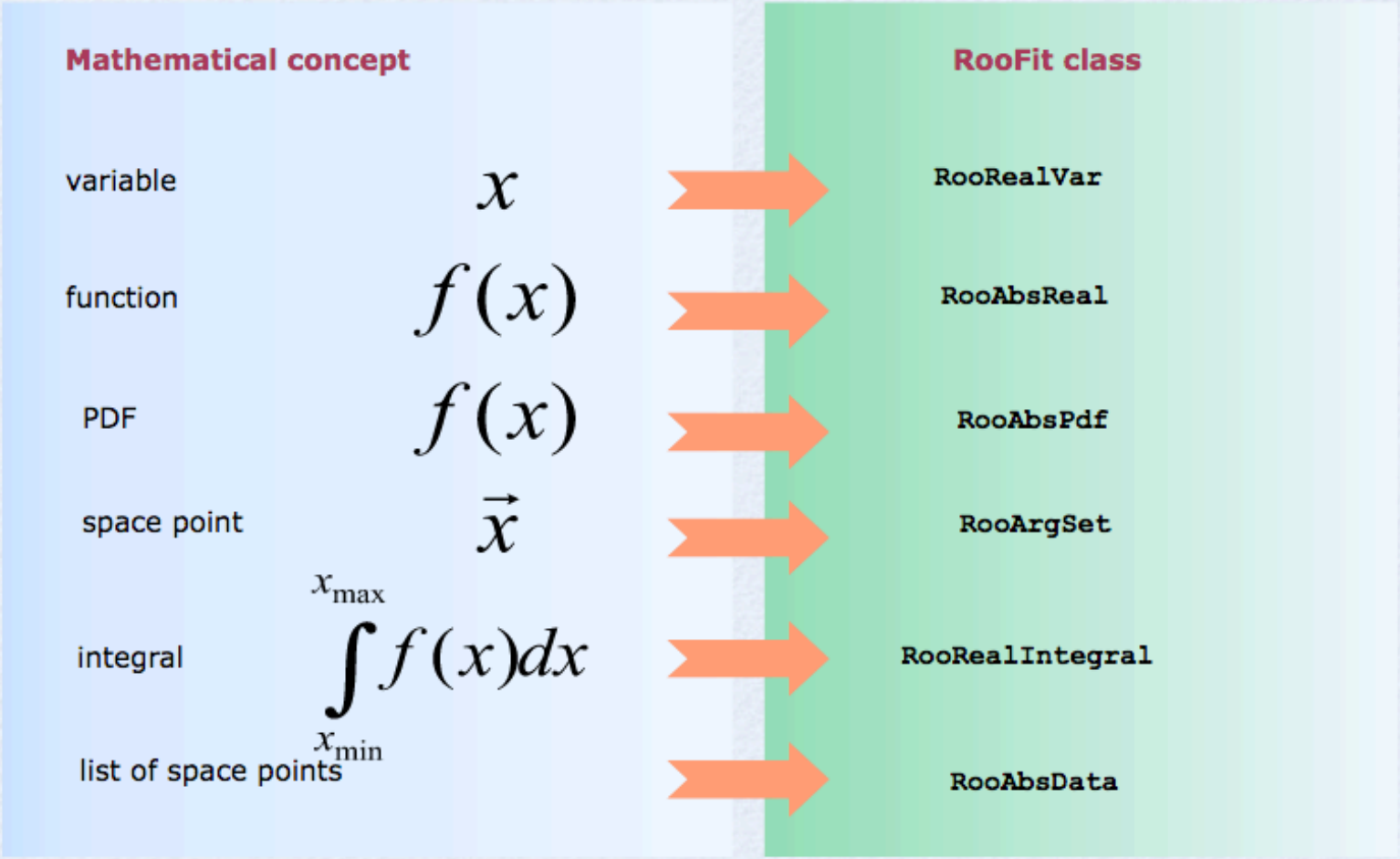
$$\int F(x,y)dxdy \equiv 1$$

- Note distinction in role between *parameters* (p) and *observables* (x)
  - Observables are measured quantities
  - Parameters are degrees of freedom in your model

- ROOT function framework can handle complicated functions
  - but require writing large amount of code
- Normalization of p.d.f. not always trivial
  - RooFit does it automatically
- In complex fit, computation performance important
  - need to optimize code for acceptable performance
  - built-in optimization available in RooFit
    - evaluation only when needed
- Simultaneous fit to different data samples
- Provide full description of model for further use

- RooFit provides functionality for building the pdf's
  - complex model building from standard components
  - composition with addition product and convolution
- All models provide the functionality for
  - maximum likelihood fitting
  - toy MC generator
  - visualization

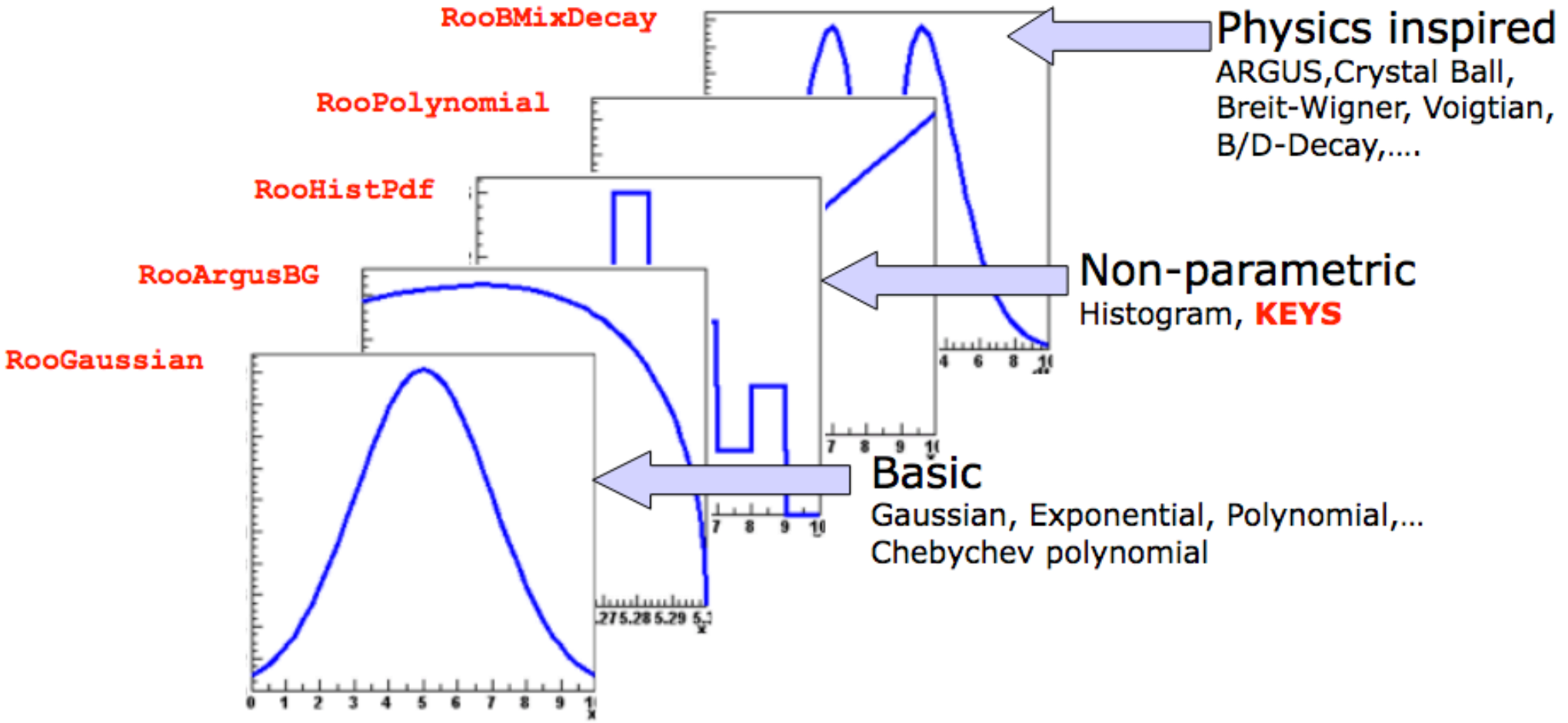
Mathematical concepts are represented as C++ objects



- List of most frequently used pdfs and their factory spec

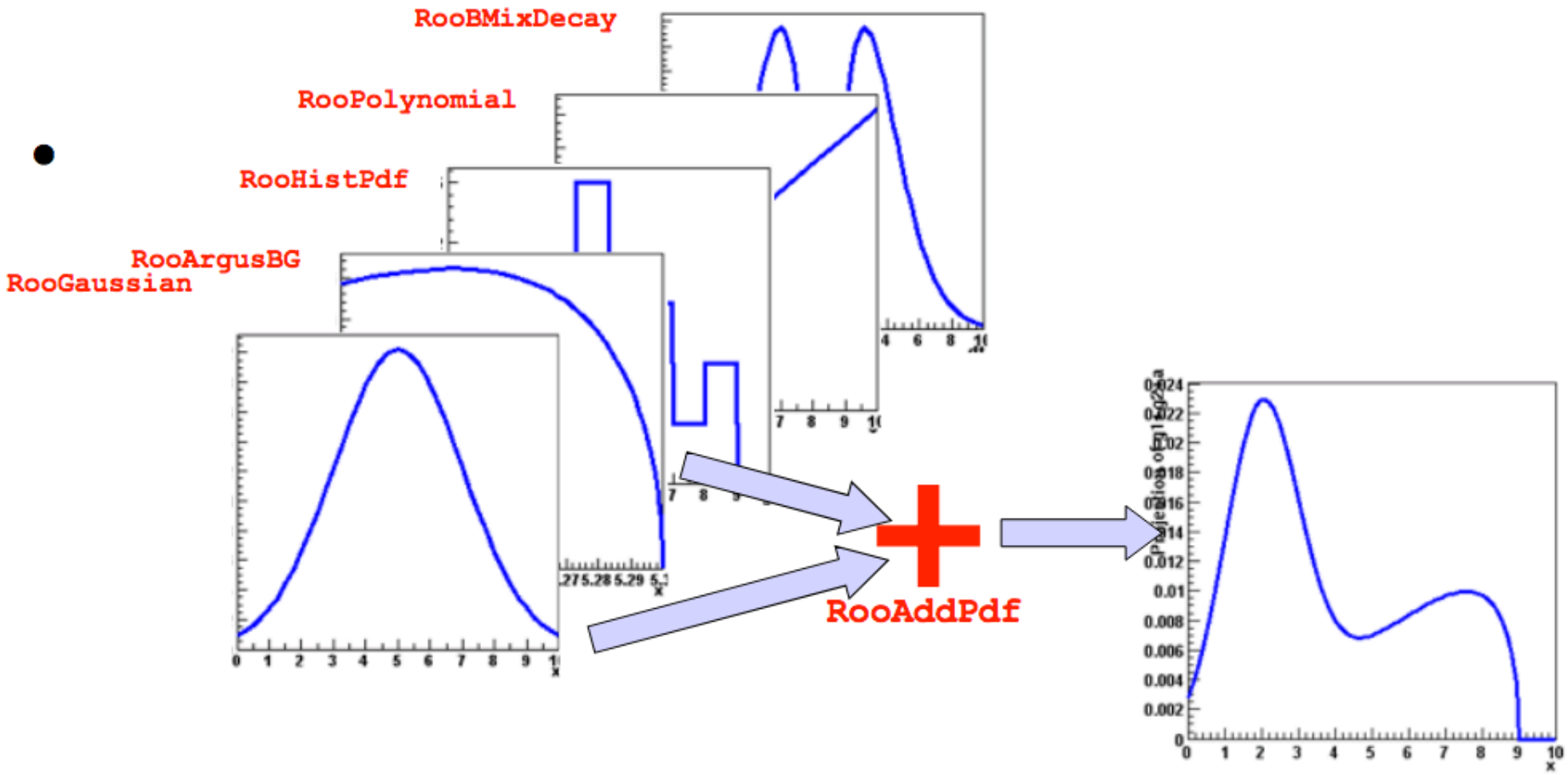
Gaussian	`Gaussian::g(x,mean,sigma)`
Breit-Wigner	`BreitWigner::bw(x,mean,gamma)`
Landau	`Landau::l(x,mean,sigma)`
Exponential	`Exponential::e(x,alpha)`
Polynomial	`Polynomial::p(x,{a0,a1,a2})`
Chebychev	`Chebychev::p(x,{a0,a1,a2})`
Kernel Estimation	`KeysPdf::k(x,dataSet)`
Poisson	`Poisson::p(x,mu)`
Voigtian (=BW⊗G)	`Voigtian::v(x,mean,gamma,sigma)`

- RooFit provides a collection of compiled standard PDF classes



Easy to extend the library: each p.d.f. is a separate C++ class

- Most realistic models are constructed as the sum of one or more p.d.f.s (e.g. signal and background)
- Facilitated through operator p.d.f **RooAddPdf**



Syntax for `RooAddPdf`

- Additions of PDF (using fractions)

```
SUM::name(frac1*PDF1,PDFN)
```

```
SUM::name(frac1*PDF1,frac2*PDF2,...,PDFN)
```

  - Note that last PDF does not have an associated fraction

$$F(x) = f \times S(x) + (1-f)B(x) \quad ; \quad N_{\text{exp}} = N$$

- PDF additions (using expected events instead of fractions)

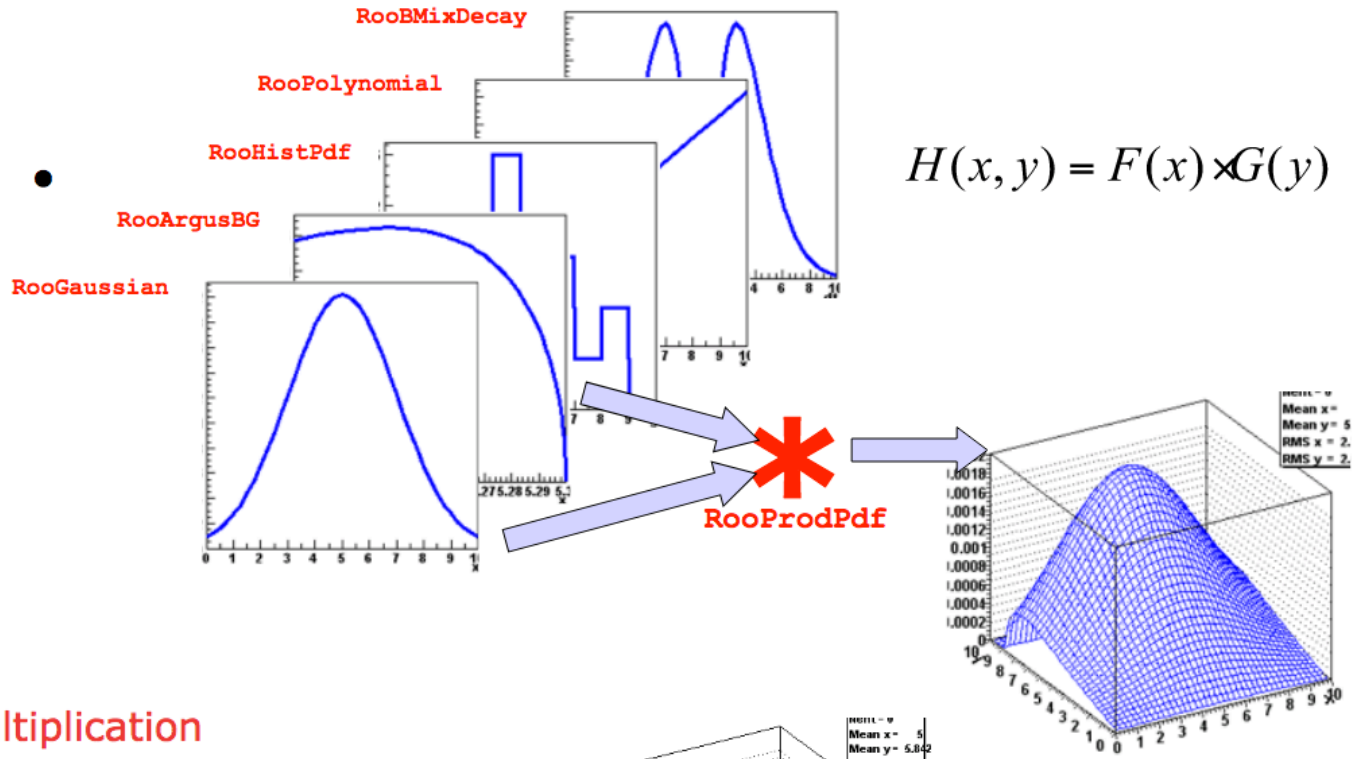
```
SUM::name(Nsig*SigPDF,Nbkg*BkgPDF)
```

$$F(x) = \frac{N_S}{N_S + N_B} \times S(x) + \frac{N_B}{N_S + N_B} B(x) \quad ; \quad N_{\text{exp}} = N_S + N_B$$

  - the resulting model will be extended
  - the likelihood will contain a Poisson term depending on the total number of expected events (Nsig+Nbkg)

L (x | p) -> L(x|p)Poisson(N_{obs},N_{exp})

RooFit model building : RooProdPdf (product of uncorrelated p.d.f.s)

- 

$$H(x,y) = F(x) \times G(y)$$

Uncorrelated products – Mathematics and constructors

- Mathematical construction of products of uncorrelated p.d.f.s is straightforward

2D

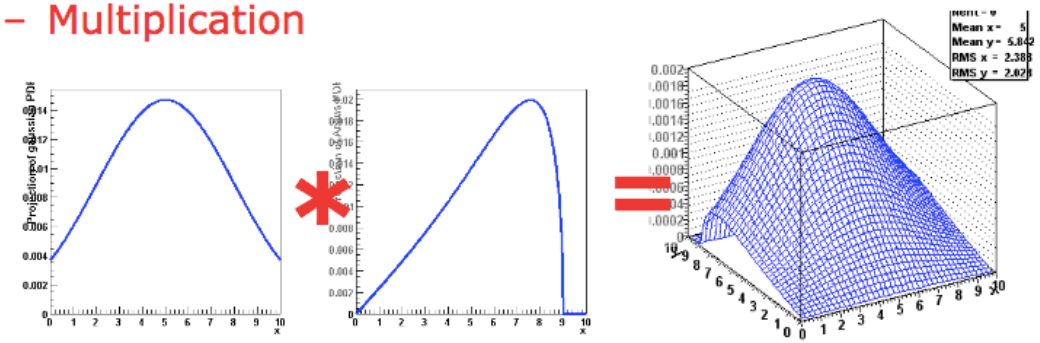
$$H(x,y) = F(x) \cdot G(y)$$

nD

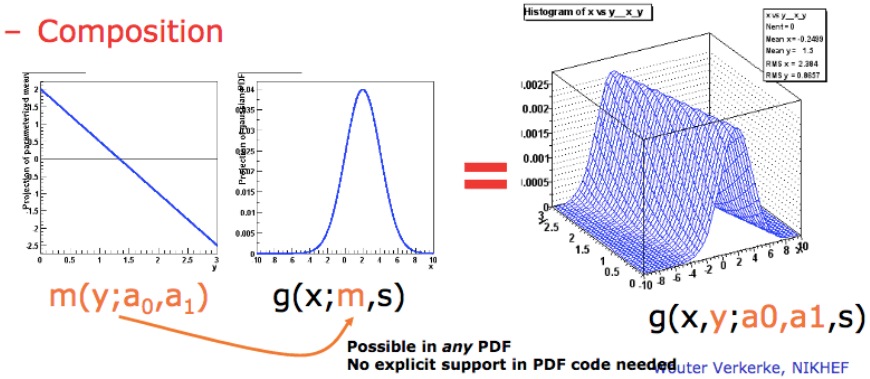
$$H(x^{\{i\}}) = \prod_i F^{\{i\}}(x^{\{i\}})$$

  - No explicit normalization required → If input p.d.f.s are unit normalized, product is also unit normalized
    (this is true *only* because of the absence of correlations)

– Multiplication



– Composition



Wouter Verkerke, NIKHEF

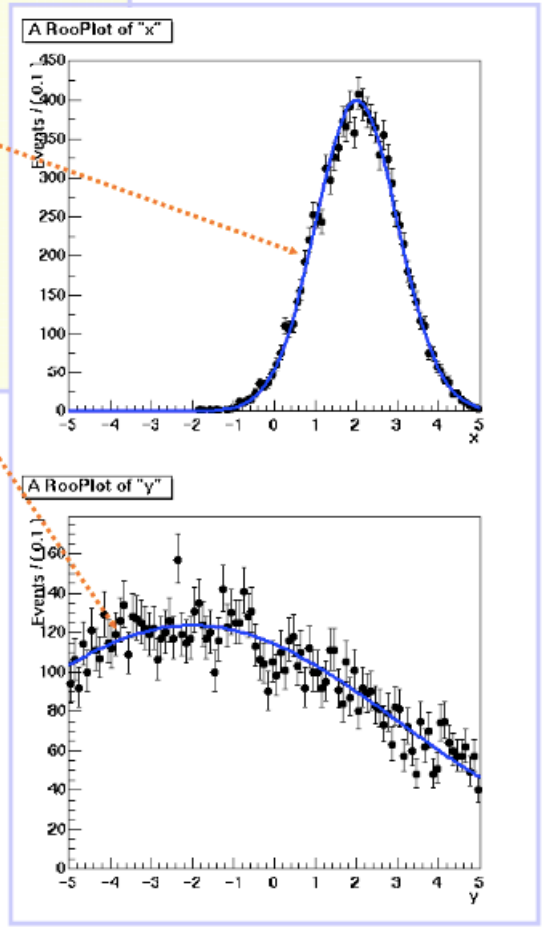
Plotting multi-dimensional PDFs

```
RooPlot* xframe = x.frame() ;
data->plotOn(xframe) ;
prod->plotOn(xframe) ;
xframe->Draw() ;

c->cd(2) ;
RooPlot* yframe = y.frame() ;
data->plotOn(yframe) ;
prod->plotOn(yframe) ;
yframe->Draw() ;
```

$$f(x) = \int pdf(x,y)dy$$

$$f(y) = \int pdf(x,y)dx$$



-Plotting a dataset D(x,y) versus x represents a *projection over y*

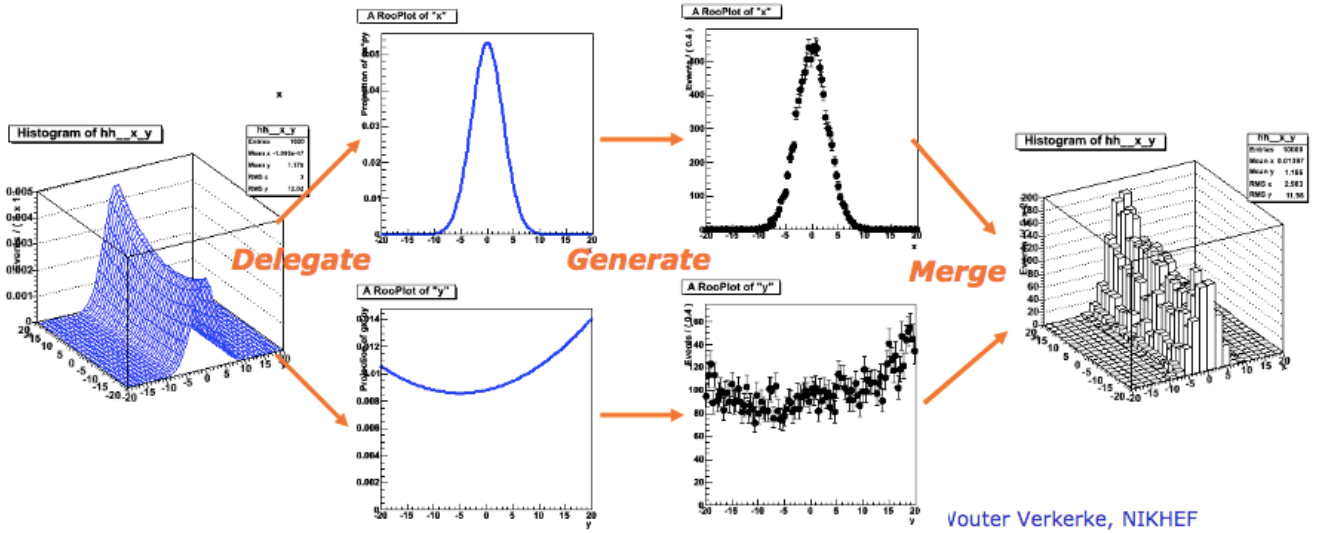
-To overlay PDF(x,y), you must plot *Int(dy)PDF(x,y)*

-RooFit automatically takes care of this!

- RooPlot remembers dimensions of plotted datasets

How it work – event generation on uncorrelated products

- If p.d.f.s are uncorrelated, each observable can be generated separately
  - Reduced dimensionality of problem (important for e.g. accept/reject sampling)
  - Actual event generation delegated to component p.d.f (can e.g. use internal generator if available)
  - **RooProdPdf** just aggregates output in single dataset



Wouter Verkerke, NIKHEF

Function minimization: minimization algorithm - I

- Common interface class (`ROOT::Math::Minimizer`)
- Existing implementations available as plug-ins:
  - **Minuit** (based on class `TMinuit`, direct translation from Fortran code)
    - with Migrad, Simplex, Minimize algorithms
  - **Minuit2** (new C++ implementation with OO design)
    - with Migrad, Simplex, Minimize and Fumili2
  - **Fumili** (only for least-square or log-likelihood minimizations)
  - **GSLMultiMin**: conjugate gradient minimization algorithm from GSL (Fletcher-Reeves, BFGS)
  - **GSLMultiFit**: Levenberg-Marquardt (for minimizing least square functions) from GSL
  - **Linear** for least square functions (direct solution, non-iterative method)
  - **GSLSimAn**: Simulated Annealing from GSL
  - **Genetic**: based on a genetic algorithm implemented in TMVA
- All these are available for ROOT fitting and in RooFit/RooStats
- Possible to combine them (e.g. use Minuit and Genetic)
- Easy to extend and add new implementations
  - e.g. minimizer based on NagC exists in the development branch (see here)

L.Moneta's slide

Function minimization: `Minuit` functionality - II

- MIGRAD
  - Find function minimum. Calculates function gradient, follow to (local) minimum, recalculate gradient, iterate until minimum found
    - To see what MIGRAD does, it is very instructive to do RooMinuit::setVerbose(1). It will print a line for each step through parameter space
  - Number of function calls required depends greatly on number of floating parameters, distance from function minimum and shape of function
- HESSE
  - Calculation of error matrix from 2nd derivatives at minimum
  - Gives symmetric error. Valid in assumption that likelihood is (locally parabolic)

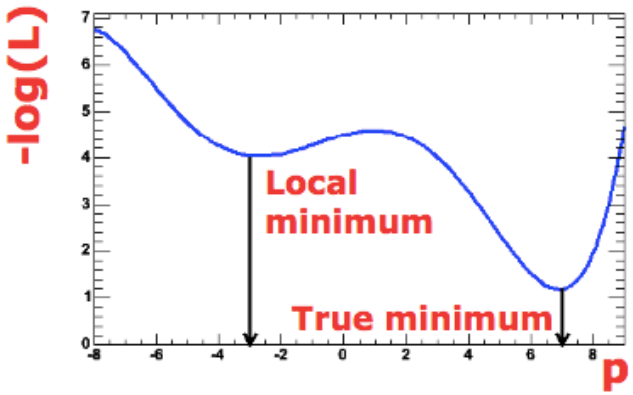
$$\hat{\sigma}(p)^2 = \hat{V}(p) = \left(\frac{d^2 \ln L}{d^2 p} \right)^{-1}$$

  - Requires roughly N^2 likelihood evaluations (with N = number of floating parameters)

- MINOS
  - Calculate errors by explicit finding points (or contour for >1D) where Δ-log(L)=0.5
  - Reported errors can be asymmetric
  - Can be very expensive in with large number of floating parameters
- CONTOUR
  - Find contours of equal Δ-log(L) in two parameters and draw corresponding shape
  - Mostly an interactive analysis tool

Function minimization: `Minuit` functionality - III

- For all but the most trivial scenarios it is not possible to automatically find reasonable starting values of parameters
  - So you need to supply 'reasonable' starting values for your parameters



Reason: There may exist multiple (local) minima in the likelihood or χ^2

  - You may also need to supply 'reasonable' initial step size in parameters. (A step size 10x the range of the above plot is clearly unhelpful)
  - Using RooMinuit, the initial step size is the value of `RooRealVar::getError()`, so you can control this by supplying initial error values

HANDS-ON SESSION 4

GETTING CONFIDENCE WITH … FUNCTION PLOTTING & HISTOGRAM HANDLING

Function minimization: Migrad - IV

- Purpose: find minimum

```
**********
**    13 **MIGRAD        1000           1
**********
 (some output omitted)
 MIGRAD MINIMIZATION HAS CONVERGED.
 MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
 COVARIANCE MATRIX CALCULATED SUCCESSFULLY
 FCN=257.304 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                     EDM=2.36773e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  mean         8.84225e-02   3.23862e-01   3.58344e-04  -2.24755e-02
   2  sigma        3.20763e+00   2.39540e-01   2.78628e-04  -5.34724e-02
                               ERR DEF= 0.5
 EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
  1.049e-01  3.338e-04
  3.338e-04  5.739e-02
 PARAMETER  CORRELATION COEFFICIENTS
       NO.   GLOBAL      1      2
        1  0.00430   1.000  0.004
        2  0.00430   0.004  1.000
```

Progress information, watch for errors here

Parameter values and approximate errors reported by MINUIT

Error definition (in this case 0.5 for a likelihood fit)

Function minimization: Migrad - V

- Purpose: find minimum

```
**********
**    13 **MIGR
**********
 (some output c
MIGRAD MINIMIZ
MIGRAD WILL VERI                    RIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                    EDM=2.36773e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  mean         8.84225e-02   3.23862e-01   3.58344e-04  -2.24755e-02
   2  sigma        3.20763e+00   2.39540e-01   2.78628e-04  -5.34724e-02
                                ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 1.049e-01  3.338e-04
 3.338e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
       NO.  GLOBAL      1      2
        1  0.00430   1.000  0.004
        2  0.00430   0.004  1.000
```

Value of $\chi 2$ or likelihood at minimum

(NB: χ^2 values are not divided by $N_{d.o.f}$)

Approximate Error matrix And covariance matrix

Function minimization: Migrad - VI

- Purpose: find minimu

```
**********
**    13 **MIGRAD        1000
**********
(some output omitted)
MIGRAD MINIMIZATION HAS CONVERGED
MIGRAD WILL VERIFY CONVERGENCE AND E     MATRIX.
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM MIGRAD    STATUS=CONVERGED      31 CALLS          32 TOTAL
                EDM=2.36773e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
  1  mean         8.84225e-02   3.23862e-01   3.58344e-04  -2.24755e-02
  2  sigma        3.20763e+00   2.39540e-01   2.78628e-04  -5.34724e-02
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 1.049e-01  3.338e-04
 3.338e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2
       1  0.00430   1.000  0.004
       2  0.00430   0.004  1.000
```

***Status*:**
Should be 'converged' but can be 'failed'

Estimated Distance to Minimum
should be small $O(10^{-6})$

Error Matrix Quality
should be 'accurate', but can be 'approximate' in case of trouble

- Purpose: calculate error matrix from $\frac{d^2L}{dp^2}$

```
**********
**    18 **HESSE        1000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM HESSE     STATUS=OK                          OTAL
                    EDM=2.36534e-06    STRAT                 CURATE
 EXT PARAMETER                                          INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR          STEP SIZE       VALUE
  1   mean         8.84225e-02   3.23861e-01   7.16689e-05   8.84237e-03
  2   sigma        3.20763e+00   2.39539e-01   5.57256e-05   3.26535e-01
                                ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 1.049e-01  2.780e-04
 2.780e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENTS
       NO.  GLOBAL      1      2
        1  0.00358   1.000  0.004
        2  0.00358   0.004  1.000
```

Symmetric errors calculated from 2nd derivative of $-\ln(L)$ or χ^2

Function minimization: Hesse - VIII

****
**
***
COV UCCESSFULLY
FCN TUS=OK 10 CALLS 42 TOTAL
de-06 STRATEGY= 1 ERROR MATRIX ACCURATE

EX NO			INTERNAL ERROR	INTERNAL STEP SIZE	INTERNAL VALUE
1			3.23861e-01	7.16689e-05	8.84237e-03
2	sig	3.20763e+00	2.39539e-01	5.57256e-05	3.26535e-01

ERR DEF= 0.5

EXTERNAL ERROR MATRIX. NDIM= 25 NPAR= 2 ERR DEF=0.5

1.049e-01	2.780e-04	
2.780e-04	5.739e-02	

PARAMETER CORRELATION COEFFICIENTS

NO.	GLOBAL	1	2
1	0.00358	1.000	0.004
2	0.00358	0.004	1.000

Error matrix (Covariance Matrix) calculated from

$$V_{ij} = \left(\frac{d^2(-\ln L)}{dp_i dp_j} \right)^{-1}$$

Function minimization: Hesse - IX

```
**********
**    18 **HESSE         1000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM HESSE     STATUS=OK              10 CALLS          42 TOTAL
                    EDM=2.36534e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR          STEP SIZE       VALUE
  1   mean         8.84225e-02                                  8.84237e-03
  2   sigma        3.20763e+00                                  3.26535e-01

EXTERNAL ERROR MATRIX.     NDIM=
 1.049e-01  2.780e-04
 2.780e-04  5.739e-02
PARAMETER  CORRELATION COEFFICIENT
      NO.  GLOBAL      1      2
       1  0.00358   1.000  0.004
       2  0.00358   0.004  1.000
```

F=0.5

Correlation matrix ρ_{ij} calculated from

$$V_{ij} = \sigma_i \sigma_j \rho_{ij}$$

Function minimization: Hesse - X

```
**********
**    18 **HESSE         1000
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=257.304 FROM HESSE     STATUS=OK              10 CALLS          42 TOTAL
                    EDM=2.36534e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR          STEP SIZE       VALUE
  1   mean                                      7.16689e-05   8.84237e-03
  2   sigma                                     5.57256e-05   3.26535e-01

EXTERNAL ERROR                                  2    ERR DEF=0.5
 1.049e-01  2.
 2.780e-04  5.739e-
PARAMETER  CORRELA   N COEFFICIENTS
       NO.   GLOBAL      1      2
        1  0.00358   1.000  0.004
        2  0.00358   0.004  1.000
```

Global correlation vector: correlation of each parameter with *all other* parameters

Function minimization: Minos - XI

```
**********
**    23 **MINOS         1000
**********
FCN=257.304 FROM MINOS     STATUS=SUCCESSFUL     52 CALLS          94 TOTAL
                     EDM=2.36534e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                PARABOLIC        MINOS ERRORS
 NO.   NAME      VALUE                          ERROR       NEGATIVE       POSITIVE
  1   mean         8.84225e-02   3.23861e-01  -3.24688e-01   3.25391e-01
  2   sigma        3.20763e+00   2.39539e-01  -2.23321e-01   2.58893e-01
                                ERR DEF= 0.5
```

Symmetric error

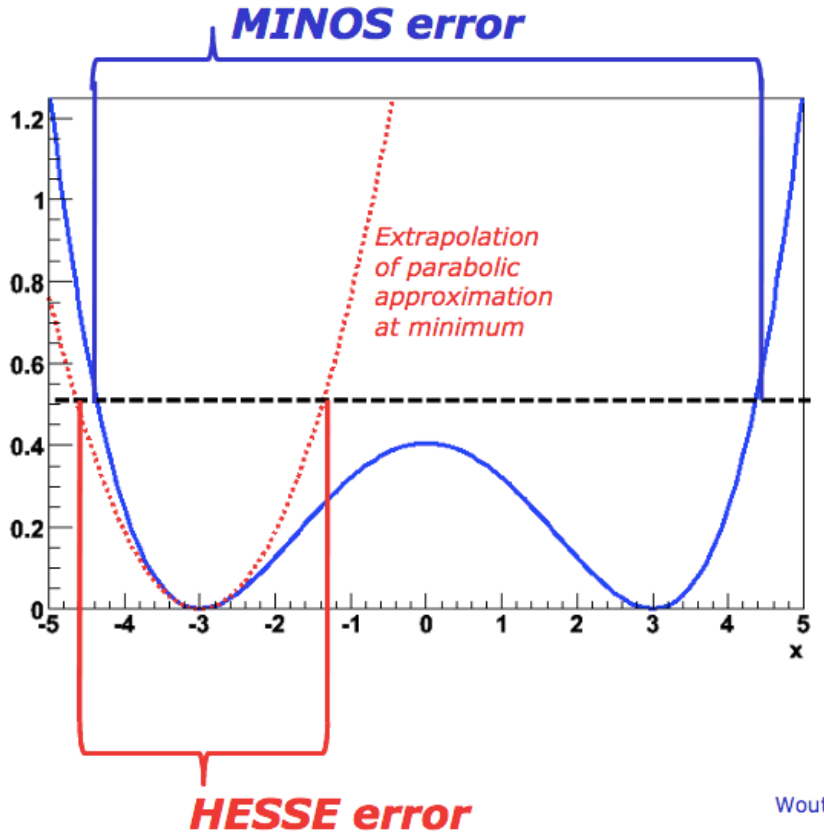
(repeated result from HESSE)

MINOS error
Can be asymmetric

(in this example the 'sigma' error is slightly asymmetric)

Illustration of difference between HESSE and MINOS errors

- 'Pathological' example likelihood with multiple minima and non-parabolic behavior



Wouter Verkerke, NIKHEF

Practical estimation – Fit converge problems

- Sometimes fits don't converge because, e.g.
  - MIGRAD unable to find minimum
  - HESSE finds negative second derivatives (which would imply negative errors)
- Reason is usually numerical precision and stability problems, but
  - The underlying cause of fit stability problems is usually by **highly correlated parameters** in fit
- HESSE correlation matrix in primary investigative tool

PARAMETER NO.	CORRELATION COEFFICIENTS GLOBAL	1	2
1	0.99835	1.000	0.998
2	0.99835	0.998	1.000

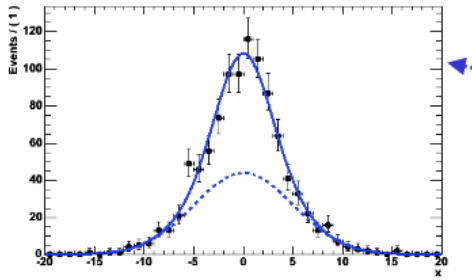
Signs of trouble...

  - In limit of 100% correlation, the usual point solution becomes a line solution (or surface solution) in parameter space. Minimization problem is no longer well defined

Function minimization: mitigating fit stability issues - XIII

- Strategy I – More orthogonal choice of parameters
  - Example: fitting sum of 2 Gaussians of similar width

$$F(x; f, m, s_1, s_2) = fG_1(x; s_1, m) + (1-f)G_2(x; s_2, m)$$



HESSE correlation matrix

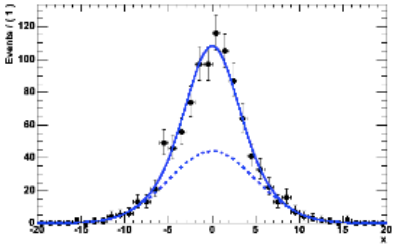
PARAMETER	CORRELATION COEFFICIENTS				
NO.	GLOBAL	[f]	[m]	[s1]	[s2]
[f]	0.96973	1.000	-0.135	0.918	0.915
[m]	0.14407	-0.135	1.000	-0.144	-0.114
[s1]	0.92762	0.918	-0.144	1.000	0.786
[s2]	0.92486	0.915	-0.114	0.786	1.000

Widths s_1,s_2 strongly correlated fraction f

  - Different parameterization:

$$fG_1(x; s_1, m_1) + (1-f)G_2(x; s_1 \cdot s_2, m_2)$$

PARAMETER	CORRELATION COEFFICIENTS				
NO.	GLOBAL	[f]	[m]	[s1]	[s2]
[f]	0.96951	1.000	-0.134	0.917	-0.681
[m]	0.14312	-0.134	1.000	-0.143	0.127
[s1]	0.98879	0.917	-0.143	1.000	-0.895
[s2]	0.96156	-0.681	0.127	-0.895	1.000



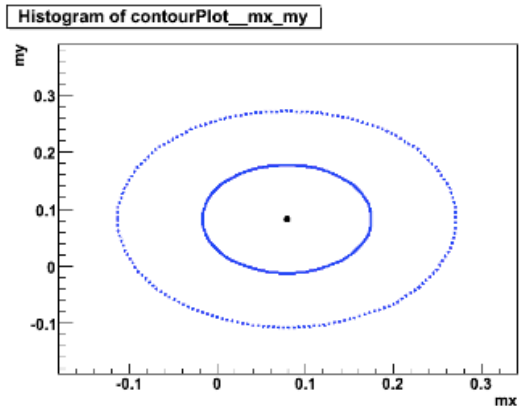
  - Correlation of width s2 and fraction f reduced from 0.92 to 0.68
  - Choice of parameterization matters!

- Strategy II – Fix all but one of the correlated parameters
  - If floating parameters are highly correlated, some of them may be redundant and not contribute to additional degrees of freedom in your model

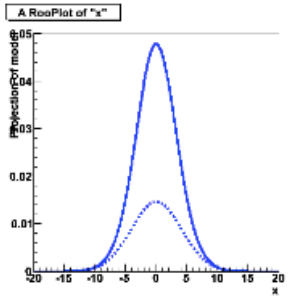
- Sometimes is it desirable to bound the allowed range of parameters in a fit
  - Example: a fraction parameter is only defined in the range [0,1]

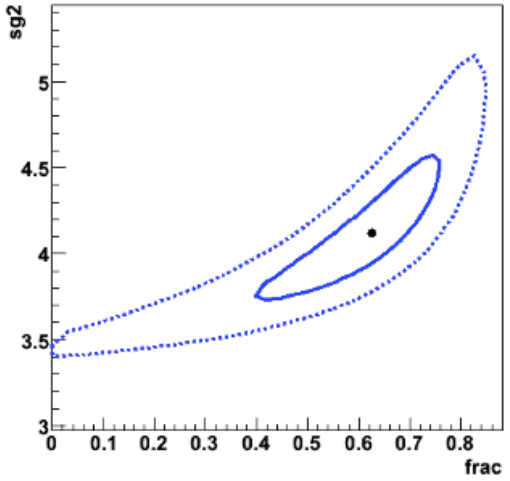
Minuit CONTOUR tool also useful to examine 'bad' correlations

- Example of 1,2 sigma contour of two uncorrelated variables
  - Elliptical shape. In this example parameters are uncorrelation



- Example of 1,2 sigma contour of two variables with problematic correlation
  - Pdf = f·G1(x,0,3)+(1-f)·G2(x,0,s) with s=4 in data





HANDS-ON SESSION-4

GETTING CONFIDENCE WITH … FITTING

Input file: `psiprime_bin9_histo.root`

(extracted from a larger ROOT file `hlt_5_newSoftMuon_alsoInPsiPrimeWind.root`)

First version of the code to run: `psiPrime_fit.C`
(it implements two subsequent fits with two different models)

1st model used: Gaussian for the Signal, Chebyshev polynomial for the background

2nd model used: Crystal-Ball function for the Signal, Exponential for the background

Let's introduce the Chebyshev polynomials and the Crystal-Ball function in the next slides.

Chebyshev polynomials - I

When using in the fit model a standard polynomial parametrization (`RooPolynomial`, https://root.cern.ch/doc/master/classRooPolynomial.html)

RooPolynomial implements a polynomial p.d.f of the form.

$$f(x) = \mathcal{N} \cdot \sum_i a_i * x^i$$

By default, the coefficient a_0 is chosen to be 1, as polynomial probability density functions have one degree of freedom less than polynomial functions due to the normalisation condition. $\mathcal{N}$ is a normalisation constant that is automatically calculated when the polynomial is used in computations.

… very often it results in strong correlations between coefficients that introduce - **issues in the fit stability**
- inability to find the right solution at high order

This can be solved (i.e. mitigating fit instability) using better polynomial parametrization, such as Chebyshev or Bernstein polynomials.

Let's discuss the first ones now.

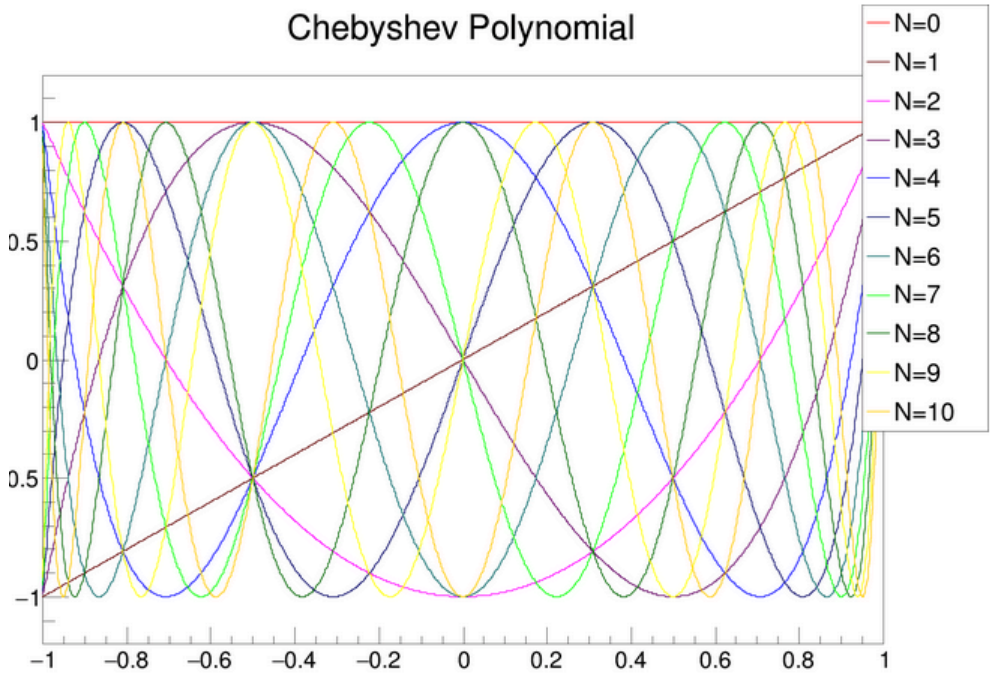
Chebyshev polynomials - II

When using the `RooChebyshev` class (https://root.cern.ch/doc/master/classRooChebychev.html) consider that ...

... the number of parameters corresponds to the degree of the considered polynomial !

The coefficient that goes with $T_0(x) = 1$ (i.e. the constant polynomial) is implicitly assumed to be 1, and the list of coefficients supplied by callers starts with the coefficient that goes with $T_1(x) = x$ (i.e. the linear term).

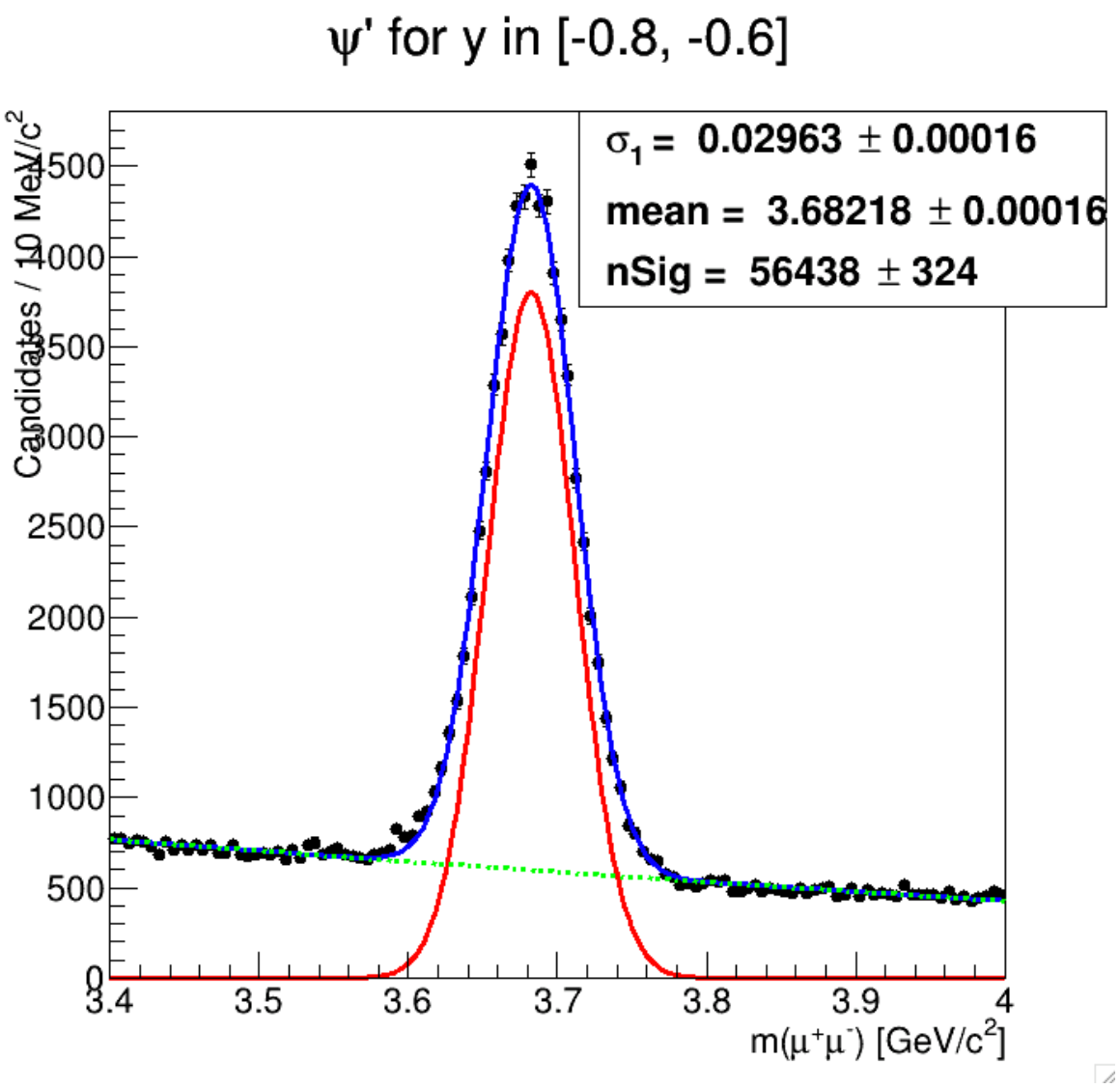
$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$
$$T_5(x) = 16x^5 - 20x^3 + 5x$$
$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$$
$$T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x$$
$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$
$$T_9(x) = 256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x$$



These polynomials are typically extensively exploited in numerical approximation tasks

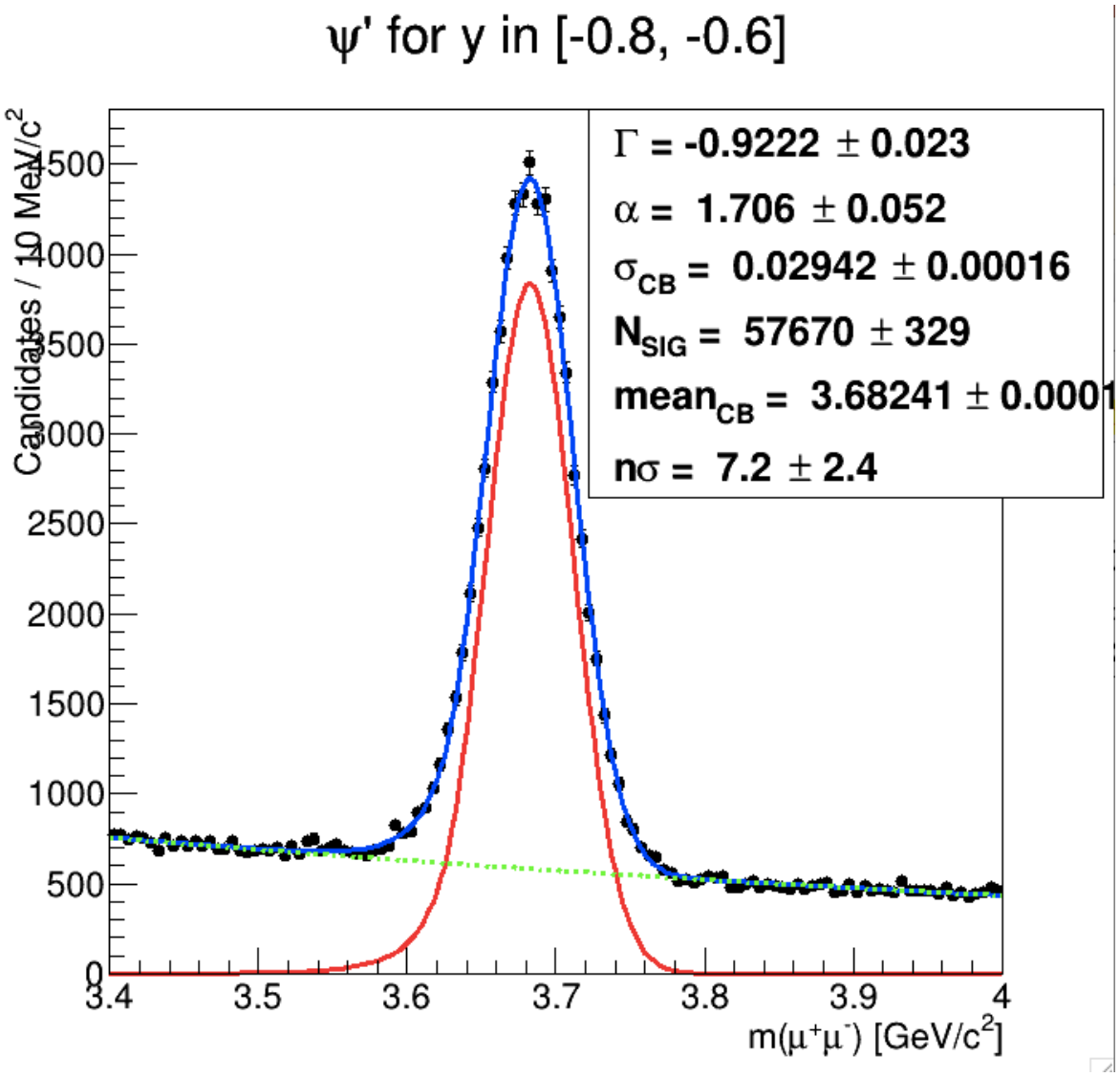
Visualize 1st fit:

```
> display PsiPrimeMassFit_gauss_poly1ord.png
```



Visualize 2nd fit:

```
> display PsiPrimeMassFit_gauss_poly1ord.png
```



Note that the fit differs especially at the lower-mass shoulder because of the tail of the Crystal-Ball function that can well adapt this shoulder: