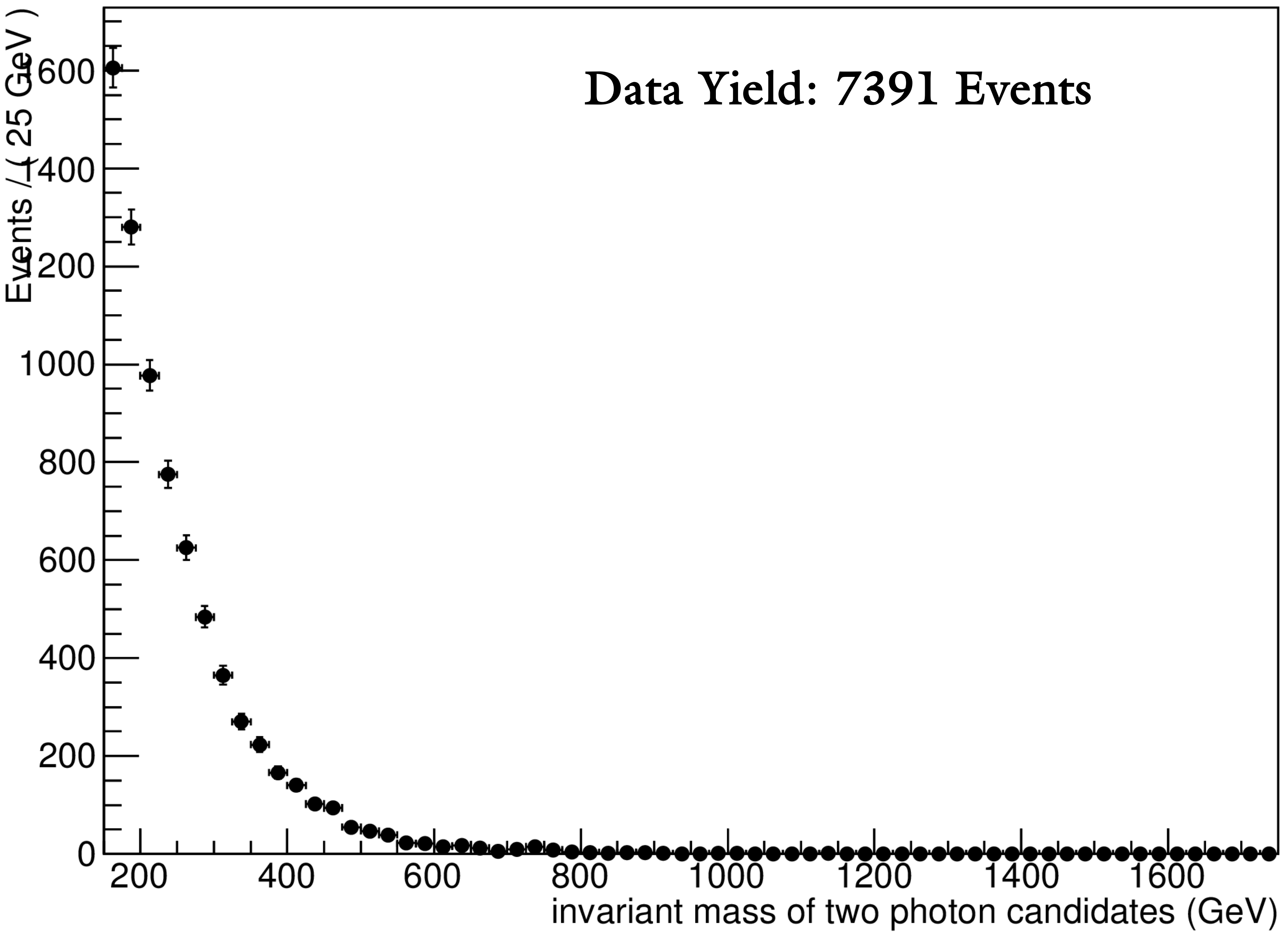
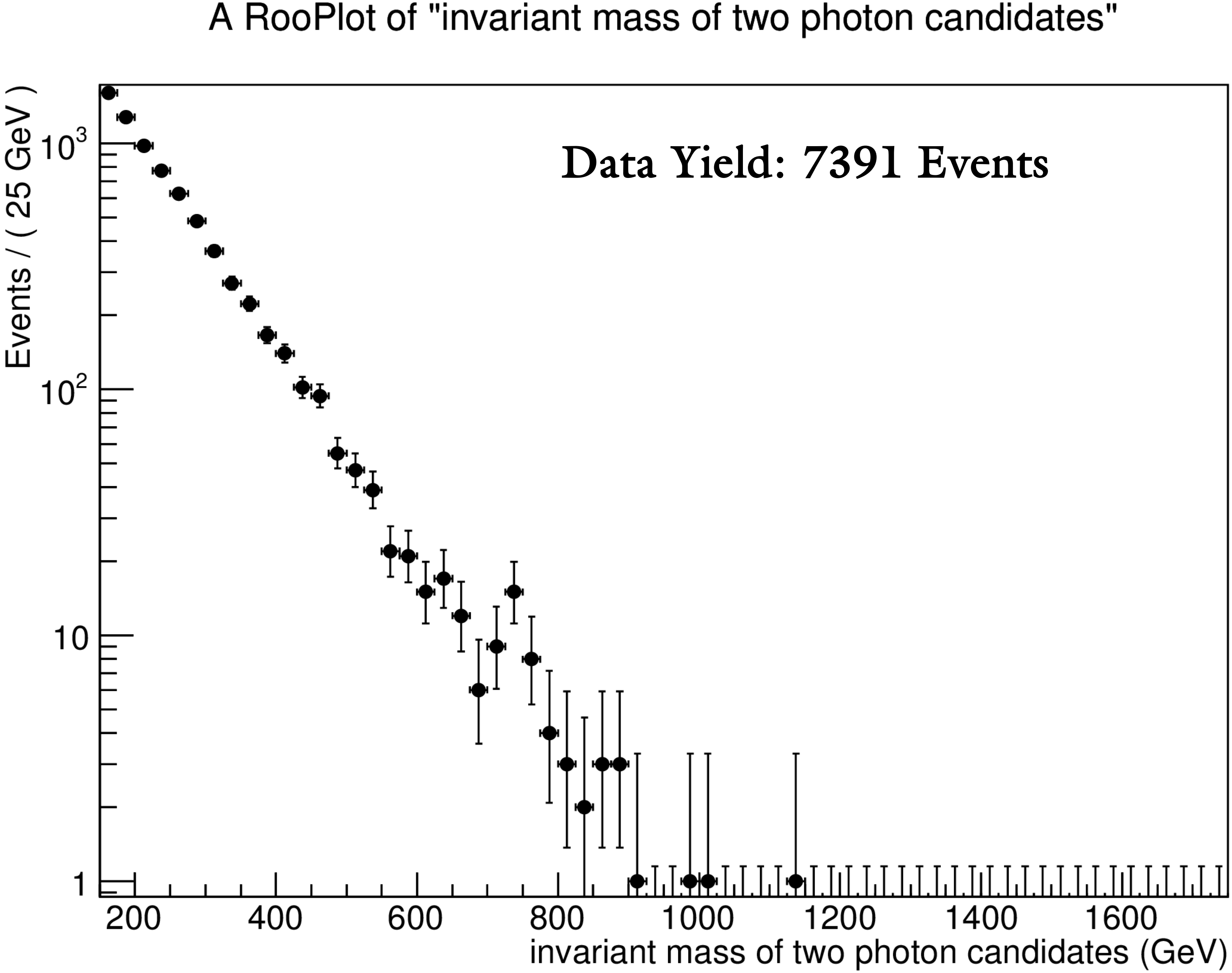
Tools and Approaches for Statistical Analysis of Data

Stephen Sekula
SMU
Presented in PHYS 7361, Spring 2016
4/14/16

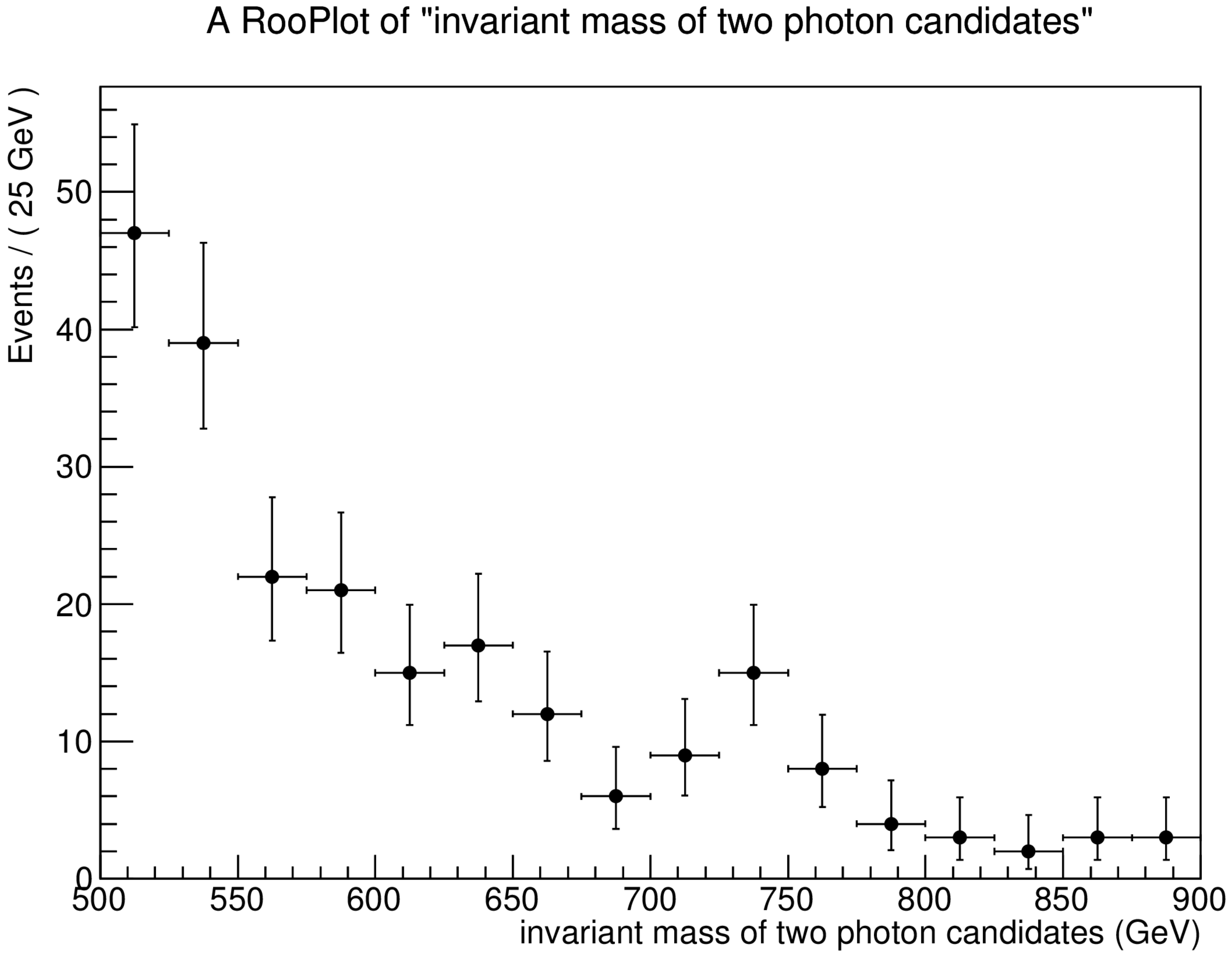
(NOTE: today, we are all Frequentists)

A RooPlot of "invariant mass of two photon candidates"





Data Yield: 7391 Events


A RooPlot of "invariant mass of two photon candidates"
Events / (25 GeV)
0
10
20
30
40
50
500
550
600
650
700
750
800
850
900
invariant mass of two photon candidates (GeV)

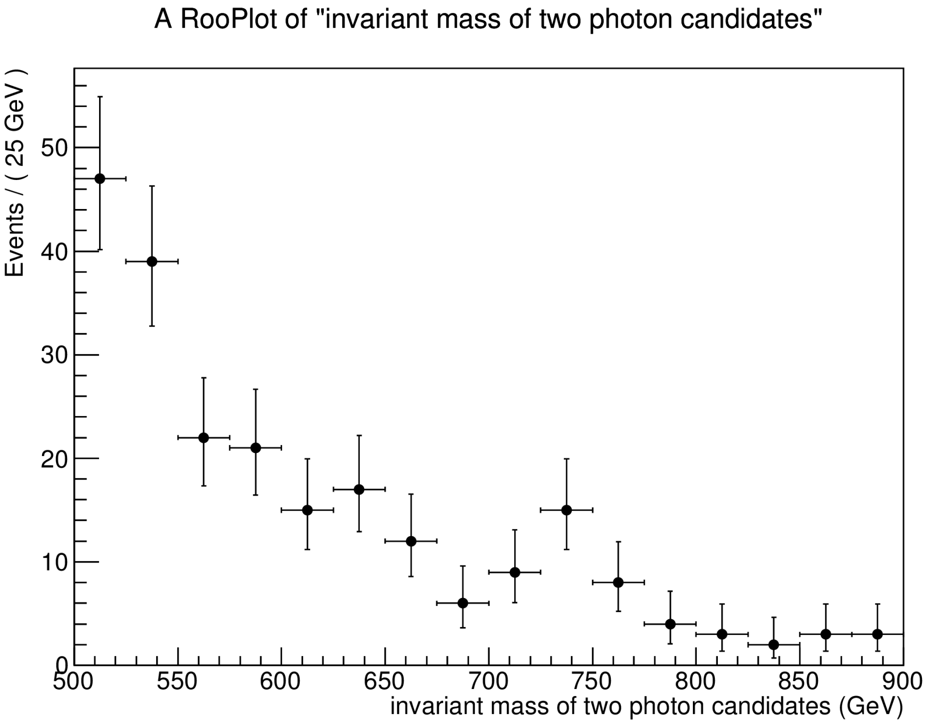
We have 75 minutes…

- The ATLAS Higgs and Exotics Group Conveners want to have a meeting in 75 minutes to determine how to proceed with this search
- In this meeting of the Exotics Diphoton Statistics Subgroup, we need to try to quantify the significance of this "bump"…
  - Should we be excited about this or not? Are we convinced that this is worth being excited about? What message do we convey to the Conveners about this bump so they can make decisions about how to proceed?

Setup on ManeFrame

```
ssh mflogin01.hpc.smu.edu
(alternatively, use mflogin02)

source /grid/software/ATLASLocalRootBase/setup.sh

mkdir PHYS7361

cd PHYS7361

cp /users/sekula/PHYS7361/diphoton.dat .
```

First Approach: Simple "Cut-and-Count" estimation



Our Goal:

- A "quick estimate" of the compatibility of the "bump" region with the background in that region

We have to "eyeball" the background – we do not yet have a model of the expected background in this region.

Suggestion: estimate the average background/bin to the left of the peak (use 3-4 bins to the left), the average background/bin to the right of the peak (again, use 3-4 bins), and then average them. Use that as the per-bin background estimate in the peak region in between.

Import and Visualize the Data

```
#include “RooGlobalFunc.h”
#include “RooRealVar.h”
#include “RooDataSet.h”
#include “RooPlot.h”
#include “TCanvas.h”

void PlotData() {

  RooRealVar  mass(“mass”,”m_{#gamma#gamma}”,150,1750,”GeV”);
  RooDataSet* data = RooDataSet::read(“diphoton.dat”,
                                      RooArgSet(mass));
  std::cout << data->numEntries() << std::endl;

  RooPlot* plot = mass.frame();
  data->plotOn(plot, RooFit::Binning(64));

  TCanvas c1(“c1”,””,800,600);
  c1.cd();
  c1.SetLogy(kTRUE);

  plot->Draw();

  c1.SaveAs(“PlotData_diphotonmass.pdf”);
}
```

Import and Visualize the Data

```
root -q -l -b ./PlotData.C'+()'
```

```
Processing .PlotData.C+()...
Info in <TUnixSystem::ACLiC>: creating shared library /users/sekula/PHYS7361/./PlotData_C.so

RooFit v3.60 -- Developed by Wouter Verkerke and David Kirkby
                Copyright (C) 2000-2013 NIKHEF, University of California & Stanford University
                All rights reserved, please read http://roofit.sourceforge.net/license.txt

[#1] INFO:DataHandling -- RooDataSet::read: reading file diphoton.dat
[#0] ERROR:DataHandling -- RooDataSet::read(static): read error at line 7392
[#1] INFO:DataHandling -- RooDataSet::read: read 7391 events (ignored 0 out of range events)
7391
[#0] ERROR:InputArguments -- RooAbsRealLValue::frame(mass) ERROR: unrecognized command:
BinningSpec
Info in <TCanvas::Print>: pdf file PlotData_diphotonmass.pdf has been created
```

To be a good citizen on ManeFrame, run your code on a worker node instead of locally on the login node:

```
srun -p serial root -q -l -b ./PlotData.C'+()'
```

(the above sends the process to a worker node in the “serial” partition of worker machines)

Select subsets of the Data

```
// Select a subset of the data and draw it with 25 GeV bins

c1.SetLogy(kFALSE);

RooDataSet* peak_region =
 (RooDataSet*) data->reduce(
 RooFit::Cut("500 < mass && mass < 900"));

plot = mass.frame(RooFit::Range(500.0,900.0));

peak_region->plotOn(plot,RooFit::Binning(64));

plot->Draw();

c1.SaveAs("PlotData_diphotonmass_500_900.pdf”);
```

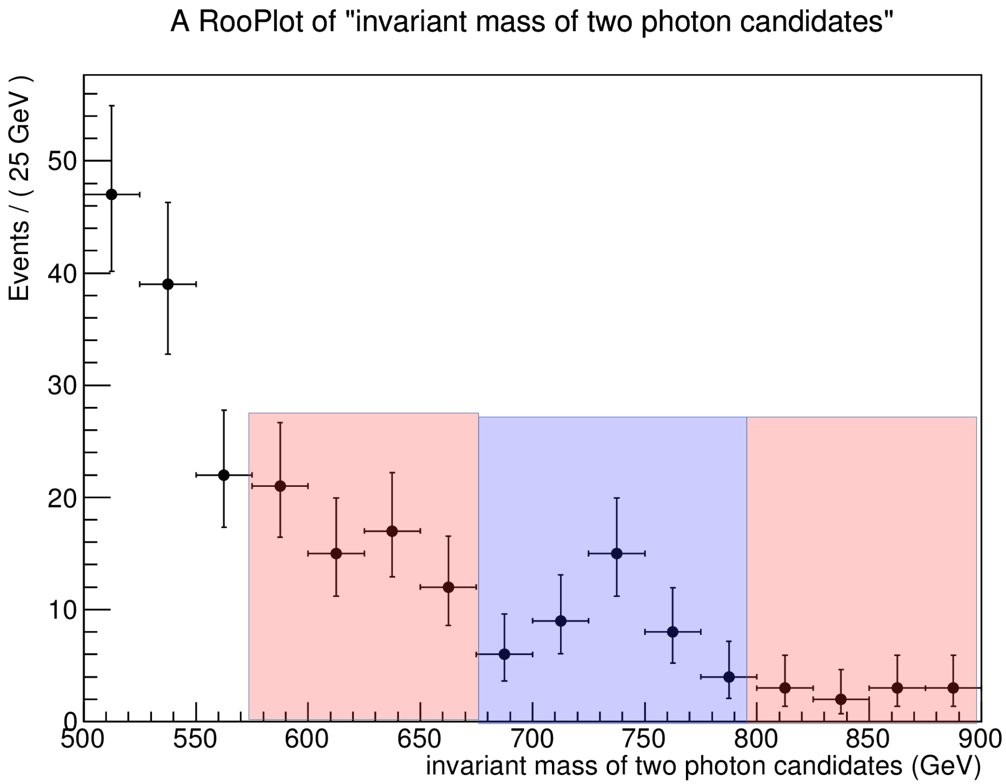
Cut-and-Count Code

```
void CutAndCount() {

  RooRealVar  mass("mass","m_{#gamma#gamma}",150,1750,"GeV");
  RooDataSet* data = RooDataSet::read("diphoton.dat", RooArgSet(mass));
  TH1F* data_hist = (TH1F*) data->createHistogram("data_hist", mass,
RooFit::Binning(64));
  Int_t low_min_bin  = data_hist->FindBin(575.);
  Int_t high_min_bin = data_hist->FindBin(800.);
  Double_t bkg_low  = data_hist->Integral(low_min_bin,  low_min_bin+3);
  Double_t bkg_high = data_hist->Integral(high_min_bin, high_min_bin+3);
  Double_t avg_background_low  = bkg_low/4.0;
  Double_t avg_background_high = bkg_high/4.0;
  Double_t avg_background_low_err  = TMath::Sqrt(bkg_low)/4.0;
  Double_t avg_background_high_err = TMath::Sqrt(bkg_high)/4.0;
  Double_t avg_background = (avg_background_low+avg_background_high)/2.0;
  Double_t avg_background_err =
TMath::Sqrt(avg_background_low_err*avg_background_low_err +
avg_background_high_err*avg_background_high_err)/2.0;
  Double_t bkg_in_peak_region = avg_background*5.0;
  Double_t bkg_in_peak_region_err = avg_background_err*5.0;
  Double_t yield_in_peak_region = data_hist->Integral(low_min_bin+4,
high_min_bin-1);
  Double_t signal_yield = yield_in_peak_region - bkg_in_peak_region;
  Double_t signal_yield_err = TMath::Sqrt( yield_in_peak_region +
bkg_in_peak_region_err*bkg_in_peak_region_err);
}
```

See ~sekula/PHYS7361/CutAndCount.C

Cut-and-Count Estimation of Signal, Background, and Significance

- I used 4 bins to the left
  - Found average of 16.25 events/bin
- I used 4 bins to the right
  - Found average of 2.75 events/bin
- Average background/bin in the "peak region"
  - 9.5 +/- 1.1 events/bin
- There are 5 bins in the peak region I selected
  - Number of events, N = 42
  - Estimated background, B = 47.5 +/- 5.4
  - N – B = estimated signal, S = -5.5 +/- 8.5
  - S/σ = -0.64 → significance above background expectation of -0.64 standard deviations


A RooPlot of "invariant mass of two photon candidates"

Comment

- That was a pretty “conservative” approach to estimating the background
  - The higher number in the average carries the bigger weight
  - The assumption was that this average was to be applied in a flat manner through the bump region
  - These are pretty strong and likely incorrect assumptions. We can see by eye that the background level declines from left to right.
- A less conservative approach, but one that carries a different set of assumptions, would be to decline the low-side average background/bin linearly toward the average value on the right.
  - **I leave this as an exercise for the student (I found S = 10.8 +/- 7.4 events, for a significance of S/σ = 1.4 standard deviations above background).**
- ***This is the method of using “sidebands” around a bump to estimate background in the bump. It's a whole area of effort in an analysis. It goes by other names, too, like “Control Region” methods.***

Taking Full Advantage of all the Data: Modeling the Data

- This was a very rough exercise, intended to familiarize you with basic ideas:
  - Data consists of an unknown set of contributions from different sources
  - We can make assumptions to try to estimate some of those contributions
    - e.g. the bins to the left and right of the “bump” contain little or no events of the class that are causing the “bump”
- But we have lots more information we are not using!
  - The full shape of the data across its entire range
  - The fact that other people might have already considered models for a possible signal that would describe well and analytically this “bump”

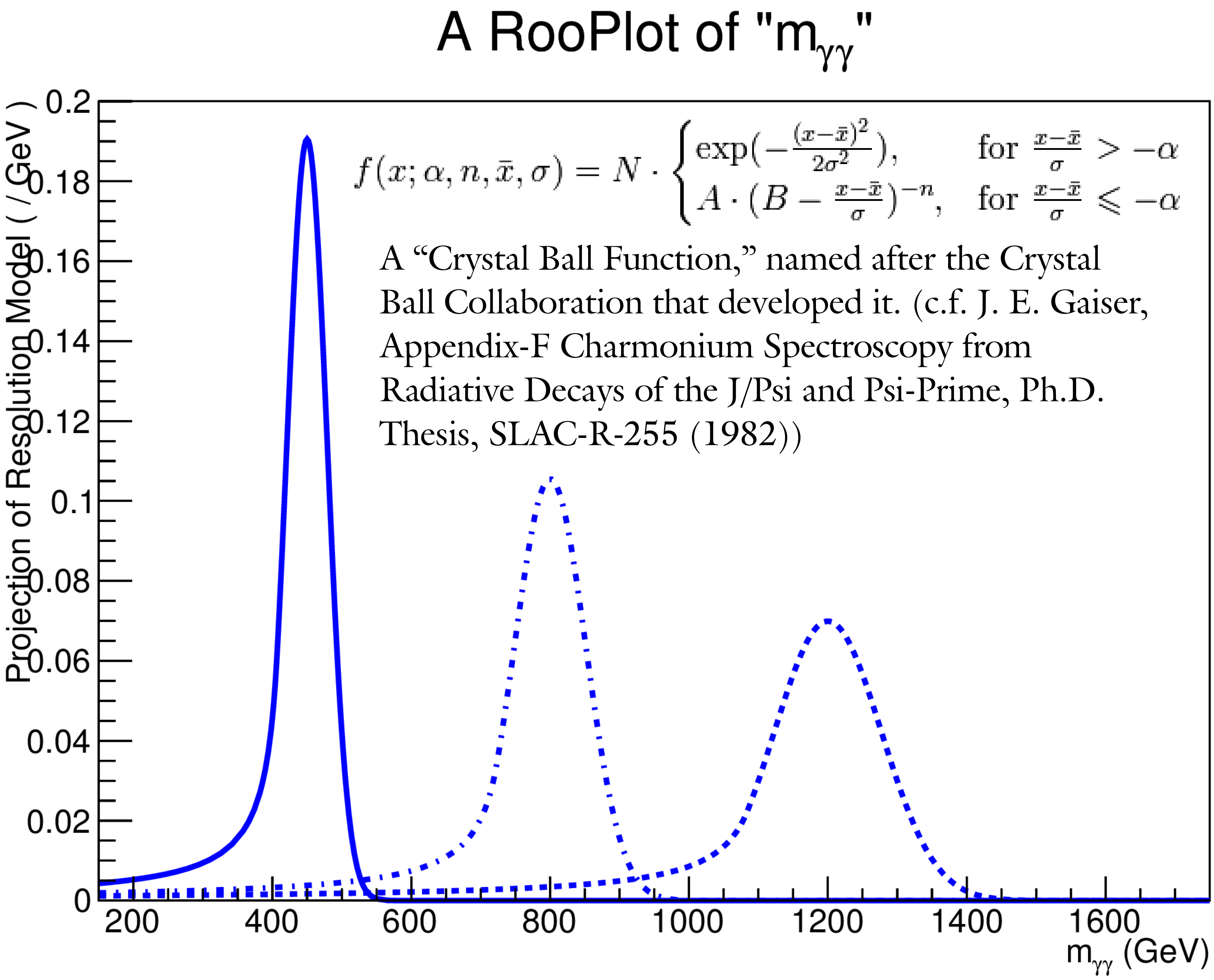
The Model

- We want to describe the data using a “Model”
  - Every model carries a number of assumptions
  - Each assumption is a place where potential error can creep in – as a result, each assumption should be assessed for its contribution of “systematic uncertainty”.
- We will use a two-component model
  - “The data consists of a non-structured/non-resonant background and a single resonant component.” (hypothesis)
    - We will build the model using RooFit and test the hypothesis against an alternative hypothesis:
      - “The data consists only of a non-structured/non-resonant background.”

The Signal Model

- Unfortunately, the expert who developed the final form of the signal model is in Beijing and isn't available to help with this effort.
  - However, they documented the model and we can rebuild it ourselves
  - It is based on a few factors:
    - First, diphoton invariant mass resolution in the region from 150-1750 GeV is about 4% of the mass
    - Second, the model we are testing predicts a non-trivial natural width for a diphoton resonance that is about 5% of the mass.
      - Added in quadrature, these effects total a 6.4% “width” on the mass peak
    - Finally, reconstruction of photons is imperfect due to energy losses in the calorimeter. This results in a longer tail in the distribution below the peak of the invariant mass for a real resonance.

Illustration of the Signal Model



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

A “Crystal Ball Function,” named after the Crystal Ball Collaboration that developed it. (c.f. J. E. Gaiser, Appendix-F Charmonium Spectroscopy from Radiative Decays of the J/Psi and Psi-Prime, Ph.D. Thesis, SLAC-R-255 (1982))

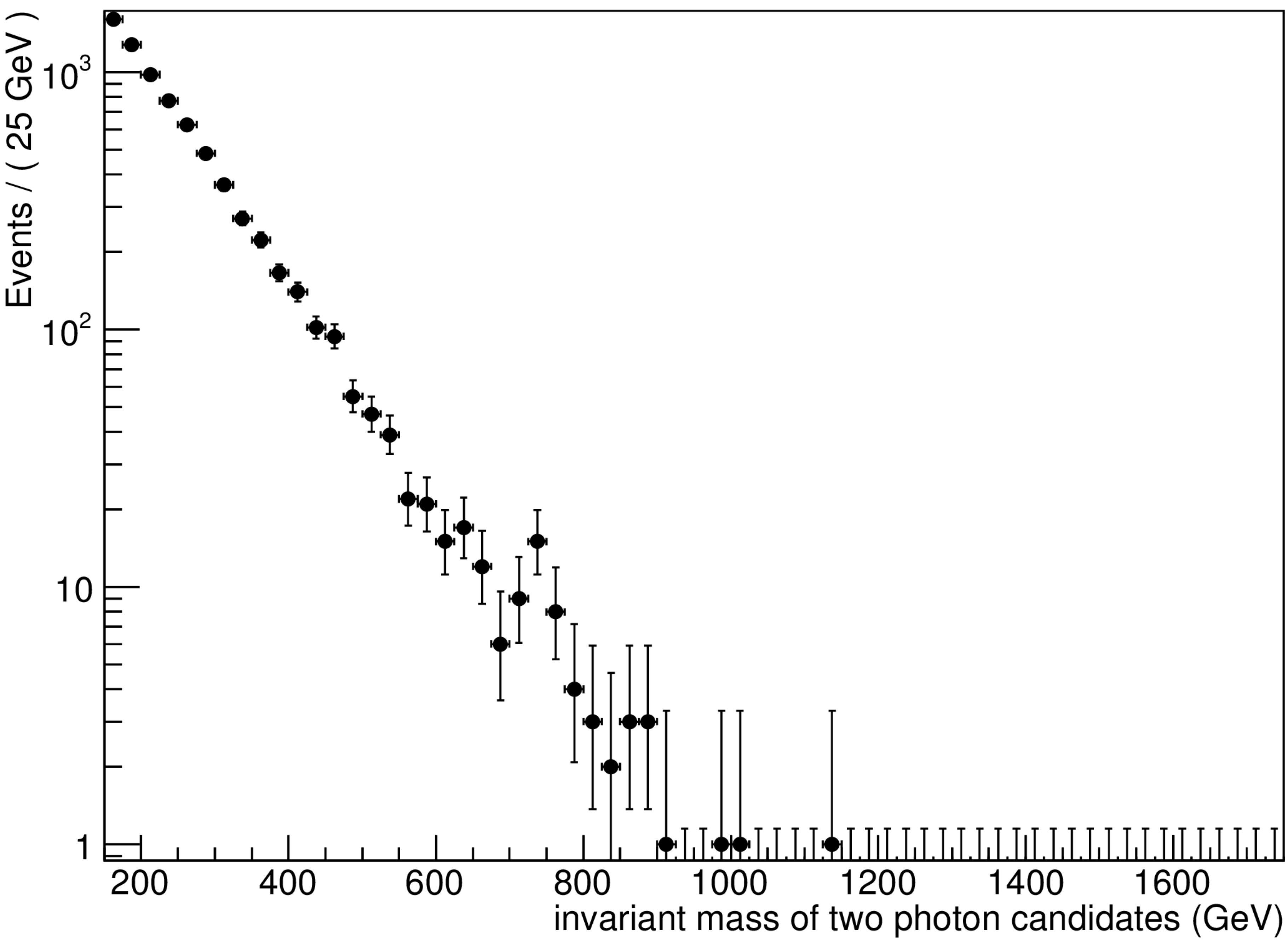
Building the Signal Model

```
// additional includes needed for this model
#include "RooFormulaVar.h"
#include "RooCBShape.h"


{
  // Code snippet for the Signal Model

  RooRealVar  mass("mass","m_{#gamma#gamma}",150,1750,"GeV");

  RooRealVar res_mean("res_mean","Resolution Mean",
                      700.0, 800.0, "GeV");
  RooFormulaVar res_width("res_width", "Resolution Width",
                          "res_mean*0.064", RooArgSet(res_mean));
  RooRealVar res_alpha("res_alpha", "Resolution alpha", 1.5);
  RooRealVar res_n("res_n", "Resolution order", 1.0);
  RooCBShape signal_model("res_model", "Resolution Model", mass,
                          res_mean, res_width, res_alpha, res_n);

}
```

Mathematically, we can denote this as $F_S(m_{\gamma\gamma})$

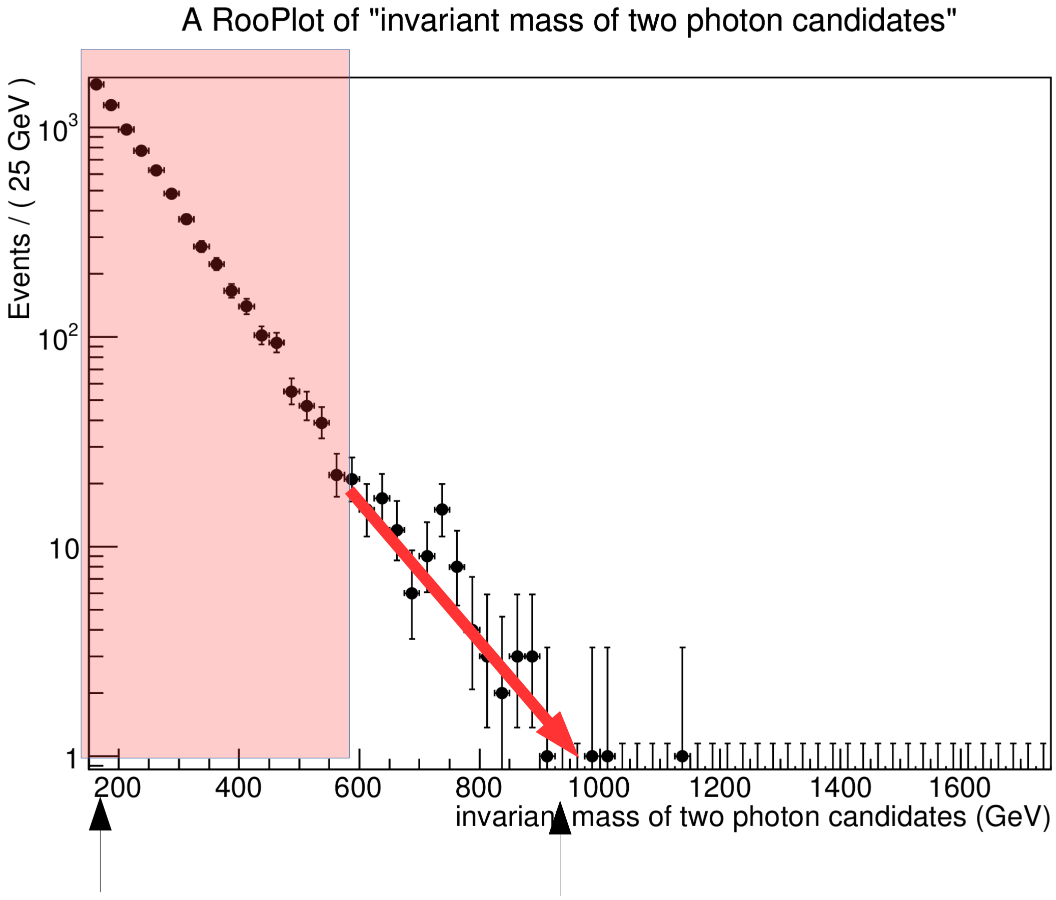
The Background Model

- We do NOT have a Monte Carlo simulation that reliably predicts this diphoton shape
- We will have to utilize the data itself to build a model
- We should try to choose a “reasonable” guess at an analytical model (polynomial, Chebychev polynomial, exponential, etc.) based on a discussion of the data trends.
- **Question: what model might best describe this data with as few free parameters as possible?**



A RooPlot of "invariant mass of two photon candidates"

Motivate the Background Model



Use data in this region to set the parameters of the background model…

Use the background model everywhere to now predict the shape of the data

We will:

- Develop an implementation of the background model
- Determine the model parameter(s) by fitting a restricted region of the data.
- Fix the parameter(s) for the model and then build a signal + background model to describe all data shown here.
- Determine the amount of signal in the "bump region"

Build the background model

```
// additional includes needed for this model
#include "RooExponential.h"


{
  // Code snippet for the Background Model

  RooRealVar bkg_par1("bkg_par1", "Background Model Parameter 1",
                      -1.0, 0, "1/GeV");
  RooExponential bkg_model("bkg_model", "Background Model", mass,
                           bkg_par1);
}
```

Determine the Background Model Parameter from Data

```
// additional includes needed for this step
#include "RooFitResult.h"


{
  // Code snippet for this step

  RooFitResult* bkgonly_result =
    bkg_model.fitTo(*data, RooFit::Save(),
                    RooFit::Verbose(kFALSE),
                    RooFit::Range(150.0, 500.0),
                    RooFit::Optimize(),
                    RooFit::Strategy(2));

  bkgonly_result->Print()
  RooPlot* plot = mass.frame();
  data->plotOn(plot,RooFit::Binning(64));
  bkg_model.plotOn(plot);
  plot->Draw();
  c1.SaveAs("FitData_bkgonly.pdf");
}
```

Results of the Background-Only Fit

```
  RooFitResult: minimized FCN value: 40393.7, estimated distance to minimum: 
8.54399e-07
                covariance matrix quality: Full, accurate covariance matrix
                Status : MIGRAD=0 HESSE=0

    Floating Parameter    FinalValue +/-  Error 
  --------------------  --------------------------
              bkg_par1   -1.0032e-02 +/-  1.34e-04
```

Comment: based on the results, our data-preferred model for the non-resonant/non-structured background is:

$$F_B(m_{\gamma\gamma}) = A\,e^{(-0.010032)\cdot m_{\gamma\gamma}}$$

What just happened? What does all of this mean? Let's briefly explore “MINUIT”, the algorithm that just did all the hard work for you!

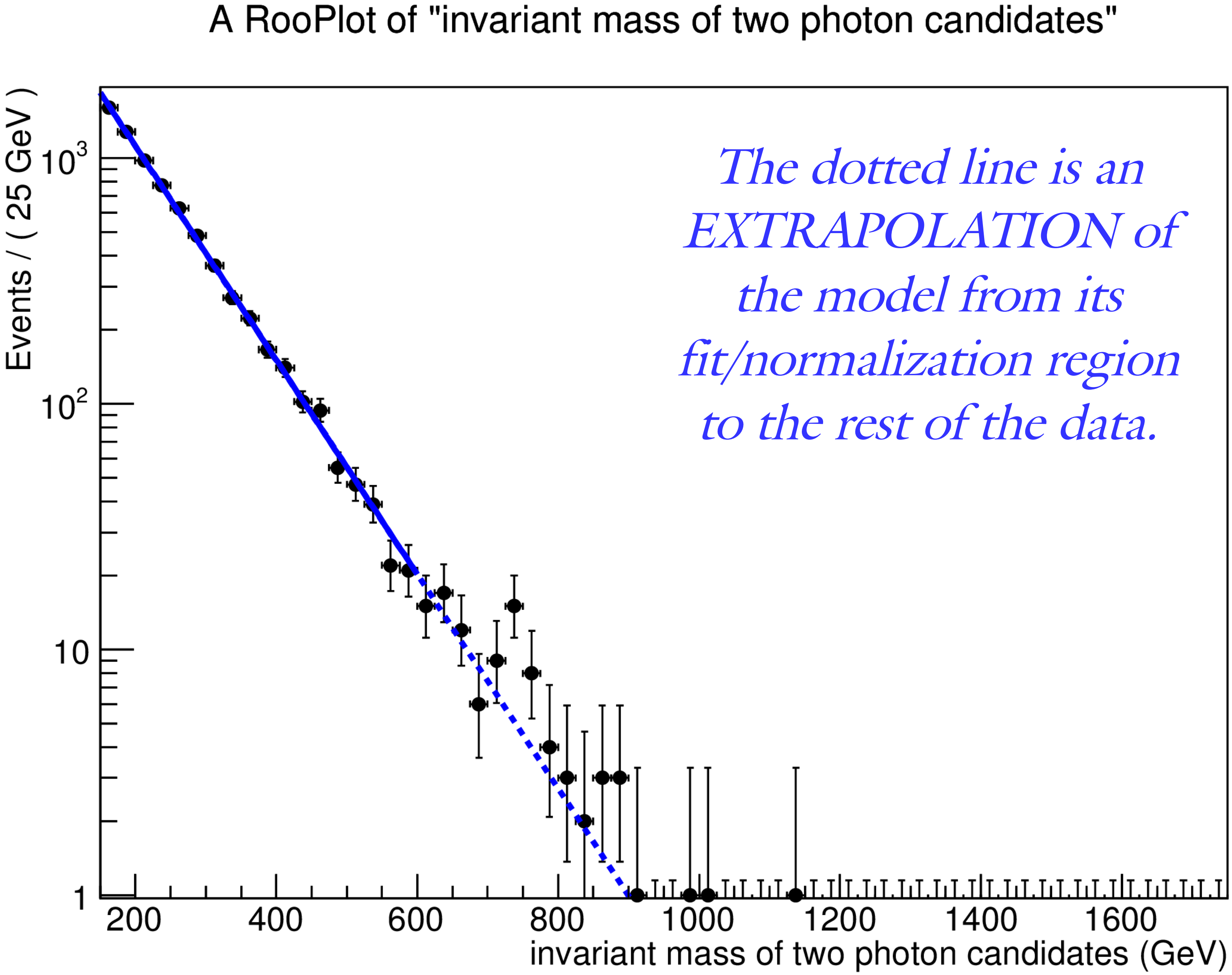
MINUIT Algorithm

- A “minimizer” - it's job is to find the minimum value of a function
- What function?
  - -log(L): the likelihood is the numerical value returned by evaluating a normalized probability density function when it is given a data set ({x}) and values for its parameters, e.g. L({a};{x}).
- MINUIT tries to scan over the space of values of {a} to find the minimum of -log(L) given {x}
- Details:

  James, F. “MINUIT User's Guide”.

  https://www.researchgate.net/publication/251575389_MINUIT_User's_Guide

Visualizing the Model


A RooPlot of "invariant mass of two photon candidates"

The dotted line is an EXTRAPOLATION of the model from its fit/normalization region to the rest of the data.

Visualization Code

```
// additional includes needed for this step

{
  // Code snippet for this step

  // define “background fit region” in variable “mass”
  mass.setRange("bkg_region", 150.0, 600.0);

  // Plot the function twice, once only in its fitted region
  // and once in all of the variable's range
  RooPlot* plot = mass.frame();
  data->plotOn(plot,RooFit::Binning(64));
  bkg_model.plotOn(plot);
  bkg_model.plotOn(plot,
                   RooFit::Range(150, 1750),
                   RooFit::NormRange("bkg_region"),
                   RooFit::LineStyle(kDashed));
  plot->Draw();
  c1.SaveAs("FitData_bkgonly.pdf");

}
```

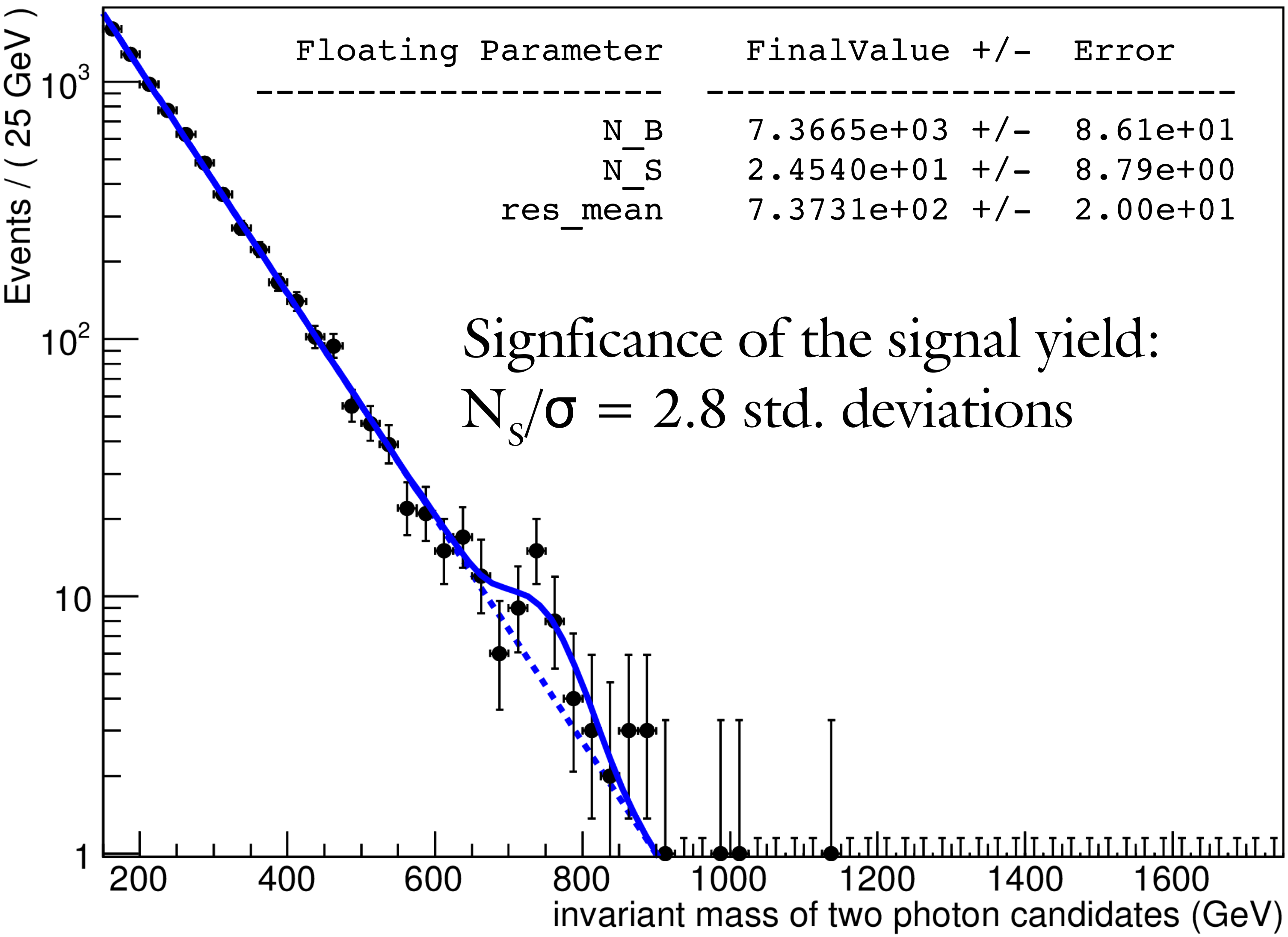
Compose the Full Data Model

$$L(N_S, N_B, m_{\gamma\gamma}) = N_S F_S(m_{\gamma\gamma}) + N_B F_B(m_{\gamma\gamma})$$

```
// additional includes needed for this step
#include “RooAddPdf.h”

{
  // Code snippet for this step
  bkg_par1.setConstant(kTRUE);

  // Total Model
  RooRealVar N_S("N_S","N_{S}", -1000.0, 10000.0);
  RooRealVar N_B("N_B","N_{B}", -1000.0, 10000.0);

  RooAddPdf data_model("data_model", "Signal + Background Model",
                      RooArgList(signal_model, bkg_model),
                      RooArgList(N_S, N_B));

  // Fit the data with the complete model

  RooFitResult* result =
     data_model.fitTo(*data,
                     RooFit::Save(),
                     RooFit::Verbose(kFALSE));
}
```

Signal and Background Yields from Data; Signal Significance

A RooPlot of "invariant mass of two photon candidates"



```
   Floating Parameter    FinalValue +/-  Error
--------------------  --------------------------
                 N_B    7.3665e+03 +/-  8.61e+01
                 N_S    2.4540e+01 +/-  8.79e+00
            res_mean    7.3731e+02 +/-  2.00e+01
```

Signficance of the signal yield:

$N_S/\sigma = 2.8$ std. deviations

What is the Probability that Background could explain this?

- The “p-value” is the probability that background (null hypothesis) by itself could have fluctuated upward to yield this number if “signal” events (or more).
  - The p-value only tells you the probability that the null hypothesis explains the data as well or better than your primary model. It does not tell you that your primarly model is “true” or “correct”
- To estimate p-values, we will use the method of “pseudoexperiments” or “toy Monte Carlo”
  - A basic discussion of Monte Carlo techniques and the “Accept/Reject” method at the heart of this approach is available in my PHYS 4321 lecture on Monte Carlo Methods.
  - http://www.physics.smu.edu/scalise/p4321/MonteCarlo.pdf

Generating and Fitting Background-Only Pseudoexperiments in RooFit

```
// additional includes needed for this step
#include “RooMCStudy.h”
#include “RooRandom.h”
#include “TRandom.h”

void PValue(UInt_t cycle = 0) {
  gRandom->SetSeed(20160414+cycle);
  RooRandom::randomGenerator()->SetSeed(20160414+cycle);

  // insert code here to define the models, fixing parameters
  // to those determined from the fit to data

  RooMCStudy mcStudy(bkg_model, RooArgSet(mass),
                     RooFit::FitModel(data_model));
  mcStudy.generateAndFit(100, 7391);

  RooDataSet ns(mcStudy.fitParDataSet());
  TH1F* ns_hist =
   (TH1F*) ns.createHistogram("ns_hist", N_S,
                              RooFit::Binning(10000,-500,500));

  TFile fout(Form("pvalue-%d.root", cycle),"recreate");
  fout.cd();
  ns_hist->Write();
  fout.Close();
}
```

Running on ManeFrame

Create a SLURM batch configuraton file. Call it “pvalue.slurm”.
Its contents should look like this:

```
#!/bin/bash
#SBATCH -J PHYS7361-%a    # job name
#SBATCH -o pvalue-%a.out
#SBATCH -e pvalue-%a.out
#SBATCH -n 1               # total number of tasks requested
#SBATCH -N 1               # total number of nodes
#SBATCH -p serial          # queue (partition) -- batch, parallel, etc.
#SBATCH -t 00:10:00        # run time (hh:mm:ss)
#SBATCH -D .               # Directory where executable will be run
/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/x86_64/root/6.04.14-x86_64-
slc6-gcc49-opt/bin/root -q -l -b PValue.C'+('$SLURM_ARRAY_TASK_ID')'
```

Then run 1000 jobs, each with a unique task ID, as follows:

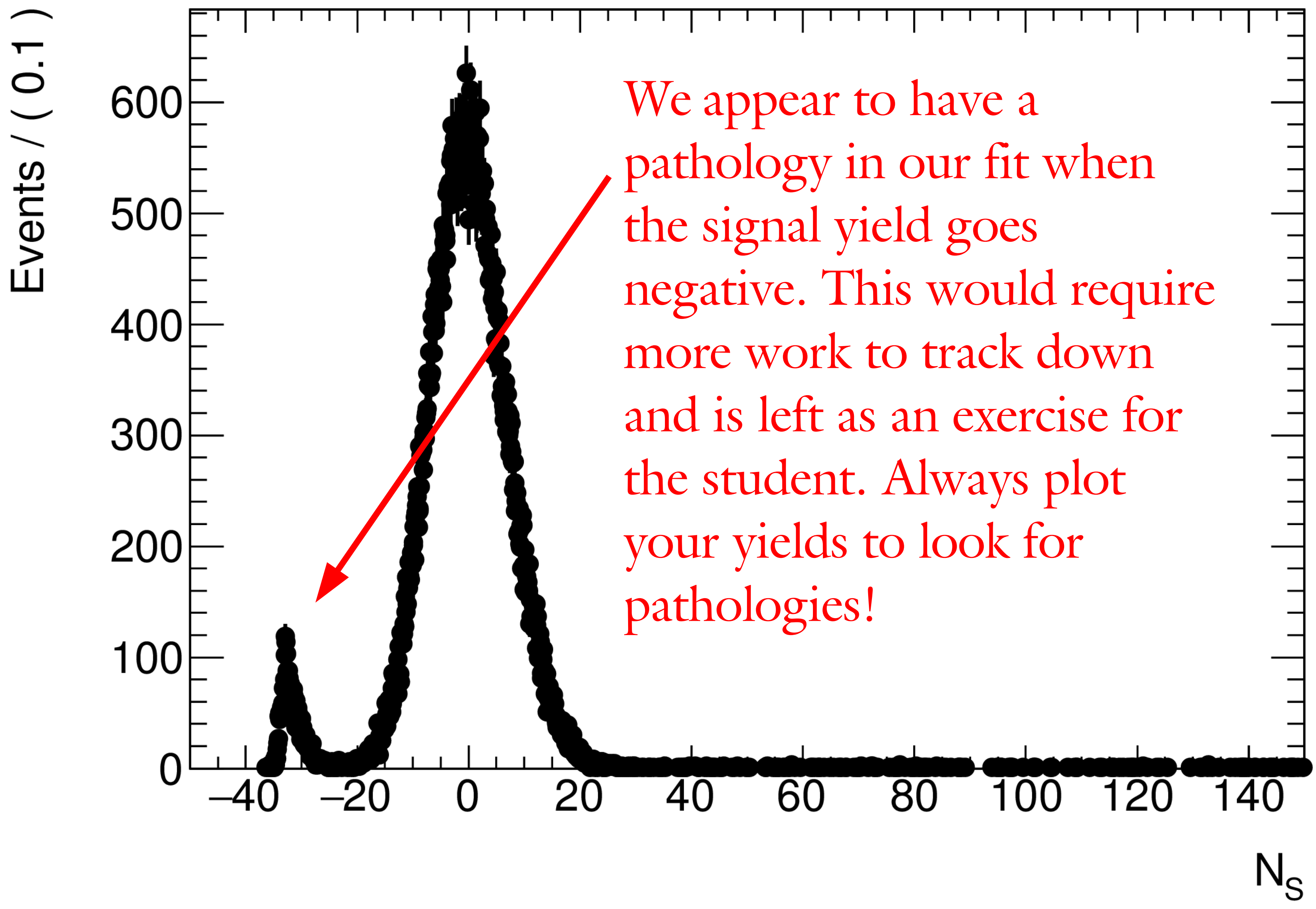
```
sbatch -a 1-1000 pvalue.slurm
```

Wait about 5 minutes for all the jobs to finish. You should have a bunch of root files named “pvalue-XXX.root”. Add them together:

```
hadd pvalue.root pvalue-*.root
```

You now have a single ROOT file containing a big histogram named “ns_hist__N_S”

Visualizing the Signal Yields from the Pseudoexperiments



We appear to have a pathology in our fit when the signal yield goes negative. This would require more work to track down and is left as an exercise for the student. Always plot your yields to look for pathologies!

Computing the p-values

$p_0 = P\left(N_S > N_S^{data} | H_0\right)$ This is merely the probability of the result given the null hypothesis (background-only model)

```
{
  TFile f("pvalue.root");
  TH1F* hist = (TH1F*) f.Get("ns_hist__N_S");
  Int_t ns_bin = hist->FindBin(24.54);
  Double_t N_above_NS = hist->Integral(ns_bin, 10000);
  Double_t pvalue = N_above_NS/10000.;
}
```

I obtain a p-value of 0.00402, or 0.4% probability that the background-only hypothesis explains the data. This DOES NOT tell you the probability of the alternative hypothesis being "true" or "correct" - it merely suggests that one might want to explore this "bump" more closely.

Also, keep in mind we've only included statistical uncertainties in our assessment – adding systematic uncertainties will increase the p-value.

Additional Exercises (I)

- Go back to the fits and see how sensitive the results are to changes in parameter ranges or starting values, e.g. the mean of the Crystal Ball Function, or the slope of the exponential. Use RooRealVar::setRange() and RooRealVar::setVal() to adjust these options.
- “Scan” in mass around the best-fit value. To do this, set the signal model peak parameter to constant (RooRealVar::setConstant(kTRUE)) and then set its value to a specific number (RooRealVar::setVal(XXX)). Redo your pseudoexperiments fits. Recompute the p-value. Try this for 4 mass values around the best-fit value. What is the trend in p-value around the best-fit value?

Additional Exercises (II)

- Instead of testing the probability of the null hypothesis as a good explanation for the data, instead try to compute a "confidence level" that the true value of the signal yield lies below your observed level. Use the 95% confidence level.
  - HINTS: you want to run pseudoexperiments using your full model. Increase slowly the true number of signal events you inject (N_S^{true}) into each ensemble of pseudoexperiments. Find the value of N_S^{true} such that 95% of the time the fitted signal yield lies BELOW N_S^{data}. This value of N_S^{true} is the 95% confidence level upper limit on the true yield of signal in the data, given the background and signal models of choice.

Additional Exercises (III)

- Look at the “pull distributions” of your signal and background yield fits in the pseudoexperiments.
  - A “pull” is the difference between the true and fitted values, divided by the uncertainty on the fitted value. Typically, you expect this to be a distribution centered at 0, distributed according to Gaussian statistics (random, uncorrelated yields from fit to fit), with a Gaussian width parameter of 1.0 (meaning that typically, 68% of the time your fitted yield lies withing 1 standard deviation of the true value).
  - If this pull distribution is shifted from 0.0, you have a “bias” in your fit procedure that favors values other than the true value. Check your model for fit pathologies (e.g. are parameters hitting their range boundaries? Are fits failing for other reasons. Read your log files!)
  - If the pull distribution has a width other than 1.0, or is skewed, your errors are not estimated correctly. Again, check for fit pathologies.