Exercise 1: Search for H -> gamma gamma

Part 5:

- **Signal strength extraction**:
  - Perform a signal + background fit. For stable fit behaviour, set the initial fit values of the background part to those obtained from the previous fit, and the signal ones to values that seem sensible.
  - From the signal component of this fit, determine the number of signal events, and compare to the number we expect from the signal simulation. The ratio extracted : expected is called the signal strength.

Ok great, that looks awesome, but how do we know how strong our signal excess is?

For this, we can do a signal + background fit. The signal looks fairly Gaussian, so let's try that

In [ ]:

```
TF1 *fit_sig_bkg = new TF1("fit_sig_bkg", "[0] + exp([2]*x+[1]) + [3]*TMath::Gaus(x,[4],[5])", 100, 160);
```

In order to help the fit converge, now that it has quite a lot of parameters, we can initialise the parameters to values that are about right

In [ ]:

```
fit_sig_bkg->SetParameter(0, fit->GetParameter(0));
fit_sig_bkg->SetParameter(1, fit->GetParameter(1));
fit_sig_bkg->SetParameter(2, fit->GetParameter(2));
fit_sig_bkg->SetParameter(4, 125);
fit_sig_bkg->SetParameter(5, 3);
```

Now we can do the fit and draw it on our plot

In [ ]:

```
pad1 -> cd();
fit_sig_bkg -> SetLineColor(kBlack);
h_data -> Fit(fit_sig_bkg);
fLegend -> AddEntry(fit_sig_bkg, "Fit signal+bkg", "l");
fLegend -> Draw(); // redraw the legend
```

We can pull the fit parameters using the GetParameter() function

In [ ]:

```
cout << "Fitted signal normalisation: " << fit_sig_bkg->GetParameter(3) << " +- " << fit_sig_bkg->GetParError(3) << endl;
cout << "Fitted signal mean: "          << fit_sig_bkg->GetParameter(4) << " +- " << fit_sig_bkg->GetParError(4) << endl;
cout << "Fitted signal width: "         << fit_sig_bkg->GetParameter(5) << " +- " << fit_sig_bkg->GetParError(5) << endl;
```

The integral over all x of a gaussian is (exercise for student :-)) sqrt(pi). For a Gaussian of width `s` and normalisation `n` it is therefore just `s * n * sqrt(pi)`. So, we can calculate the number of signal events:

In [ ]:

```
float fittedSignalEvents = fit_sig_bkg->GetParameter(3) * fit_sig_bkg->GetParameter(5) * TMath::Sqrt(TMath::Pi());
```

Now we can compare this to the number of signal events we expect from simulation. The signal strength mu is defined as the number of events we extract above our background prediction divided by the number of signal events predicted by our model. It should be close to 1 if we are seeing the same thing as our prediction!

In [ ]:

```
float mu = fittedSignalEvents / hSig -> Integral();
cout << "Signal strength: " << mu << endl;
```

To round things off, let's plot the fitted signal in the ratio plot so we can see it by eye

In [ ]:

```
pad2 -> cd();
TF1 * fit_sig = new TF1("fit_sig","gaus(0)",100, 160);
fit_sig->SetParameter(0, fit_sig_bkg->GetParameter(3));
fit_sig->SetParameter(1, fit_sig_bkg->GetParameter(4));
fit_sig->SetParameter(2, fit_sig_bkg->GetParameter(5));
fit_sig->SetLineColor(kBlack);
fit_sig->Draw("same");
```