ROOT practicalities

- Practical workflow:
  - You can run on your laptop (if you installed it yourself), or run on stoomboot
  - You need ROOT version 5.34.21 or higher, preferably 5.34.36
  - I have tested all exercise macros with ROOT 5.34.36, which I installed on stoomboot. You can use this version as follows:

    ```
    source /project/atlas/user/verkerke/root-53436/bin/thisroot.sh
    (or .csh)
    ```
- Input files for exercises (ex1.C etc) can be found in directory `~verkerke/stat2016/` on stoomboot or login.nikhef.nl
- Annotations in exercises:
  - 'CODE' – means that you need to write some code
  - 'EXEC' – means that you need to run your code and interpret its output
- Macros have been tested with ROOT 5.34/21 & 5.34.36
  - Problems? Please ask (I didn't test everything on all platforms)

Exercise 5 – Unbinned Maximum Likelihood fit

- This a mostly a demonstration exercise only that shows some of the functionality and the syntax of the RooFit toolkit for data modeling that we'll be using in the next exercises
- Copy file `ex5.C`, look at it and run it. This macro does the following
  - It creates a Gaussian probability density function
  - It generates an unbinned dataset with 10k events from that function
  - It performs and unbinned ML fit of the Gaussian model to the toy dataset
  - It makes a plot of the data and overlays it with the Gaussian model
- Now comment out the part of the code labeled as 'block 1' and run the macro again
  - This code will print out the covariance matrix and correlation matrix of the fit parameters. Verify that cov(m,s)=corr(m,s)*σ(m)*σ(s) using the printed errors for mean.
- Now comment out the code labeled 'block 2' and run again.
  - This code will visualize the uncertainty of the model on the canvas using the error propagation technique. At 10K the event the uncertainty is very small (you can see it if you zoom in on the peak region of the pdf)

Exercise 5 – Unbinned Maximum Likelihood fit

- Change the number of generated events from 10K to 100 and change the binning of the data in the plot from 100 bins to 10 bins (this is the argument in the `w.var("x")->frame()` call. Run again.
- Lower the number of generated events from 100 to 10 and run again. The error on the shape will now be significant, but you see that an unbinned ML can reliably fit very small event samples

- Now comment out code block 3
  - This will visualize the error on the pdf shape due to the uncertainty on the mean parameter only

Exercise 6 – Analytical vs numeric MLE

- For certain simple cases it is possible to calculate the ML estimate of a parameter analytically rather than relying on a numeric estimate. A well known case is that of the fit of a lifetime of an exponential distribution
- Copy `ex6.C` and run it. This example performs an unbinned MLE fit to an exponential distribution of 100 events to estimate the lifetime.
- Now we aim to construct the *analytical* ML estimator of the lifetime (do this part on paper, not by computer)
  - Write down the probability density function for the exponential lifetime distribution.
  - It is essential that you formulate a *normalized* expression, i.e. Int F(t) dt = 1 when integrated from 0 to ∞.
    The easiest way to accomplish that is to divide whatever you expression you have by the integral of that expression over dt. (You can calculate that normalization integral on paper)

Exercise 6 – Analytical vs numeric MLE

- Next write down the analytical form of the negative log-likelihood –log(L) for that probability density function given a dataset of N events labeled these x_i in your expression. *Be sure to also include the pdf normalization term in the expression*
- The analytical ML estimate of the lifetime tau then follows the requirement that d(-logL)/dtau = 0. Calculate the expression for this derivative solve and derive the value of tau for which the requirement d(-logL)/dtau holds.

- Finally, implement the analytical calculation of MLE estimator for tau in the code of `ex6`.
  - Uncomment block one, which implements a look over the dataset, retrieving the values of the decay time one by one and build your calculation of the analytical estimate of tau with that of the numeric calculation from fitTo()
  - Explain why you might have minor discrepancies between the analytical and numeric calculations.
  - Increase the event count from 100 to 10000 and run again

Exercise 7 – The effect of outliers on fits

- Outliers in distributions that are strongly peaked can create serious converges problems in fits that model such peaked distributions and do not take outliers into account.
  - Copy `ex7.C`, look at it and run it. This example generates a Gaussian distribution with a width that is relatively narrow compared to the defined range on x, and fits it to a Gaussian model.
  - Now uncomment BLOCK1. The added code 'manually' adds an event at x=3 and refits the distribution. Look at both fits carefully (just by eye, no need to make a true quantitivative check using a χ^2). Zoom in on the x axis if necessary. Does the outlier impact the result of the fit? (Also check the impact of the fitted value of mean,sigma)
  - Now move the position of the outlier event to x=4, run again and evaluate the situation again. Calculate what is the probability to obtain an event at x=4 or larger for this model? (You can use the 'TMath::Erfc' formula of Ex 1 to calculate this, but keep in mind that that formula evaluate the probability for |x|>Z, rather than x>Z)
  - Repeat the above exercise (including evaluation) at x=9.
  - Now uncomment BLOCK2. This fits the data to an improved model that foresees in a (small) flat background component that absorbs the outlier events and make the fit of the Gaussian component of the model function well in the presence of outliers. What value of `fsig` do you expect a priori to get out from the fit?
  - Now change the code fragment '`fsig[0,1]`' by '`fsig[0.999']` modifies to model to have a one permille fixed background component instead of a floating component. Rerun the fit. Does it still work well? Explain why (not)?

Exercise 8 – An MLE fit for an efficiency function

- This exercise demonstrates the procedure of an unbinned ML fit for an efficiency curve.
  - Copy `ex8.C`, look at it and run it. This macro create a data sample (x,c) in which c is a discrete observable that tells us if the event with value x has passed a selection (or not). The goal is to determine the efficiency function `eff(x)` of the selection encoded in observable c.
  - The initial exercise creates an efficiency histogram from the dataset D(x,c) where each bin in x contains the fraction of events that have passed the cut. The efficiency histogram is constructed to have symmetric binomial errors on the data. Look at the slides of Module 1 to remind yourself how symmetric binomial errors are defined. An efficiency function 'fitfunc' is then fit to the data using a χ^2 fit. Explain what approximation we are making by doing this?
  - Now uncomment BLOCK 1 of the the exercise and run again. This will make a plot 'all data' and 'accepted data', as well as perform an unbinned ML fit to measure the efficiency function. This fit is done using a probability density function F(c|x) that returns 'effFunc(x)' if the event is accepted and '1-effFunc(x)' if the event is rejected and is fit to the full dataset D(x,c)
  - Lower the number of events generated from 1000 to 200 (change variable N) and see what happens. Do the χ2 and likelihood fits return correct results? Then lower N to 50 and run again.

Exercise 9 – A Poisson counting experiment

- Run macro `ex9.C`.
- This macro does the following for you:
  - It creates an empty RooFit workspace
  - Fills the workspace a Poisson probability model Poisson(N,S+B) with B fixed to 2, and signal floating (but chosen at 0)
  - It prints the contents workspace: it will show 3 variables (B,N,S) one function object Nexp(B,S) and one probability model 'model(N,Nexp)'.
- Look at the macro and understand how the variables and function objects are created
- Plotting the probability model
  - Comment the return statement at the STEP1 comment, and run again.
  - The macro will proceed to make a plot of the probability model for the observable N, for the parameter configuration B=5,S=0
  - Uncomment the return statement at the STEP2 comment, and run again.
  - The macro will change the value of S from 0 to 2, and plot the distribution of N on the same plot frame

Exercise 10 – Adding a nuisance parameter

- We will now move to one of the core topic of this lecture: introducing a systematic uncertainty the model of ex9 by introducing a subsidiary measurement and a nuisance parameters
- Run macro `ex10.C`
- This macro does the following for you
  - It makes a slight variation of the model of Ex1, but expresses the signal strength as the product S*mu of the (fixed) nominal signal strength S and a floating signal strength modifier mu (the modifier is then independent of the absolute yield, mu=0 → no signal, mu=1 → expected signal, mu=2 → twice expected signal)
- Now we introduce a nuisance parameter
  - Make a fit (either using RooMinimizer or using fitTo) that measures the uncertainty on mu, using both HESSE and MINOS. [Insert code before the Step1 return]
  - OPTIONAL: Make a plot of –logL versus mu in the range [0,2] using your experience of Ex 2.

Exercise 10 – continued

- Now we introduce a nuisance parameter (continued)
  - Now comment the step-1 return statement.
  - Now make a fit of 'model2' similar to the fit of 'model' before
  - Compare what parameters are fitted, what the fitted values are, and how the uncertainties on the fitted parameters compare
  - What happens to the uncertainty on mu between the 1st and 2nd fit?
- Congratulations – you have just performed your first profile likelihood fit that includes a systematic uncertainty (on the background estimate) in your fitted estimate of mu!