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 11 – A sideband measurement

- We will now explore the similarity between subsidiary measurements and sideband measurements
  - In the model of Ex9 the background rate was constrained by a Gaussian subsidiary measurement that measurement B=20 with an uncertainty of 5
- Run macro `ex11.C`
- This macro does the following for you
  - It rebuild the model of Ex 10 in a compact syntax, and fits it to the data
- Now we rebuild the model assuming that B is measurement in a control region, rather than describing an 'abstract' Gaussian uncertainty
  - Construct a Poisson model for a fictitious control region that measures the model parameter B from an observed number of event NCTL=20 in the control region (Hint: name this model '`control_model`', and name the observable for this control region '`Nctl`' and set it to a constant value of 20
  - Once the control measurement is made, construct a new product (name it '`model3`' of the original measurement '`model`' and '`control_model`')
  - Fit `model3` to the data, compared the results

Exercise 11– continued

- Comparing the results
  - You will find that the uncertainty on mu between the fit to `model2` and `model3` is somewhat different. This is driven by the fact that the uncertainty on B in both models is also somewhat different: `model2` implements a Gaussian uncertainty of width 5, whereas the sideband measurement with `Nctl` measures and uncertainty of sqrt(20).
  - We have so far assumed that the control region measures the same B as '`model`', but it could very well be that the control region is larger, and would effectively measure twice the rate (i.e. if Nctl =40 then B=20). To introduce this effect of the 'size' of the control region, we introduce an extra (constant) parameter in the model that expresses this rescaling: Construct a new sideband model (name it `model_control2`) that implements Poisson(Nctl| tau*B) where tau is a constant parameter with value 2.
    Hint: you can use an "`expr::tauxb('tau*b',tau,b)`" function expression to construct an object that represents the product 'tau*b'.
  - Once this is done, construct a new full model (named model4) that is the product of '`model`' and '`model_control2`' and fit this again to the data. What happens to the uncertainty on B and mu?
  - What value of tau should you use to obtain uncertainties on B and tau that are identical to those of `model2`?

Exercise 12

- Template fits
  - We will now construct a first template fit, where a signal and a background model are described by a histogram obtained from MC simulation
- Run ex12.C
  - Note that this macro uses input file `ex12.root`
- This macro does the following for you
  - It opens ex12.root and uses the a template histogram in `ex12.root` to construct a probability model for 'signal' in an observable x
- Performing a simple template fit
  - Open first `ex12.root` and look at the `TH1` histograms stored in here: there is a signal template, a background template and a 'data' histogram
  - In a new root session, run macro `ex12.C`. You now see the signal histogram used to construct a yield function (a `RooHistFunc`) in. Add code to also do this for the background template (the `TH1` is called `h_bkg`, name the corresponding `RooDataHist` and `RooHistFunc` `dh_bkg` and `fh_bkg` respectively)

Exercise 12 - continued

- Performing a simple template fit
  - Now construct from the sum of two yield functions a probability model as follows (in the workspace factory)

    ```
    ASUM::model(mu[1,0,5]*hf_sig,nu[1]*hf_bkg)
    ```

    This class takes two yield histograms and turns the weighted sum of these in a probability model that can fitted.
  - Fit the model to the data, make a plot of the data overlaid with the fitted model (hint: first call `data.plotOn(frame)` and then `model.plotOn(frame)`. You can also overlay the background component of the model using

    ```
    pdf("model")->plotOn(frame,
                 Components("hf_bkg"),LineStyle(kDashed)) ;)
    ```
  - OPTIONAL: repeat this exercise with different templates and datasets to observe how signal/background shape and yields affect the fitted signal rate mu. To make these modified inputs, copy file `makeinput_ex10.C`, adjust the parameters inside it, and run it to regenerate `ex10.root`

Exercise 13

- Constructing a template morphing model that accounts for a 'jet energy scale' (JES) uncertainty in the signal template
- Run macro `ex13.C`
- What does this macro do for you?
  - It opens `ex13.root` and uses the a template histogram in ex13.root to construct a probability model for 'signal plus background' in an observable x
  - Note that we switched back to 100 bins for a more 'dramatic' visualization
- Constructing a template morphing model
  - Run the macro as provided and observe the fit result and plotted result.
  - The first step towards setting up a template morphing model is constructing `HistFunc` objects for the JES-up and JES-down variation templates (the datasets are already imported by the macro)
  - The next step is to make a template morphing signal model. The 'magic' class to do this is called `PiecewiseInterpolation.` The workspace factory string to make such an object is

    ```
    PiecewiseInterpolation::pi_sig(Fnom,Flo,Fhi,NP)
    ```

    where `Fnom/lo/hi` are the `RooHistFuncs` representing the nominal, down and up templates and NP is the nuisance parameter associated with the systematic uncertainty. Construct the `PiecewiseInterpolation` function, and the nuisance parameter (call that one 'alpha' with a range [-5,5]).

Exercise 13 - continued

- Constructing a template morphing model
  - Make a 2D plot of the template morphing signal model in the observable x and the nuisance parameter alpha

    ```
    w->function("pi_sig")->createHistogram("x,alpha")-
    >Draw("SURF")
    ```

  - You will clearly see that in the default configuration the signal model is allowed to extrapolate to negative signal yields. Disable this feature (`w->function("pi_sig")->setPositiveDefinite(kTRUE)`) and remake the above plot
  - You also clearly see the kinks in the predictions at alpha=0, as the model by default implements a piece-wise linear model. Switch this to polynomial interpolation model (`w->function("pi_sig")->setAllInterpCodes(4)`) and remake the above plot.
  - Finally construct the full template morphing model by 1) replacing in the '`model`', the simple signal model '`hf_sig`' with the morphing model '`pi_sig`' 2) constructing the full likelihood '`model2`' as the product of '`model`' and Gaussian subsidiary measurement on alpha (with observed value 0 and width 1)
  - Fit the template morphing model to the data and observe the effect of the introduction of the JES uncertainty on mu.
  - Also look at the fitted value of alpha and its uncertainty. Is the physics measurement able to constrain the JES uncertainty beyond the 'input' of the subsidiary measurement?

Exercise 15 – (Optional, skip if you are short on time!)

- Performing a template fit accounting for MC statistical uncertainties 'Beeston-Barlow-style'
- Run macro `ex15.C`
  - Note that this macro uses input file `ex15.root`
- This macro does the following for you
  - It opens `ex15.root` and uses the template histograms in `ex15.root` to construct a probability model for 'signal plus background' in an observable x
  - Note that the number of bins has changed from 100 to 20
- A template fit accounting for statistical uncertainties
  - Perform a fit of the 'model' to the 'data' dataset and plot the dataset and model overlaid, following the example of ex12.
  - Now change the 'rigid' template for signal and background in a 'flexible' template for signal and background as follows:: change class `HistFunc` in class `RooParamHistFunc`
  - When you fit again you will that result is (still) the same, as parameters that can change each bin the templates are initially constant.

Exercise 15 – (Optional, skip if you are short on time!)

- A template fit accounting for statistical uncertainties
  - Now we need to construct the classes that introduces the subsidiary Poisson measurements that constrain the parameters of the flexible template parameters to the “measured” MC event counts:

    ```
    HistConstraint::hc_sig(hf_sig)
    ```

    The only constructor argument is the template function (`RooParamHistFunc`, named ‘`hf_sig`’ in the code example above) for which it makes subsidiary measurement.

    (The construction of this subsidiary measurement will ‘automagically’ make all parameters of the `RooHistFunc` floating)

    Construct objects of type for both the signal and background template (name them `hc_sig` and `hc_bkg`)

    Finally, construct the full model multiplying the template model and the two `HistConstraint` objects (use `PROD::model2(….)` to construct the product.

    - Note that you can use one PROD() object to multiply any number of models
  - Fit the template ‘model2’ that now includes Beeston-Barlow MC statistical uncertainty treatment. Look at the values of all fit parameters and in particular compare the uncertainty on mu of this fit w.r.t. the earlier fit to the rigid template model. Is the difference between mu uncertainties consistent with your expectation?