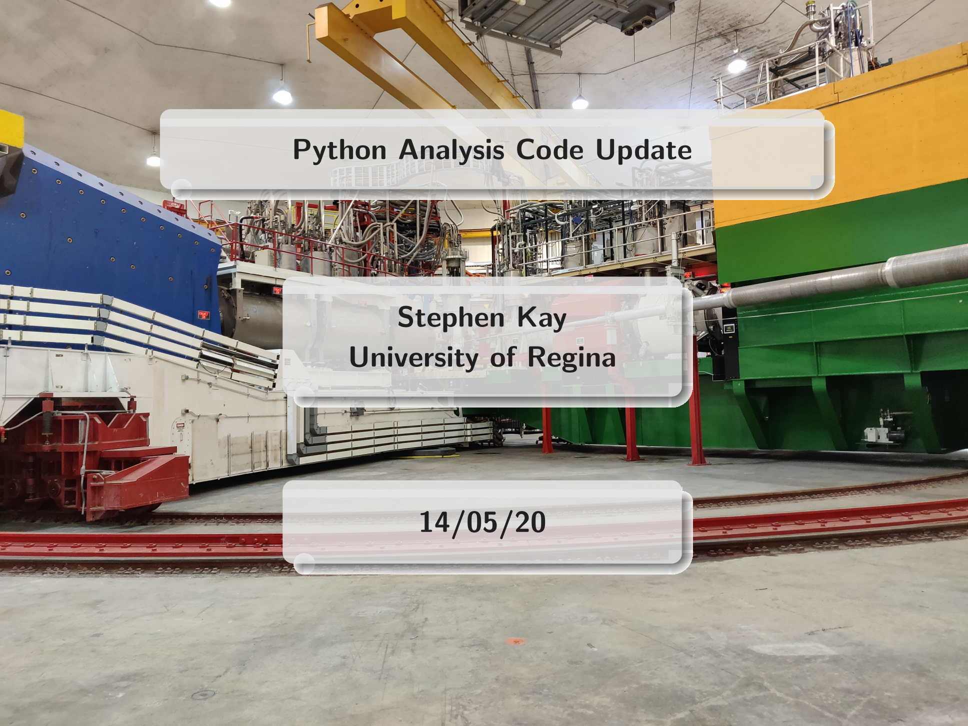


Python Analysis Code Update

Stephen Kay
University of Regina

14/05/20

Introduction

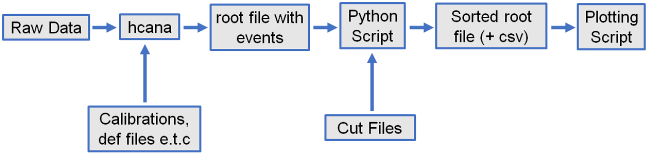
- Switching to python based analysis structure
- Aim for this to be clearer, more transportable and more accessible
- Need to be sure analysis is “working”
  - Compared new analysis code to previous TProof based scripts

Previous Script Issues

- Previously used a root macro to process replayed files and apply cuts, plot and save data
  - All in one script quickly bloats and becomes quite cumbersome
- Old scripts used root TProof to process data
  - TProof parallelises the processing of a chain
  - Fast, once it gets going
  - Very non-intuitive, debugging is not straightforward
  - Setup and initialisation of the analysis can be slow
- Previous scripts had hardcoded cuts and outputs in places
  - Not very flexible

General Data Flow

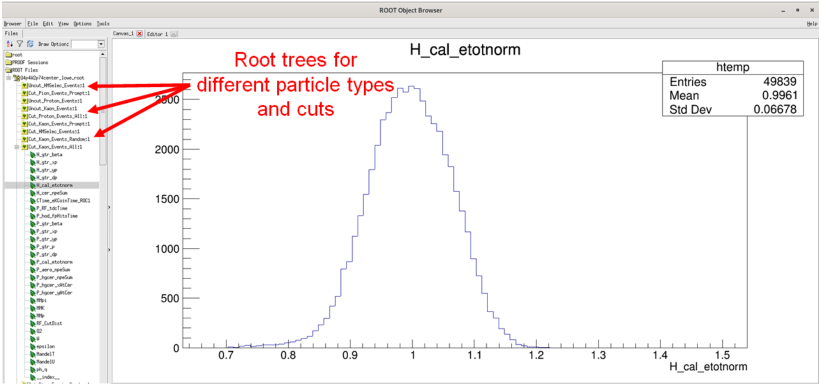
- Starting from raw data, process through hcana
- Get a resulting root file based on our defined def files etc.
- Run large root file through python analysis script, get a trimmed and sorted root file (and csv if desired) as output
  - Choice from the user as to which they use after that
  - Could use python based plotting/fitting if they want



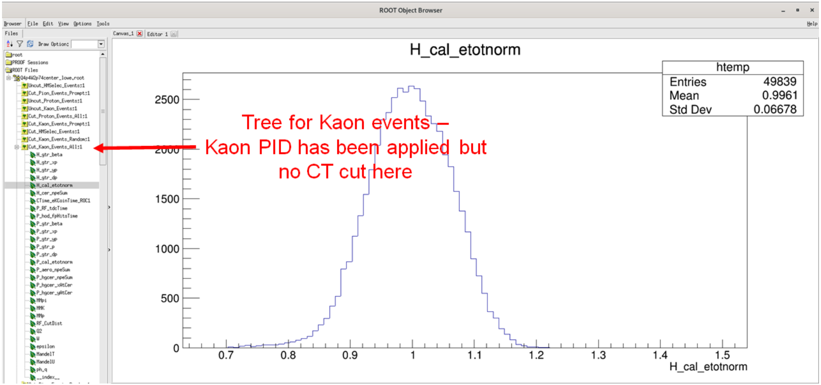
Python Script

- Python analysis script takes a replayed root file and trims it down to a smaller, more manageable file
- Select out the branches you want, apply the cuts you want and save the output
  - Output saved as leaves in trees you can define
  - No longer all in one
  - New output can be as small or as complicated as you want
- Cuts are applied based on values that are read in from parameter files
  - No more hardcoded cuts
  - Just tweak the values for the run you're looking at in the parameter files

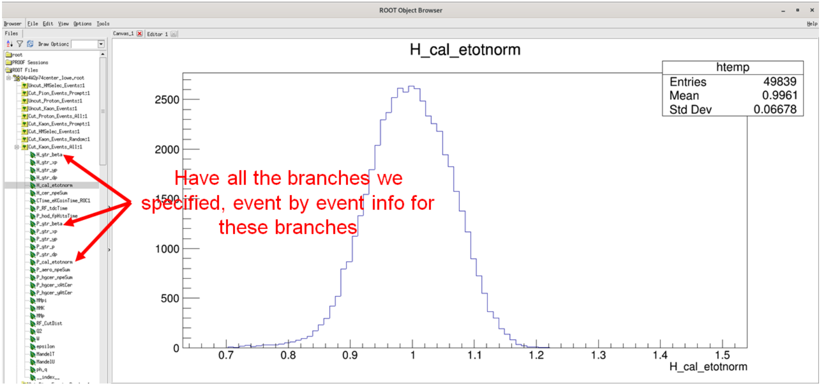
Python Script - Output Example 1/4



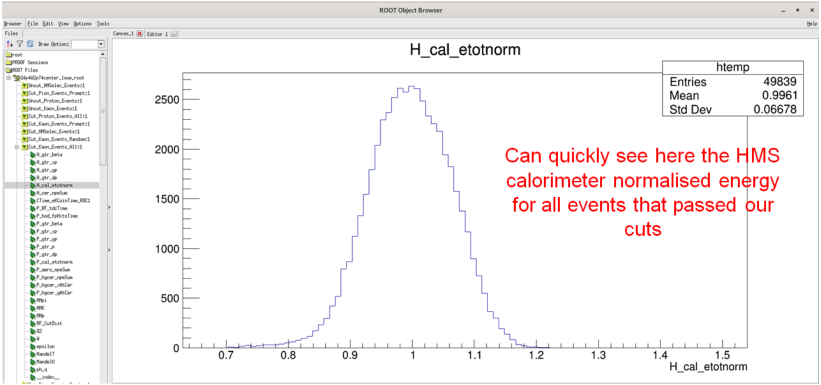
Python Script - Output Example 2/4



Python Script - Output Example 3/4



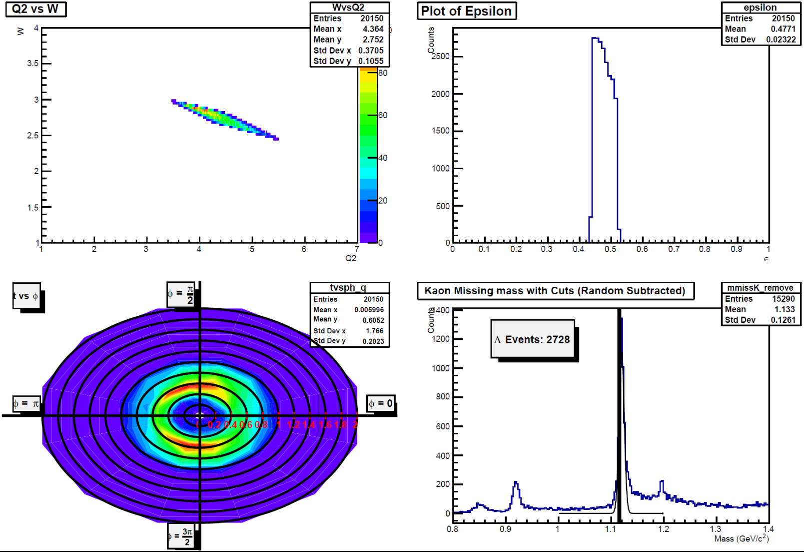
Python Script - Output Example 4/4



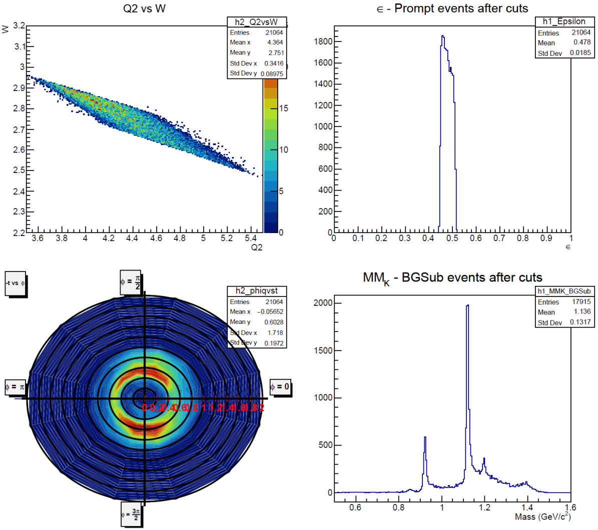
Plotting the Output

- Typically want to plot some specific things and fit them next
- As script doesn't do everything in one go, how the user chooses to plot and fit things is left quite open
  - In addition to saving a root file, the data *can* also be saved as a csv if this is more useful for use in python based plotting/fitting modules
  - .csv output is disabled by default in my example script
- As a demonstration, quickly made a short root based macro to plot some Kaon info
  - Designed it to produce similar output to our online plots for comparison

$Q^2 = 4.4$, $W = 2.74$ Central setting - Online

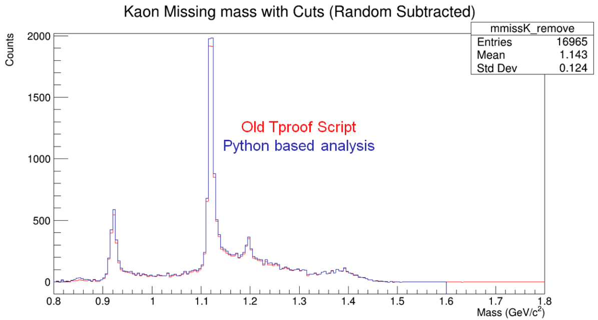


$Q^2 = 4.4$, $W = 2.74$ Central setting - New Output



$Q^2 = 4.4$, $W = 2.74$ Central setting - Further Comparisons

- Fairer comparison would be to a very recent run of the TProof script on the same replay files
- Shown below is the BG subtracted missing mass in such a case
- Note, there are minor changes in the cut ranges which are why there is a slight difference in the totals



Repository and Script Info

- All of the new scripts are included in the Offline branch of the UTIL_KAONLT repository
  - https://github.com/JeffersonLab/UTIL_KAONLT
- The scripts discussed and demonstrated here are in
  - scripts/kaonyield
- The main python script is under the src directory
  - Kaonyield.py
- A README with instructions for running is provided
- Note, some minor tweaks and modifications are still being made

Quick Use Case

RF Timing - Overview

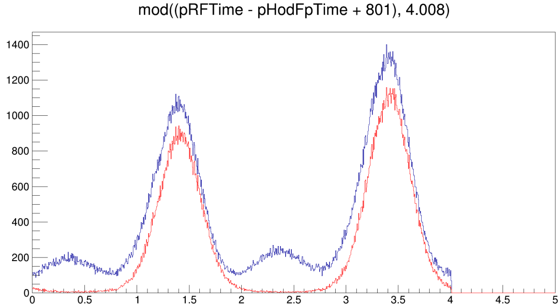
- From discussion with Peter Bosted and Hem, need to take difference between RF time and hodoscope start time
- Need to add an offset this difference, then take modulo
  - Take mod 4.008 → from bunch spacing for this run set
  - Offset varies by run and by beam conditions, a value between 0 and 4.008
- Value plotted as time difference is -

$$\mathrm{fmod}(\mathrm{P.hod.fpHitsTime[0]} - \mathrm{T.coin.pRF\_tdcTime} + \textcolor{red}{\mathrm{offset}}, 4.008)$$

- The offset needed can shift quite a bit
  - For example, MCC switching the beam bucket we get causes a shift
- Applying the same offset value and not accounting for this leads to an odd double peaked plot

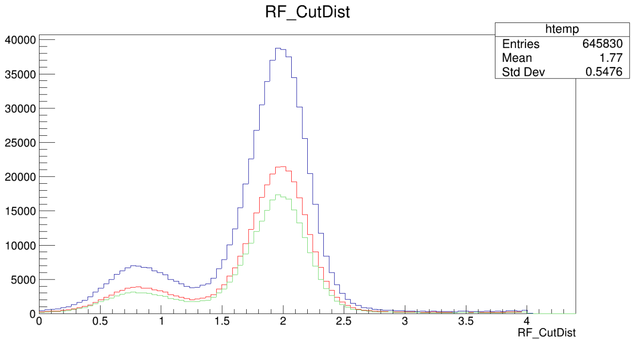
RF Timing Example

- RF time differences, after common cuts, shown in blue
- Events with pion PID cuts applied shown in red
- Without accounting for the change in beam bucket, clearly see the weird double peaking



RF Timing Corrected

- New method of reading cut values means this can easily be accounted for
- Offset chosen to centre the distribution at ~ 2
- Combined events, events before the MCC change, events after the MCC change



Backup Zone

Plotting the Output - Additional Info

- I glossed over a little of how exactly I processed the setting I demonstrated earlier
- I created a short shell script which runs both scripts for a provided run number
  - I.e it runs the python script, then runs the root macro on the output
- I then made a further shell script which simply checks a list of runs (corresponding to the runs in the setting) and executes the previous script for all those runs
  - It then just hadds the result at the end and executes the plotting macro again
- See - https://github.com/sjdkay/UTIL_KAONLT/blob/offline/scripts/kaonyield/Analyse_Kaons.sh and - https://github.com/sjdkay/UTIL_KAONLT/blob/offline/scripts/kaonyield/kinematics/Q4p4W2p74center_lowe_Pt1.sh respectively