Please run these checks

On a MINERvA GPVM

```
$ source /cvmfs/minerva.opensciencegrid.org/minerva/NUISANCE_080117/nuisance/v2r6/builds/genie2126-nuwrov11qrw/Linux/setup.sh
$ nuiscomp -h
$ ls /minerva/data/users/jstowell/NUISTUTORIAL/MC/*/
```

On a Non-MINERvA GPVM

```
$ source /cvmfs/minerva.opensciencegrid.org/minerva/NUISANCE_080117/nuisance/v2r6/builds/genie2126-nuwrov11qrw/Linux/setup.sh
$ nuiscomp -h
```

Download Files (Non-MINERvA GPVM or not on FNAL machine)

```
$ git clone https://github.com/NUISANCEMC/nuisance_tutorial.git
$ cd nuisance_tutorial/mcfiles/
$ source downloadfiles.sh ALL
```

NUISANCE Tutorial

Comparing Generators

Patrick Stowell, Luke Pickering,
Clarence Wret, Callum Wilkinson

31/08/17

Workshop Aims

Few different ways you can use NUISANCE. Today we will cover

1. Using NUISANCE to compare different MCs and data.
2. Performing simple model tunings, placing a constraint on free parameters in GENIE ReWeight

In the next session (September 7th)

3. Adding a new dataset to NUISANCE.

Workshop Layout

- Three different applications we want to cover today.
- Have a list of problems to solve for each application.

Excercises:

- Will go through the first simple problem on the screen for each session.
- Then will have a break whilst you work through the rest.
- Some of you have used NUISANCE before, so I've included a few different problems.
- Don't worry if you haven't finished all of them when we move onto the next section.
- Solutions are in our repo (but try not to cheat!)

```
$ git clone https://github.com/NUISANCEMC/nuisance_tutorial.git
```

Fair Warning

- Model building is hard! Not all datasets agree with each other, and models sometimes don't have the freedom you want.
- NUISANCE tries to help, but this business is a tricky one.
- Need as much data and manpower as possible!

- Join our slack channel to discuss anything neutrino generator related or message developers informally.

https://nuisance-xsec.slack.com/

- I'm based on the 10^{th} floor in Wilson Hall, next to Dan Ruterbories desk, until $\sim 16^{th}$ October. Come find me if you have any issues!

Setting up NUISANCE

- On the FNAL machines NUISANCE can be setup using the following commands:

```
$ export NUPSBASE=/cvmfs/minerva.opensciencegrid.org/minerva/
$ export NUISANCE=$NUPSBASE/NUISANCE_080117/nuisance/v2r6/
$ source $NUISANCE/builds/genie2126-nuwrov11qrw/Linux/setup.sh
[INFO]: Adding NuWro library paths to the environment.
[INFO]: Adding PYTHIA6 library paths to the environment.
[INFO]: Adding GENIE paths to the environment.
```

- On a non-FNAL machine the setup script should be located in your build area under the install folder

```
$ cd /path/to/nuisance/build/folder/
$ source $(uname)/Linux/setup.sh
```

- Run the following command to check it works!

```
$ nuiscomp -h
# Running nuiscomp with a  card file
########################################
```

MC Files

- Event files have been saved into the /minerva/data/ area.
- Check you can access them!

```
$ ls /minerva/data/users/jstowell/NUISTUTORIAL/MC/*/
```

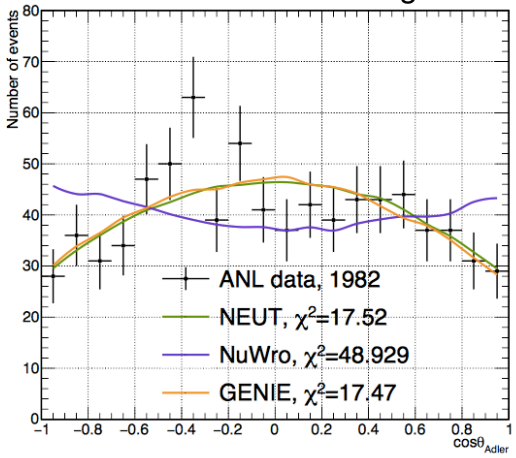
- Anyone not working on the MINERvA FNAL machines will need to download their own copies.

```
$ git clone https://github.com/NUISANCEMC/nuisance_tutorial.git
$ cd nuisance_tutorial/mcfiles/
$ source downloadfiles.sh ALL
```

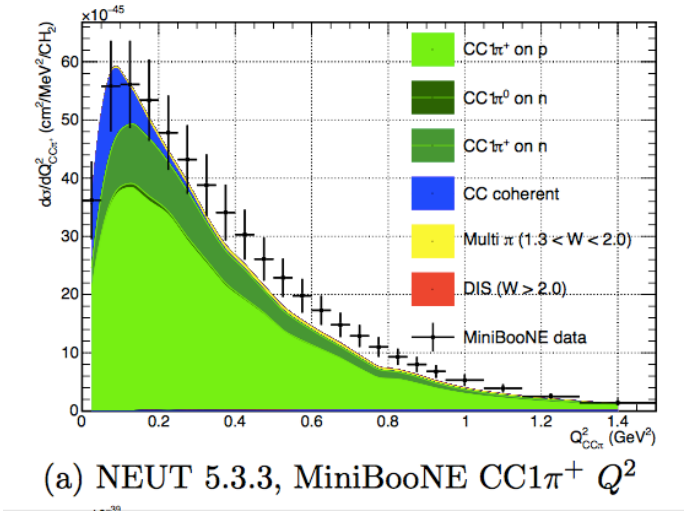
NUISANCE Introduction

- Developed on T2K through efforts to tune NEUT to external cross-section data.

Generators vs Adler Angler

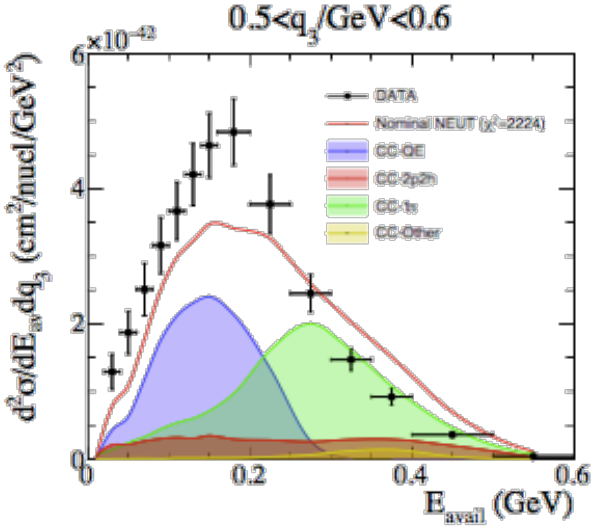


Clarence Wret



(a) NEUT 5.3.3, MiniBooNE CC1π^+ Q^2

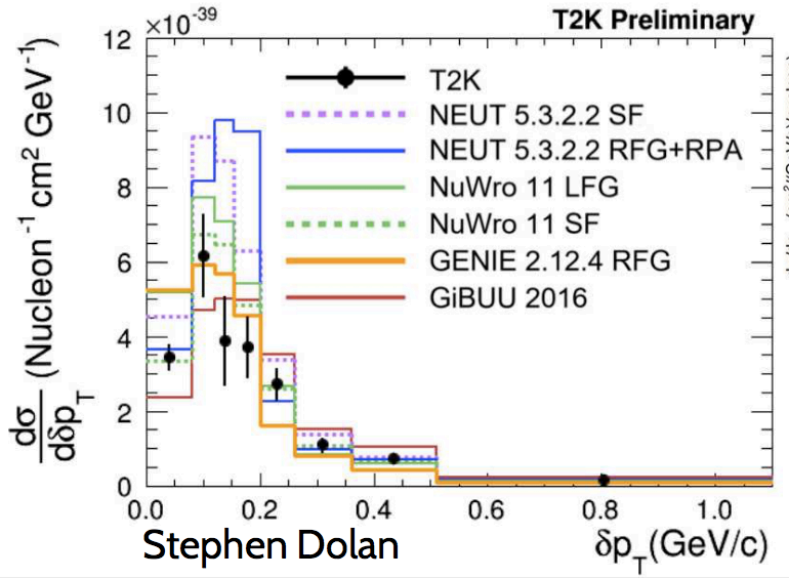
NEUT vs MINERvA Low Recoil

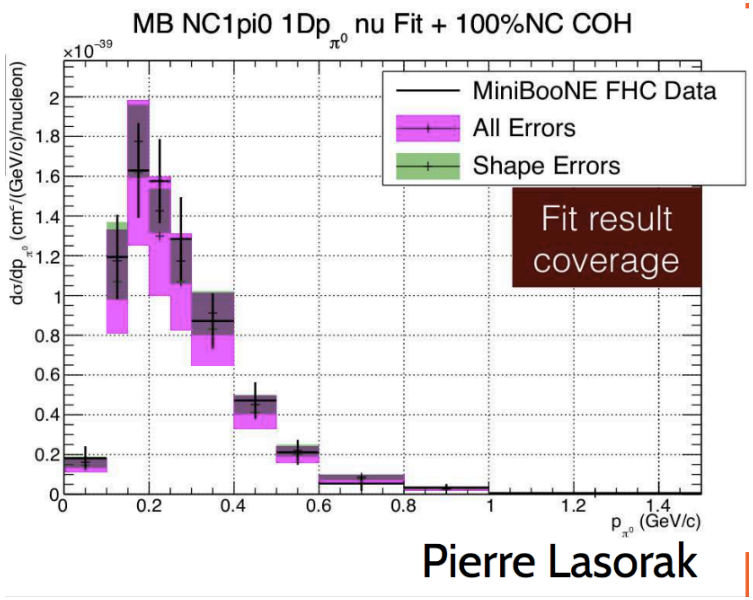


- Supports multiple generators and their reweight engines.
- Broad range of cross-section data to tune reweight parameters.

NUISANCE Introduction

- **Relevant for cross-section analysers, not just model builders.**
  - Dominated by interactions you don't have a sideband for? Get constraints from external data!
  - Easily look at effects of free model parameters on kinematic distributions.
  - Adding your data into our database after release means people can easily use it how you like.





What is inside NUISANCE?

1. Generator Convertors
   - Direct interface with generators and reweight engines.

2. Measurement Routines
   - Signal Selections
   - Data/MC Comparisons
   - Likelihood Calculations

3. Model Tuning+Systematics
   - Top-level analysis routines.
   - Interface with Migrad.

Comparisons

Generator vs Generator

Event Convertors (InputHandler)

- At the core of NUISANCE is a set of routines that convert generators into a common format.
- Designed with final state particle analysis routines in mind.

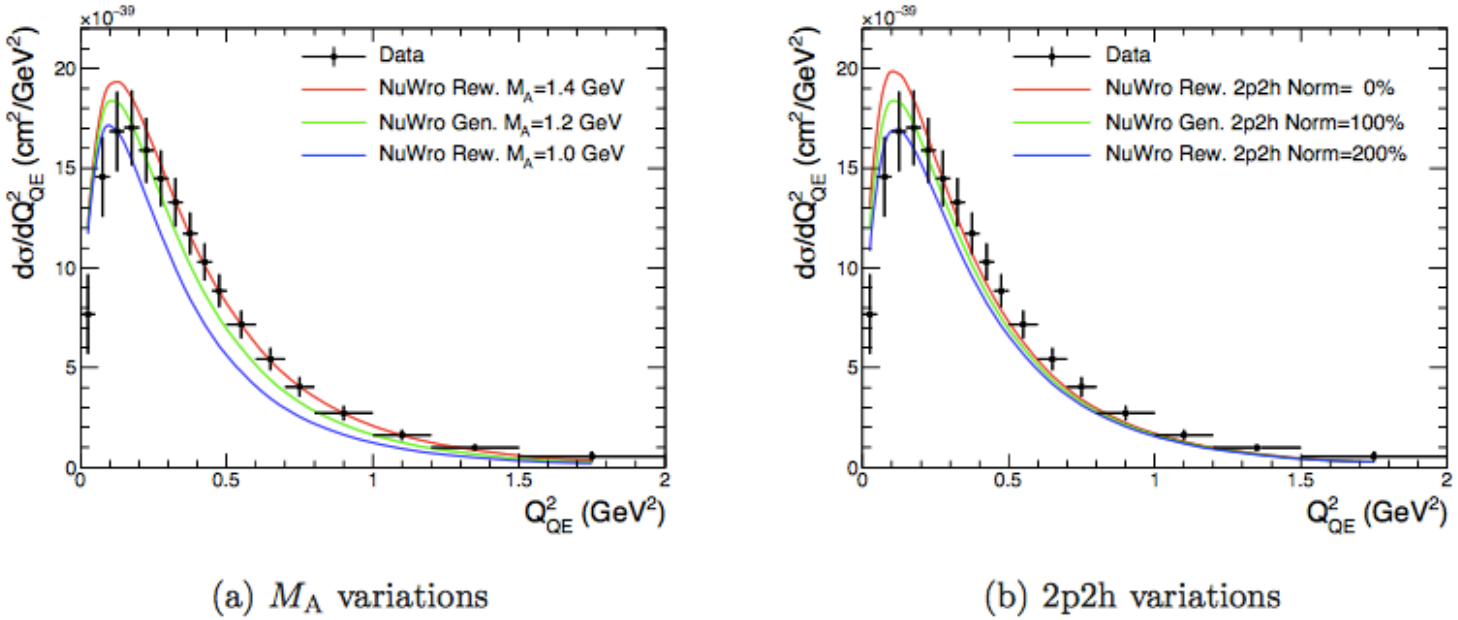
GeneratorEvent **InputHandlers** **Simple FitEvent**

- FitEvent class is a common generator event wrapper

Event ReWeighting (FitWeight)

- FitEvent has original generator event that it was derived from enabling ReWeight (RW) support.

Model Variations in NuWro ReWeight



(a) M_A variations

(b) 2p2h variations

- **FitWeight class is a common RW engine wrapper**
  - Generator RW (e.g. NEUT, NuWro, GENIE)
  - Experimental RW (e.g. T2KReWeight)

Common Formats

- FitEvent/FitWeight class design tries to keep any event analysis that happens above the core InputHandlers as generator independent as possible.

- Cross-section data shouldn't depend on the generator. Method we use to compare MC events to the data also shouldn't.

- **Design provides two advantages:**
  - Any new analysis automatically supports all generators.
  - Any new generator automatically supports all analyses.

Excercises : Generator Comparisons

1. Plot the event spectrum as a function of Neutrino energy (Enu_true) for GENIE Default events.

2. Produce a differential cross-section in ELep for GENIE Default (hint: use fScaleFactor and divide by bin width)

3. Compare the pi+ energy (EPiP) spectrum between GENIE's ValenciaQEBergerSehgalCOHRES and Default models.

4. Plot CC0pi Q2_QE distributions for the models below:
   - GENIE Default
   - GENIE ValenciaQEBergerSehgalCOHRES
   - NuWro LocalFGNievesQEAndMEC

Requires building NUISANCE against NuWro!

nuisflat

- Simplest NUISANCE application "nuisflat" takes an event input and converts it into a simpler TTree format.

- Very similar to GENIE's gntpc convertor.

- Given the path to a NUISANCE-ready MC file of a given "INTYPE" it can be ran using the following:

```
$ nuisflat -i INTYPE:/path/to/inputfile.root \
           -o output.root \
           -f GenericFlux \
         [ -n NEVENTS ]
```

NUISANCE Ready?

- NUISANCE needs flux and cross-section histograms to normalize events to the correct rate

$$R(E_\nu) = \Phi(E_\nu) \times \sigma(E_\nu) \times T$$

N-Targets

Predicted rate given the flux

Flux

Total Xsec spline

- Standard gevgen doesn't save this in the exact format we need.

- Have custom NUISANCE applications that can generate/prepare events with this information (see BACKUPS)

- Time consuming to make events so won't cover this today.

NUISANCE Ready

- Have placed prepared MC files here:

```
/minerva/data/users/jstowell/NUISTUTORIAL/MC/genie/
```

```
/minerva/data/users/jstowell/NUISTUTORIAL/MC/nuwro/
```

- Good idea to symbolic link these folders to working area

```
$ ln -s /minerva/data/users/jstowell/NUISTUTORIAL/MC/*
```

- We also keep MC on our website, if you are ever in need of some MC events to run NUISANCE with!

Models

- There are a number of models for you to choose from in those MC folders. I'll refer to each model by its tag in the exercises.

```
$ ls ./genie/ ./nuwro/
gntp.DefaultPlusMECWithNC.MINERvA_fhc_numu.CH.2500000.ghep.root
```

- Format: **GENERATOR.TAG.FLUX.TARGET.NEVENTS.root**

Tag (model)	Notes
gntp.Default	GENIE 2.12.6 Default
gntp.DefaultPlusMECWithNC	GENIE Default+Empirical 2p2h
gntp.ValenciaQEBergerSehgalCOHRES	GENIE LFG+NievesQE/2p2h+BergerSehgal
nuwroev.LocalFGNievesQEAndMEC	NuWro Default+LFG+Nieves RPA/2p2h

Input Types

- We have our event sample, now we just have to tell NUISANCE what type it is when loading them in.

- Input Format: `"FILETYPE:/path/to/eventfile.root"`

- Format is the same for all applications as this string is passed to the InputHandler creator.
- Uses `FILETYPE` to figure out what InputHandler to create.

- Some possible `FILETYPE`s:

```
GENIE:/path/geniefile.root
NUWRO:/path/nuwrofile.root
NEUT:/path/neutfile.root
GiBUU:/path/gibuufile.root
FEVENT:/path/fitevent.root
```

Problem 1

1. Plot the event spectrum as a function of Neutrino energy for GENIE Default events.

- Can generate nuisflat output by running the following command

```
$ nuisflat \
  -i GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root \
  -o genie.Default.flat.root \
  -f GenericFlux \
  -n 50000
```

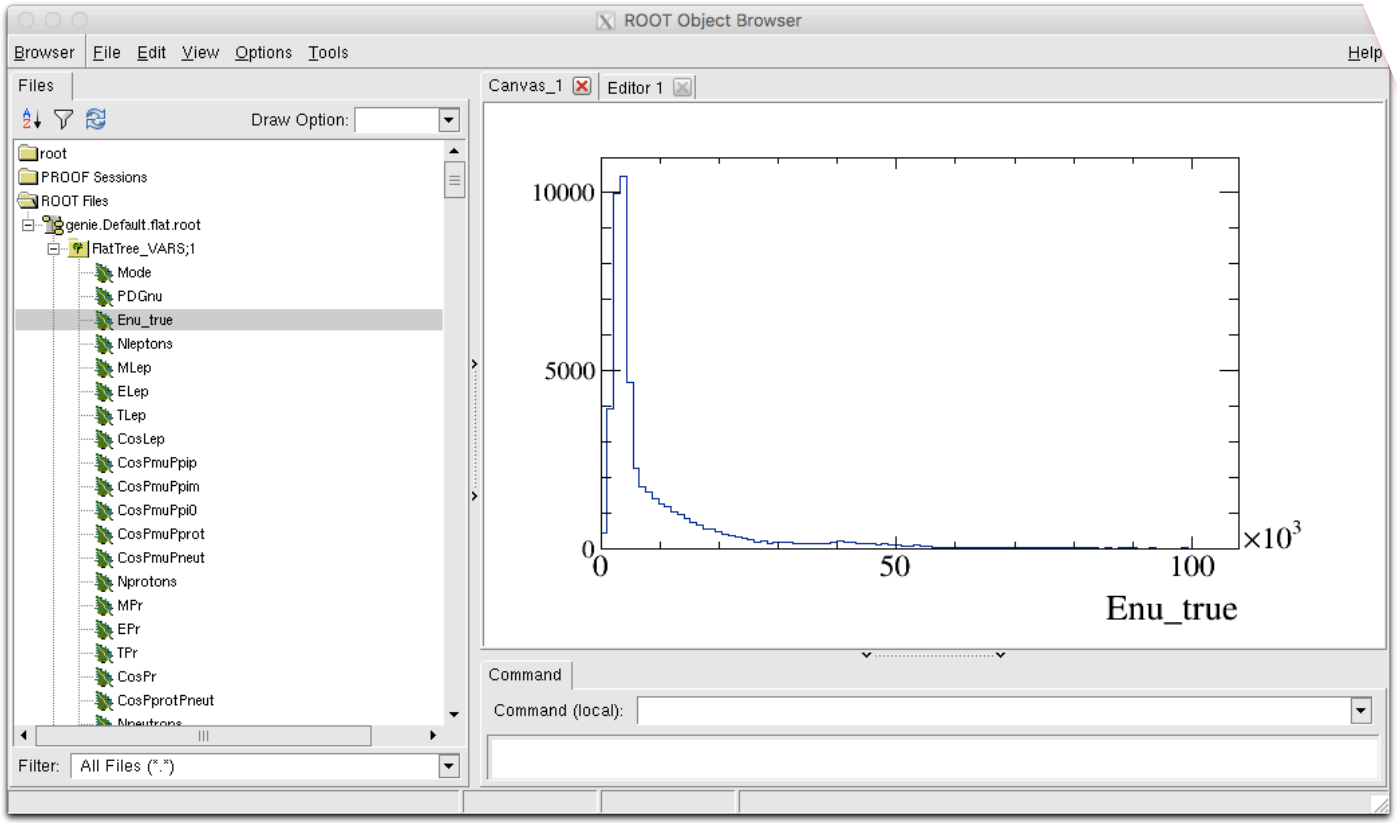
All applications let you restrict the total MC events processed using the (-n) argument.

For examples today we will be restricting the number of events to save time.

Problem 1

```
[jstowell@minervagpvm03 MC]$ root genie.Default.flat.root
root [0] Attaching file genie.Default.flat.root as _file0…
root [1] TBrowser b
root [2] FlatTree_VARS->Draw("ELep", "fScaleFactor");
```

Hint for problem 2.



Excercises : Generator Comparisons

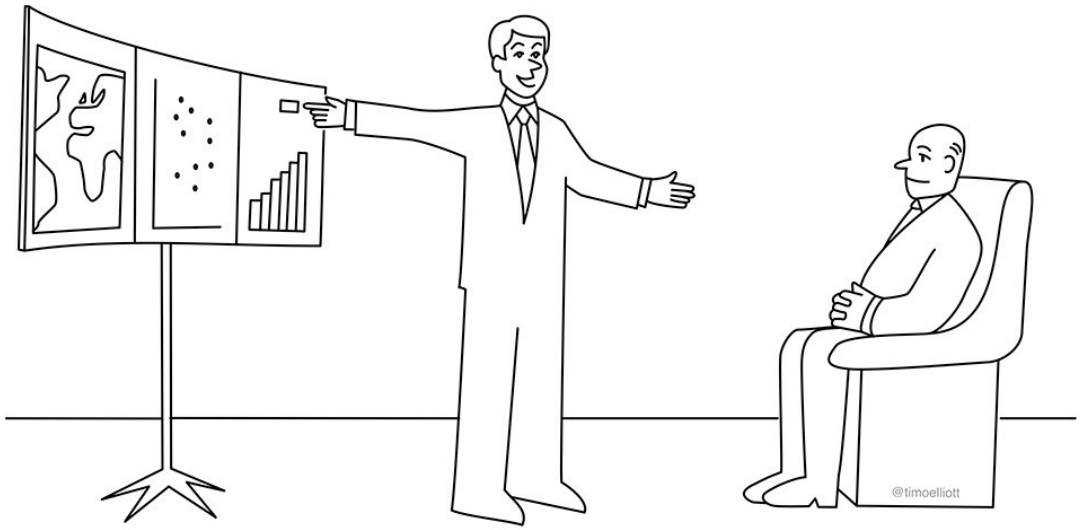
1. Plot the event spectrum as a function of Neutrino energy (Enu_true) for GENIE Default events.

2. Produce a differential cross-section in ELep for GENIE Default (hint: use fScaleFactor and divide by bin width)

3. Compare the pi+ energy (EPiP) spectrum between GENIE's ValenciaQEBergerSehgalCOHRES and Default models.

4. Plot CC0pi Q2_QE distributions for the models below:
   - GENIE Default
   - GENIE ValenciaQEBergerSehgalCOHRES
   - NuWro LocalFGNievesQEAndMEC

Requires building NUISANCE against NuWro!



"And our unique JustifyIt™ feature uses deep learning to find data that agrees with your point of view!"

Comparisons

Generator vs Data

Data Comparison Classes

- Above the convertors are the measurement classes.
- Set of analysis classes that loop over a collection of FitEvent's and generate distributions.

- **These are called "samples" inside NUISANCE.**

```
$ ls $NUISANCE/src/MINERvA/

MINERvA_CCQE_XSec_1DQ2_nu.cxx
MINERvA_CCQE_XSec_1DQ2_nu
MINERvA_CCinc_XSec_1DEnu_nu.cxx
MINERvA_CCinc_XSec_1DEnu_nu.h
```

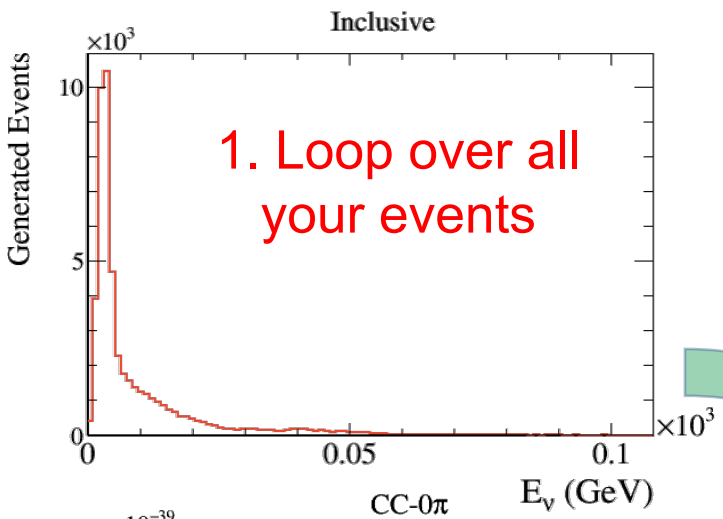
Usually one class implementation for each sample

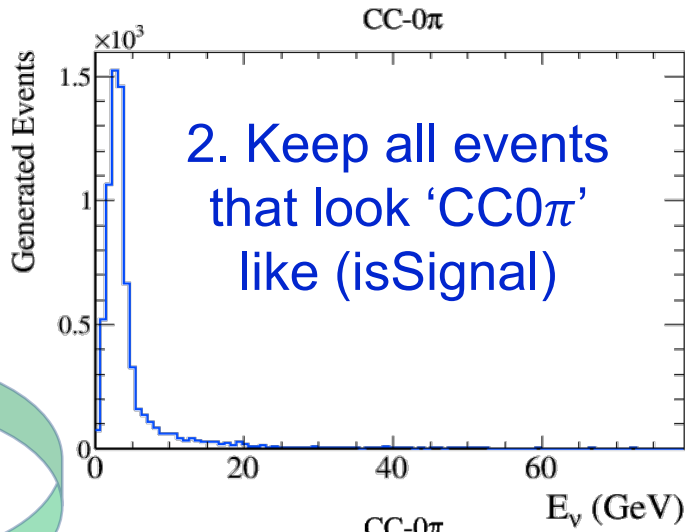
- Each sample inherits from a base "Measurement" class containing useful functions.
- Base classes help automate the processing chain.

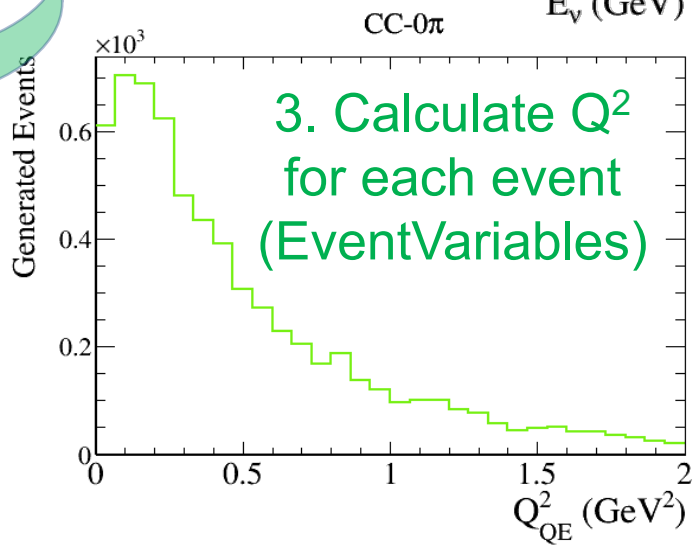
Analysis Process

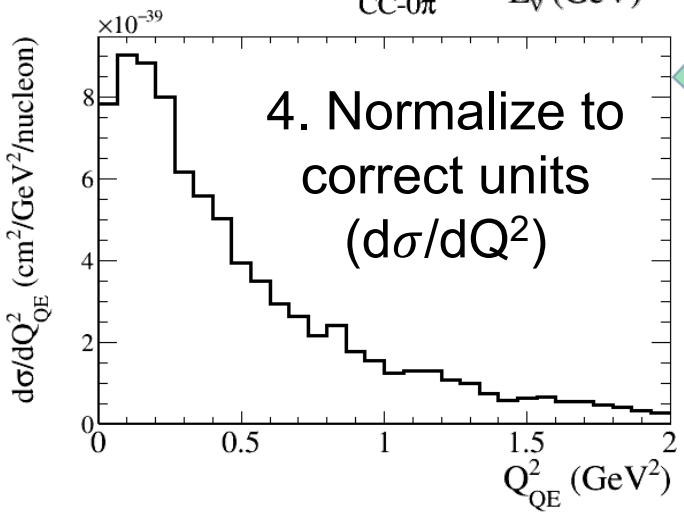
- Samples function in a similar way to a real cross-section (without detector/systematics...)

E.g. CC0π 1DQ^2_{QE} Distribution









Comparisons

- Event processing requirements (isSignal/EventVariables) on the previous slide are implemented in the sample class.

- Samples can then be called easily by string, and passed an input file in a similar way to how we passed files to nuisflat.

```
<nuisance>

    <!-- Samples -->
    <sample name="MINERvA_CCQE_XSec_1DQ2_nu" input="GENIE:@GENIE_DIR/gntp.CH.root" />
    <sample name="MINERvA_CC1pip_XSec_1DTpi_nu" input="GENIE:@GENIE_DIR/gntp.CH.root" />
    <sample name="MINERvA_CC1pip_XSec_1Dth_nu" input="GENIE:@GENIE_DIR/gntp.CH.root" />

</nuisance>
```

- Adding your own sample that can be called here will be covered in the next workshop!

Excercises : Data Comparisons

1. Compare GENIE Default to MINERvA CCQE 1DQ2 data.
   (sample : MINERvA_CCQE_Xsec_1DQ2_nu)

2. Compare the GENIE Default to MINERvA CC1pip 1DTpi+1Dth data with M_A^{RES} set to -1.0 sigma.
   (sample : MINERvA_CC1pip_XSec_1DTpi(th)_nu)
   (dial name : "MaCCRES" dial type: "genie_parameter")

3. Compare GENIE ValenciaQEBergerSehgalCOHRES and Default models to MINERvA CC1pip 1Dth data.
   (sample : MINERvA_CC1pip_XSec_1Dth_nu)

4. Compare the NuWro LocalFGNievesQE and GENIE ValenciaQEBergerSehgalCOHRES models to MINERvA Low Recoil Data *(sample : MINERvA_CCinc_XSec_2DEavq3_nu)*

Running comparison

- Comparison application : **nuiscomp**

- nuiscomp requires you to write a card file to tell you what comparison you want it to create.

```
nuiscomp -c cardfile.xml -o output.root
```

cardfile.xml

```
<nuisance>
    <config GENIE_DIR="/path/to/my/genie/events/" />
    <parameter type="genie_parameter" name="MaCCQE" nominal="1.0" state="FIX" />
    <sample name="MINERvA_CCQE_XSec_1DQ2_nu" input="GENIE:@GENIE_DIR/gntp.CH.root" />
</nuisance>
```

- Cardfile is just a xml file listing the parameters and samples we want and the input files for each one.

Writing a card file

```
<nuisance>
  <!-- List of Samples -->

</nuisance>
```

- Open a new file in a text editor ***"samplecard.xml"***
- XML Card files wrapped in a nuisance statement so first we need to add those.
- Comments given as standard XML comments.

Writing a card file

```
<nuisance>
  <!-- List of Samples -->
  <sample name=""
          input="" />
</nuisance>
```

- To tell NUISANCE to load a new sample we need to include a "sample" XML structure.

```
<sample name="NAMEDEF"  input="INPUTDEF" />
```

Required Keys:

- "NAMEDEF" = Name of the sample to load
- "INPUTDEF" = Input MC File Information

Same format as nuisflat inputs!

Finding a sample

- Want to compare to MINERvA CCQE data.
- "nuissamples" script provided to search for sample names that can be used.

```
$ nuissamples [substring]
```

1st argument is a search substring. If none given, the full sample list is returned.

```
$ nuissamples MINERvA_CCQE
MINERvA_CCQE_XSec_1DQ2_nu
MINERvA_CCQE_XSec_1DQ2_nu_20deg
MINERvA_CCQE_XSec_1DQ2_nu_oldflux
MINERvA_CCQE_XSec_1DQ2_nu_20deg_oldflux
MINERvA_CCQE_XSec_1DQ2_antinu
...
```

This is the one we need!

Format: EXPERIMENT_CHANNEL_TYPE_DISTRIBUTION_EXTRAIDs

Writing a card file

```
<nuisance>
  <!-- List of Samples -->
  <sample name="MINERvA_CCQE_XSec_1DQ2_nu"
          input="" />
</nuisance>
```

- Now we just need to include our input file. Can use the same input files we used for nuisflat. e.g.

```
input="FILETYPE:/path/to/file.root"
```

- Want to compare to GENIE Default:

```
input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root"
```

Writing a card file

```
<nuisance>
  <!-- List of Samples -->
  <sample name="MINERvA_CCQE_XSec_1DQ2_nu"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- Combining everything we should have a cardfile ready.
- We can now run it using

```
$ nuiscomp -c samplecard.xml -o sampleccqe.root -n 100000
```

Again we are using 1E5 events to save time, but proper comparisons should use full event sample.

NUISCOMP Output

```
[LOG Minmzr]:- Getting likelihoods...                          : -2logL
[LOG Minmzr]:- -> MINERvA_CCQE_XSec_1DQ2_nu                    : 17.289/8
[LOG Fitter]: Likelihood for JointFCN:                          17.289
[LOG Fitter]: ------------
[LOG Fitter]: Saving current full FCN predictions
[LOG Minmzr]:- Writing each of the data classes...
[LOG Sample]:-- Written Histograms: MINERvA_CCQE_XSec_1DQ2_nu
[LOG Fitter]: ----------------------------------- -
[LOG Fitter]: Comparison Complete.
[LOG Fitter]: ----------------------------------- -
```

- NUISANCE Automatically calculates you a χ^2/NDOF value for the data/MC comparison.
- Tries to use full covariance information where possible.

- Saves all the histograms produced into our output file.

NUISCOMP Output (2)

- Number of different objects saved into the output file.
- Each one has been prepended with the sample name.
- **Examples**
  - samplename**_data** : Data distribution
  - samplename**_MC** : MC distribution in data binning
  - samplename**_MC_FINE** : MC in fine binning
  - samplename**_MC_SHAPE** : MC normalised to data
  - samplename**_data_ratio** : data/MC ratio
  - samplename**_MODES** : THStack true interaction channels

Hint: Use gPad->BuildLegend() to see all the included true interaction channel labels.

Excercises : Data Comparisons

1. Compare GENIE Default to MINERvA CCQE 1DQ2 data.
   (sample : MINERvA_CCQE_Xsec_1DQ2_nu)

2. Compare the GENIE Default to MINERvA CC1pip 1DTpi+1Dth data with M_A^{RES} set to -1.0 sigma.
   (sample : MINERvA_CC1pip_XSec_1DTpi(th)_nu)
   (dial name : "MaCCRES" dial type: "genie_parameter")

3. Compare GENIE ValenciaQEBergerSehgalCOHRES and Default models to MINERvA CC1pip 1Dth data.
   (sample : MINERvA_CC1pip_XSec_1Dth_nu)

4. Compare the NuWro LocalFGNievesQE and GENIE ValenciaQEBergerSehgalCOHRES models to MINERvA Low Recoil Data *(sample : MINERvA_CCinc_XSec_2DEavq3_nu)*

Multiple Samples

```
<nuisance>
  <!-- List of Samples -->
  <sample name="MINERvA_CC1pip_XSec_1DTpi_nu"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
  <sample name="MINERvA_CC1pip_XSec_1Dth_nu"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- NUISANCE reads events from disk, and then distributes them to relevant sample classes.
- Minimal extra overhead when loading a number of different distributions or datasets from one MC file.

- Just add an extra sample xml entry for every dataset you care about and NUISANCE will load them all at once.

Multiple Sample Output

- Run this joint sample in a similar fashion.

```
$ nuiscomp -c samplecc1pip.xml -o samplecc1pip.root -n 100000
```

- Likelihoods for both samples added uncorrelated, to form a joint total likelihood for the comparison.

```
[LOG Minmzr]:- Getting likelihoods...                    : -2logL
[LOG Minmzr]:- -> MINERvA_CC1pip_XSec_1DTpi_nu           : 44.6792/7
[LOG Minmzr]:- -> MINERvA_CC1pip_XSec_1Dth_nu            : 260.352/13
[LOG Fitter]: Likelihood for JointFCN:                    305.031
```

- Two sets of histograms also now contained in the output file.

ReWeighting Predictions

```
<nuisance>
  <!-- List of parameters -->
  <parameter type="" name="" nominal="" />

  <!-- List of Samples -->
  <sample name="MINERvA_CC1pip_XSec_1DQ2_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
  <sample name="MINERvA_CC1pip_XSec_1Dth_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- Reweight parameters can be added to NUISANCE card files using a "parameter" xml structure.

```
<parameter type="DIAL_TYPE" name="NAME" nominal="DIAL_VALUE" />
```

- **Requirements:**
  - NAME : Name of the dial inside the RW Engine
  - DIAL_TYPE : RW Type (e.g. genie_parameter)
  - DIAL_VALUE : Current Value to use

Finding ReWeight Dials

```
<nuisance>
  <!-- List of parameters -->
  <parameter type="genie_parameter" name="" nominal="" />

  <!-- List of Samples -->
  <sample name="MINERvA_CC1pip_XSec_1DQ2_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
  <sample name="MINERvA_CC1pip_XSec_1Dth_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- Any dial name recognised by the RW engine is supported.
- List is in **$GENIE/src/ReWeight/GSyst.h**

```
$ grep case $GENIE/src/ReWeight/GSyst.h
case ( kXSecTwkDial_MaNCEL          ) : return "MaNCEL";              break;
case ( kXSecTwkDial_EtaNCEL         ) : return "EtaNCEL";             break;
case ( kXSecTwkDial_NormCCQE        ) : return "NormCCQE";            break;
...
```

- We want the Resonant Axial Mass ("MaCCRES")

Adding ReWeight

```
<nuisance>
  <!-- List of parameters -->
  <parameter type="genie_parameter" name="MaCCRES" nominal="" />

  <!-- List of Samples -->
  <sample name="MINERvA_CC1pip_XSec_1DQ2_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
  <sample name="MINERvA_CC1pip_XSec_1Dth_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- Now need to choose our current dial value.

```
nominal="DIAL_VALUE"
```

- Units are whatever the reweight engine uses.
- E.g. GENIE ReWeight usually considers dials in units of "1-sigma" from nominal with 0.0 being the default value

Running nuiscomp

```
<nuisance>
  <!-- List of parameters -->
  <parameter type="genie_parameter" name="MaCCRES" nominal="-1.0" />

  <!-- List of Samples -->
  <sample name="MINERvA_CC1pip_XSec_1DQ2_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
  <sample name="MINERvA_CC1pip_XSec_1Dth_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- **Can rerun our edited card file and save the output somewhere else for later comparisons**

```
$ nuiscomp -c samplecc1pip.xml -o samplecc1pip-rw.root -n 100000
```

- All MC curves have now been weighted with GENIE ReWeight set to MaCCRES = -1

Excercises : Data Comparisons

1. Compare GENIE Default to MINERvA CCQE 1DQ2 data.
   (sample : MINERvA_CCQE_Xsec_1DQ2_nu)

2. Compare the GENIE Default to MINERvA CC1pip 1DTpi+1Dth data with M_A^{RES} set to -1.0 sigma.
   (sample : MINERvA_CC1pip_XSec_1DTpi(th)_nu)
   (dial name : "MaCCRES" dial type: "genie_parameter")

3. Compare GENIE ValenciaQEBergerSehgalCOHRES and Default models to MINERvA CC1pip 1Dth data.
   (sample : MINERvA_CC1pip_XSec_1Dth_nu)

4. Compare the NuWro LocalFGNievesQE and GENIE ValenciaQEBergerSehgalCOHRES models to MINERvA Low Recoil Data *(sample : MINERvA_CCinc_XSec_2DEavq3_nu)*

Tuning Generators

Reweight vs Data

Minimiser Routines

- Final layer of NUISANCE takes the convertor+comparison routines and tries to use it for model tuning.

- Direct interface with ROOT's minimiser libraries.

  (Creating a Minimizer via the Plug-In Manager)

  https://root.cern.ch/numerical-minimization

- Can treat any reweight parameter as free and try to minimise the joint χ^2 value between all samples.

- Mostly automated procedure, just list the datasets you want and the parameters you want to tune and leave it to run.

Excercises : Tuning M_A

1. Tune MaCCRES to MINERvA CC1pip 1DQ2 data

2. Extract a value for MaCCQE when fitting to MiniBooNE CCQE 1DQ2 data with a floating flux norm (HINT: FREE)

3. Tune MaCCRES to MINERvA CC1pip 1DQ2 data with a prior of $M_A = 0.0 \pm 1.0\sigma$

4. Run a joint fit for MaCCQE to MiniBooNE and MINERvA CCQE 1DQ2 data (again treat MiniBooNE as FREE)

NuisMin Application

- Tuning application : **nuismin**

```
$ nuismin -c samplecard.xml \
          -o sampleccqe.root \
         [-n NEVENTS ]
```

- Runs using XML card just like nuiscomp. Can take our previous card files as a starting point.

- Main difference is in nuismin we must specifiy that we want some parameters to be treated as FREE.

Free Parameter Structures

- Free parameter structures are very similar to fixed parameters, but they require you to tell NUISANCE the limits, and what state it is in FIX or FREE.

```
<parameter type="genie_parameter" name="MaCCRES"
           nominal="" low="" high="" step="" state="" />
```

Requirements:

- Nominal = Current (Starting) Value
- Low = Lower Limit
- High = Upper Limit
- Step = Migrad Starting Step Size
- State = Parameter state : FIX or FREE

Multiple Parameters

- NUISANCE will take all "FREE" parameters and load them into a multi-dimensional ROOT minimisation.

```
<parameter type="genie_parameter" name="MaCCRES"
            nominal="1.0" low="-3.0" high="3.0" step="1.0" state="FREE" />
<parameter type="genie_parameter" name="MaCCQE"
            nominal="1.0" low="-3.0" high="3.0" step="1.0" state="FREE" />
```

- Possible to keep some parameters fixed at nominal in the fit by instead putting the state to "FIX"

```
<parameter type="genie_parameter" name="MaCCRES"
            nominal="1.0" low="-3.0" high="3.0" step="1.0" state="FREE" />
<parameter type="genie_parameter" name="MaCCQE"
            nominal="1.0" low="-3.0" high="3.0" step="1.0" state="FIX" />
```

Tuning MaCCRES

```
<nuisance>
  <!-- List of parameters -->
  <parameter type="genie_parameter" name="MaCCRES"
             nominal="0.0" low="-3.0" high="3.0" step="1.0" state="FREE" />

  <!-- List of Samples -->
  <sample name="MINERvA_CC1pip_XSec_1DQ2_nu_2017"
          input="GENIE:genie/gntp.Default.MINERvA_fhc_numu.CH.2500000.root" />
</nuisance>
```

- We just want to float MaCCRES freely so we change our new parameter line to treat it as FREE.

- This card file will vary MaCCRES between $\pm 3\sigma$ and find the best fit it can to the 1DQ2 dataset.

Making a nominal plot

- Before we start tuning, we can use this new card file to create a nominal prediction for later comparisons.

- Running **nuiscomp** first over your new minimisercard.xml will produce the MC output at your starting values.

```
$ nuiscomp -c minimisercard.xml \
           -o minimiser-nominal.root \
           -n 100000
```

- This is a good practice as it also lets you check if all your samples are setup correctly before running a long fit.

Running

- Once you've written your card file it can be run in the same fashion but using nuismin this time.

```
$ nuismin -c minimisercard.xml \
          -o minimiser-tuned.root \
          -n 100000
```

- Minuit will scan the parameter space and try to find best fit.

```
[LOG Reconf]:---  -> Par 0. MaCCRES 0
[LOG Reconf]:--- Starting Reconfigure iter. 0
[LOG Minmzr]:- -> MINERvA_CC1pip_XSec_1DQ2_nu_2017          : 21.4312/8
...
[LOG Reconf]:---  -> Par 0. MaCCRES 0.0101951
[LOG Reconf]:--- Starting Reconfigure iter. 1
[LOG Minmzr]:- -> MINERvA_CC1pip_XSec_1DQ2_nu_2017          : 21.4962/8
```

Long Iterations

- Fits with multiple parameter scan take on the order of a day.

- NUISANCE is very I/O heavy. Have to read the full MC event so that it can be passed to the RW engine.
- 90% of each event loop is just reading events from disk. ☹

- Event loop optimized for multiple samples (each event is read from disk only once per event loop)

- Additional config flag (SignalReconfigures) can speed it up further by looping over only signal events after the first pass.

```
$ nuismin [ options ]  -q SignalReconfigures=1
```

Running Faster

```
$ nuismin -c minimisercard.xml \
          -o minimiser-tuned.root \
          -n 100000 \
          -q SignalReconfigures=1
```

- Turning on SignalReconfigures will speed things up quite a bit.

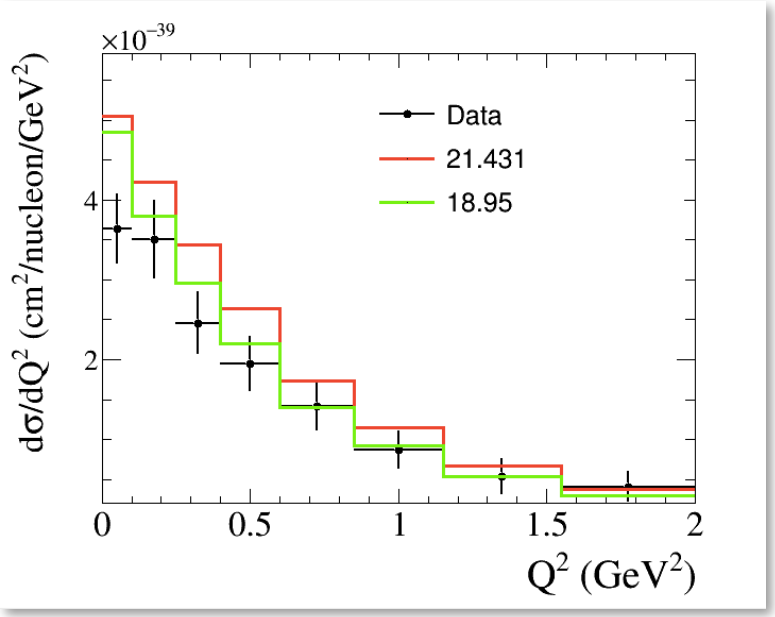
```
[LOG Minmzr]:- Finished Reconfigure iter. 4 in 3s
[LOG Minmzr]:- Getting likelihoods...                    : -2logL
[LOG Minmzr]:- -> MINERvA_CC1pip_XSec_1DQ2_nu_2017       : 21.4239/8
```

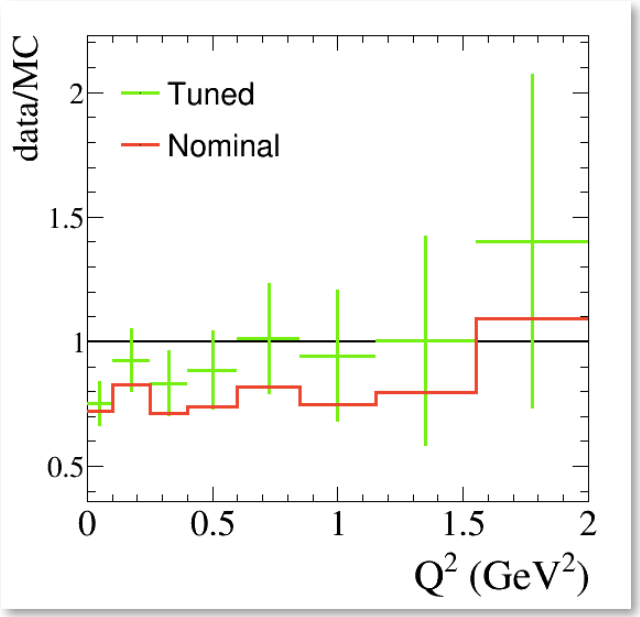
- Option still not fully validated with all samples so I advise caution for now!

Minimiser Output

```
#    Parameter  = Value      +- Error      (Units)  Conv. Val  +- Conv. Err  (Units)
0  . MaCCRES    = -0.789184  +- 0.510301   (sig.)   -0.789184  +- 0.510301   (sig.)
```

- **nuismin** saves the same MC histograms as **nuiscomp** did.



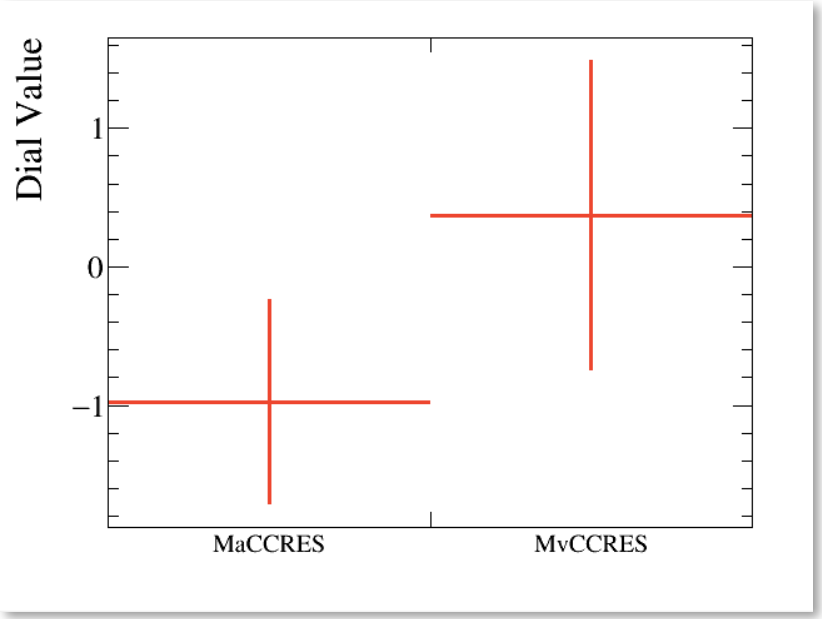


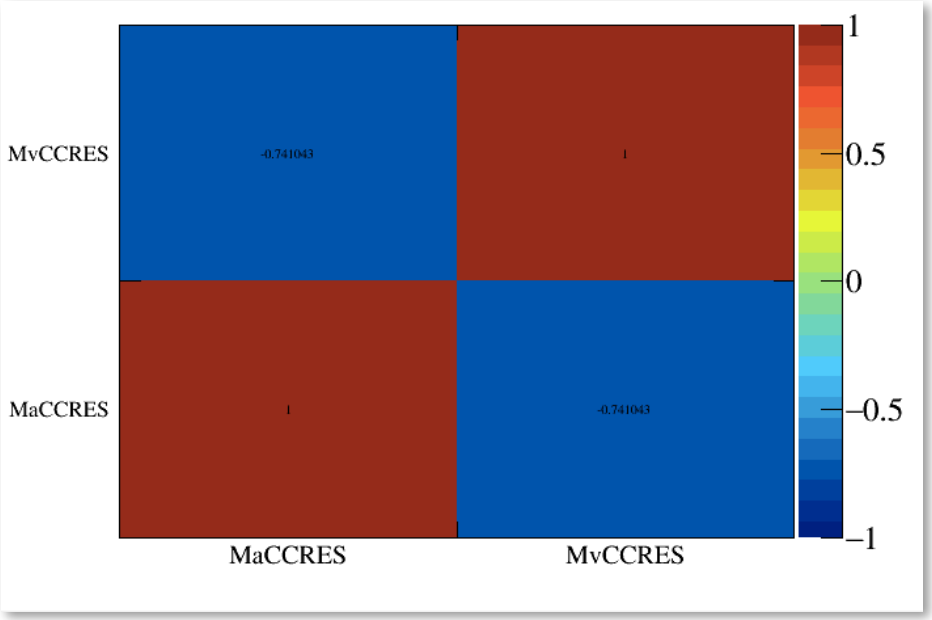
- Can compare our "nominal" and "tuned" results

```
$ root minimiser-tuned.root minimiser-nominal.root
```

Minimiser Output (2)

- Lots of other fit information saved into the output file.





- Some interesting ones:
  - **start_dials** : starting parameters
  - **fit_dials** : best fit parameters
  - **fit_iterations** : Parameter/Likelihood state at each iter.
  - **fit_result** : best fit parameter result (CHI2 = best joint fit likelihood)

Excercises : Tuning M_A

1. Tune MaCCRES to MINERvA CC1pip 1DQ2 data

2. Extract a value for MaCCQE when fitting to MiniBooNE CCQE 1DQ2 data with a floating flux norm (HINT: FREE)

3. Tune MaCCRES to MINERvA CC1pip 1DQ2 data with a prior of $M_A = 0.0 \pm 1.0\sigma$

4. Run a joint fit for MaCCQE to MiniBooNE and MINERvA CCQE 1DQ2 data (again treat MiniBooNE as FREE)

Fit Types

- Extra handling options can be passed to each sample through the optional "type" field.

```
<sample name="NAME"    type="TYPEDEF"    input="INPUT" />
```

- Usually handles likelihood options. Many non-conflicting terms can be passed at once (e.g. DIAG/FREE/NORM).

- **A few examples for TYPEDEF:**
  - DIAG : Use diagonal errors instead of a covariance
  - SHAPE : Treat as a shape-only likelihood
  - FREE : Freely float the normalisation as a fit parameter

Changing the Routines

- Minimiser interface has a few different minimiser routines.

```
Brute
Simplex
Minuit
Combined
Fumili
```

```
ConjugateFR
ConjugatePR
BFGS
BFGS2
SteppDesc
```

```
GSLSimAn
```

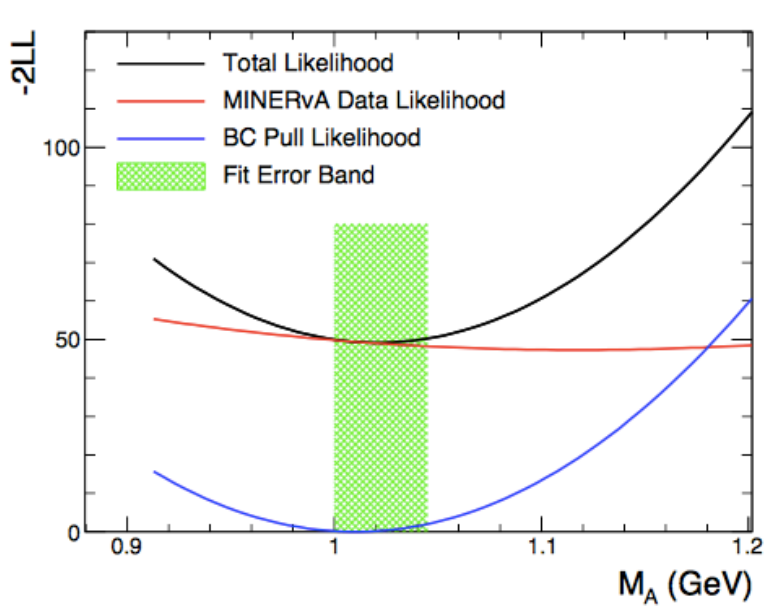
- Can use the (-f) flag in nuismin to choose which routine to run.

- Routines are comma seperated and ran in sequence, with the results of one routine being passed into the other.
- **Example** : run a brute force ND scan, then run Minuit from the new starting point.

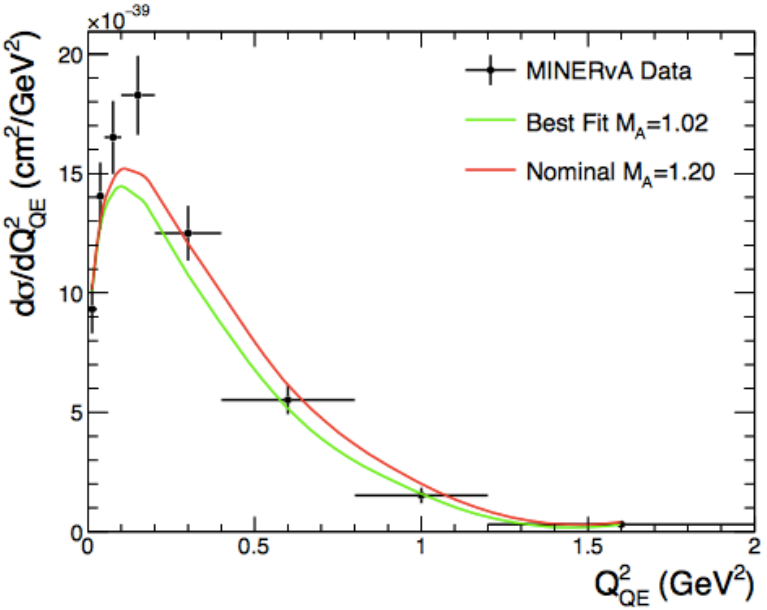
```
$ nuismin -c card.xml -o out.root -f Brute,Minuit
```

Running with simple priors

- Can include χ^2 penalty terms in fit using "covar" objects.



(a) χ^2 scan across the M_A parameter space



(b) Nominal and best fit MINERvA distribution

- Allows you to use the results of previous internal/external fits as an additional constraint in your tuning.

Simple Priors : DIAL

- Simplest "covar" type is an uncorrelated Gaussian **DIAL** pull.

- If you already have a parameter included in your card file called PARNAME you can place a Gaussian prior on it with:

```
<covar name="PARNAME_prior"
       input="DIAL:PARNAME;CENTRAL_VALUE;ERROR_VALUE"
       type="GAUSPULL" />
```

- **Example:** Float MaCCRES, but include an additional constraint of MaCCRES = -0.5 +- 1.0 sigma in the fit.

```
<parameter name="MaCCRES" type="genie_parameter"
           nominal="0.0" low="-3.0" high="3.0" step="1.0" state="FREE" />
<covar name="MaCCRES_prior" input="DIAL:MaCCRES;-0.5;1.0" type="GAUSPULL" />
```

Simple Priors : ROOT

- Can also specify similar Gaussian priors with full correlations between parameters.

- Requires TH1D central values (TH1DCV) and TH2D covariance (TH2DCOV) with the bin labels set to match the dial names.

```
<covar name="ID" input="ROOT:FILEPATH;TH1DCV;TH2DCOV" type="GAUSPULL" />
```

- Exact histogram format that NUISANCE saves outputs, so can easily use a previous fit result as a future prior.

```
<parameter name="MaCCRES" type="genie_parameter"
           nominal="0.0" low="-3.0" high="3.0" step="1.0" state="FREE" />
<parameter name="NonResBkgvnCC1pi" type="genie_parameter"
           nominal="0.0" low="-3.0" high="3.0" step="1.0" state="FREE" />
<covar name="fit_prior" input="ROOT:result.root;fit_dials;covariance" type="GAUSPULL" />
```

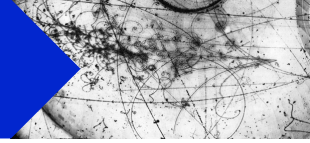
Excercises : Tuning M_A

1. Tune MaCCRES to MINERvA CC1pip 1DQ2 data

2. Extract a value for MaCCQE when fitting to MiniBooNE CCQE 1DQ2 data with a floating flux norm (HINT: FREE)

3. Tune MaCCRES to MINERvA CC1pip 1DQ2 data with a prior of $M_A = 0.0 \pm 1.0\sigma$

4. Run a joint fit for MaCCQE to MiniBooNE and MINERvA CCQE 1DQ2 data (again treat MiniBooNE as FREE)



Backup

Extra Methods

gevgen_nuisance

- Custom gevgen app can be used to generate GENIE events specially for NUISANCE.
- Automatically saves required histograms into output file.
- Can also run with both combined targets and combined beams (i.e. nue+nuebar)

```
gevgen_nuisance [-h]
          [-r run#]
           -n nev
           -e energy (or energy range)
         -p neutrino_pdg
         -t target_pdg       [DIFFERENT TO GENIE's]
         -f flux_description [DIFFERENT TO GENIE's]
        [-o outfile_name]
        [-w]
        [--seed random_number_seed]
        [--cross-sections xml_file]
        [--event-generator-list list_name]
        [--message-thresholds xml_file]
        [--unphysical-event-mask mask]
        [--event-record-print-level level]
        [--mc-job-status-refresh-rate  rate]
        [--cache-file root_file]
```

gevgen_nuisance

- Options are similar to the standard gevgen app, but target and flux are different (and easier!)

```
gevgen_nuisance -f MINERvA_fhc_numu -t CH  <other arguments>
```

- Only works with GENIE 2.12 and later!

- To build this application, build NUISANCE with the following flags

```
cmake -DUSE_GENIE=1 -DBUILD_GEVGEN=1
```

gevgen_nuisance (2)

- Possible to generate events with the standard gevgen application and "prepare" them for NUISANCE if needed.

- Example generate MINERvA CH events and prepare them.

```
gevgen -f minerva_flux.root,numu -e 0.0,100.0
                -t 1000060120[0.9231],1000010010[0.0769]
                -r 1 -n 2500000 --cross-sections gxspl.gz
```

- Once the sample is made, need to prepare it using our PrepareGENIE application (note the target def is different!)

```
PrepareGENIE -i gntp.ghep.1.root -f minerva_flux.root,numu
             -t 1000060120,1000010010
```

Target is not fractional! For CH2 use
`1000060120,1000010010,1000010010`

Have to pass in the same flux again too..

PrepareNuWro

- NuWro events generated with nuwro-reweight (our special branch) automatically saves the information needed.

- Otherwise, there is another PrepareNuWro app.

```
[USAGE]: PrepareNuwro [-h]
                      [-f]
                      [-F <FluxRootFile>,<FluxHistName>]
                      [-o output.root]
                      inputfile.root [file2.root ...]

-h : Print this message.
-f : Pass -f argument to '$ hadd' invocation.
-F : Read input flux from input descriptor.
-o : Write full output to a new file.
```

Configs

- NUISANCE keeps a global configuration list accessible throughout the code.
- Defaults kept in `$NUISANCE/parameters/config.xml`
- These can be overriden at run time in the card file or on the command line
- Card File

```
<config NAME="OVERRIDE_VALUE" />
```

- Command Line (all applications support –q)

```
nuiscomp -c cardfile.xml -o output.root -q NAME=OVERRIDE_VALUE
```