NUISANCE tutorial

NeUtrino Interaction Systematics ANalyser by Comparing Experiments
NeUtrino Interaction Synthesiser Aggregating Constraints from Experiments
NeUtrino Interaction Systematics from A-Neutrino sCattering Experiments

https://nuisance.hepforge.org/




Imperial College London
The University Of Sheffield.
Imperial College London
UNIVERSITÄT BERN

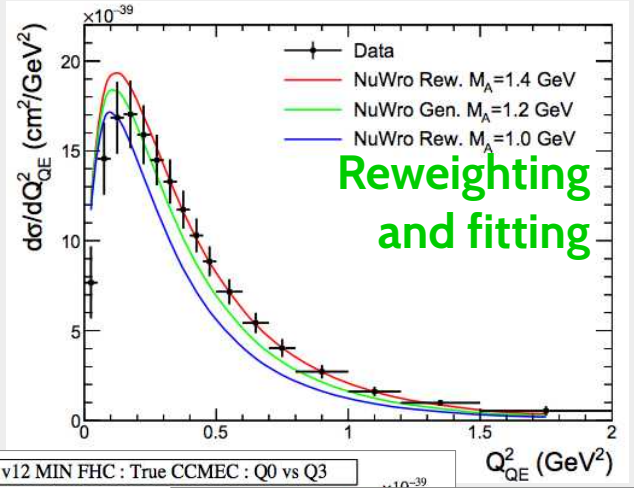
T2K WS, Toronto
19 June 2017

Clarence Wret, Patrick Stowell, Luke Pickering, Callum Wilkinson (main devs)
Help from S. Dytman, U. Mosel, Hayato-san, J. Sobczyk, C. Juszczak, K. Mahn, K. McFarland, G. Perdue, S. Dolan, P. Lasorak, J. Calcutt, C, L. O'Sullivan, and more...

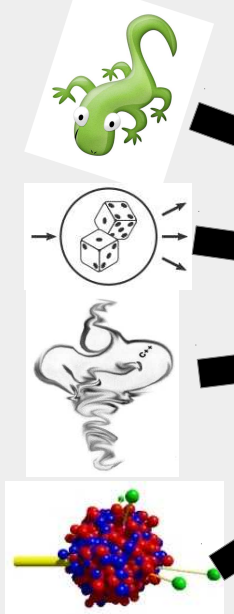
The low-down of NUISANCE

- Interfaces with neutrino interaction generators to give you event distributions
- Confront with data
- Play-time!

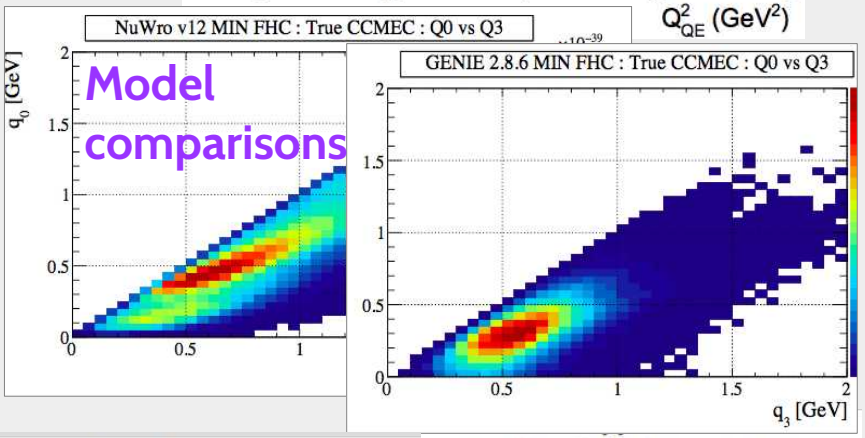
Event selection, cross-section scaling, reweight

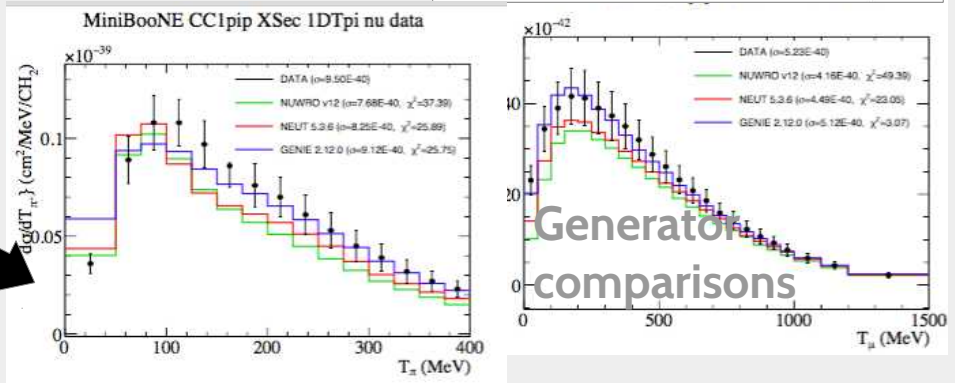


Particle kinematics, interaction targets, etc



Common event format

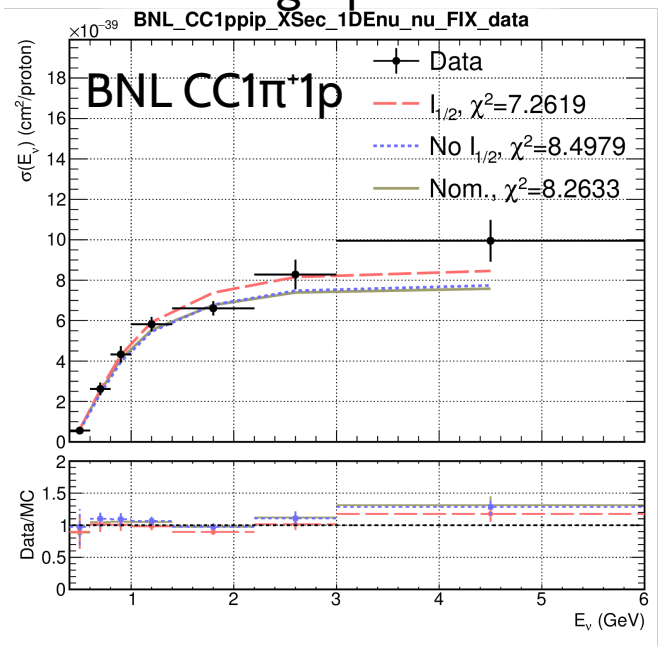




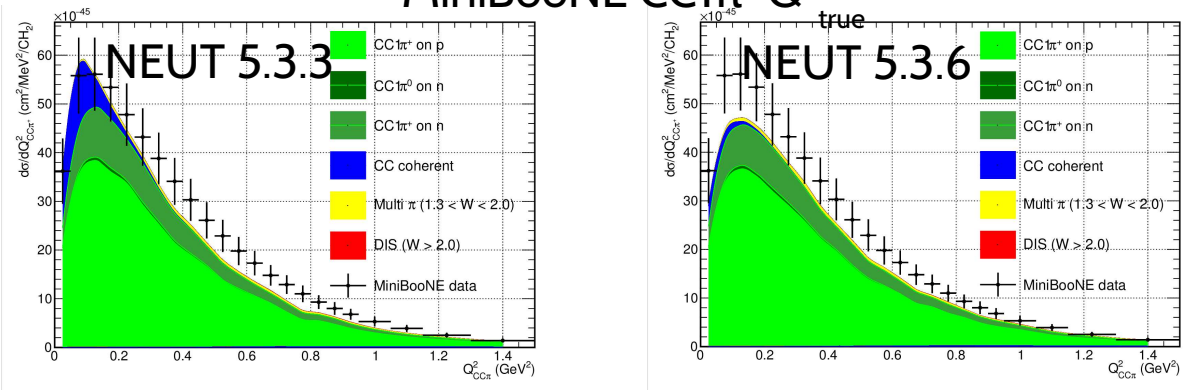
The low down of NUISANCE

- Developed for NEUT model selection and giving OA central values and uncertainties
  - Grew to support all generators and reweighting libs

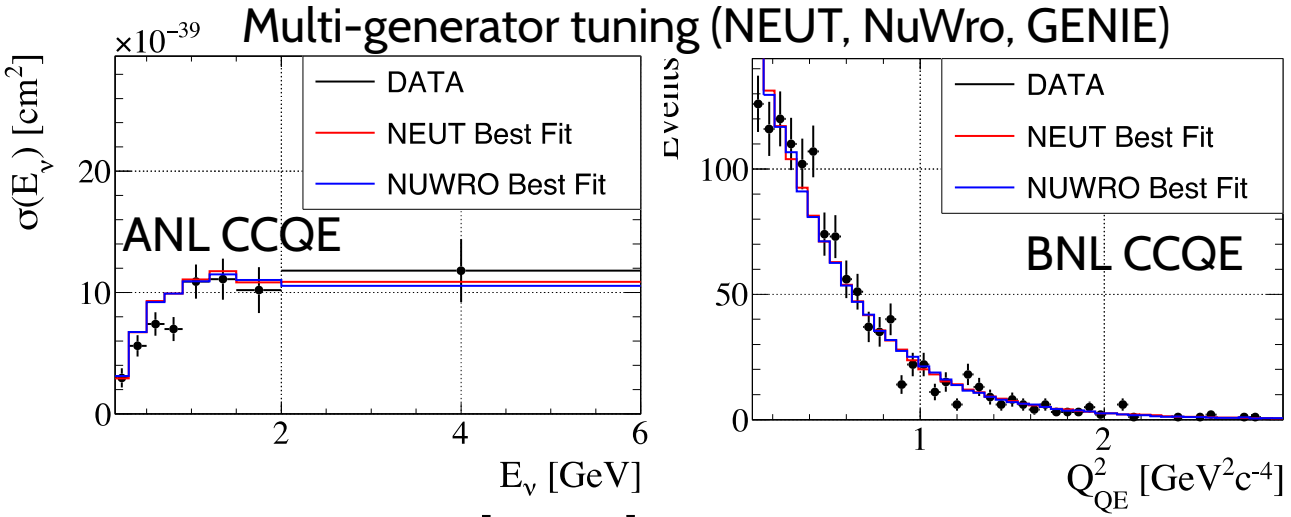
Tune all single pion channels



MiniBooNE CC1π⁺ Q^2



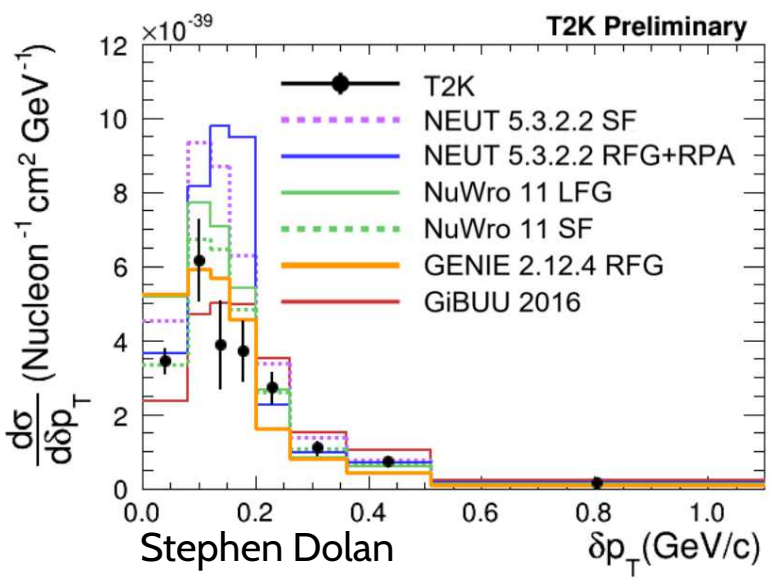
Multi-generator tuning (NEUT, NuWro, GENIE)



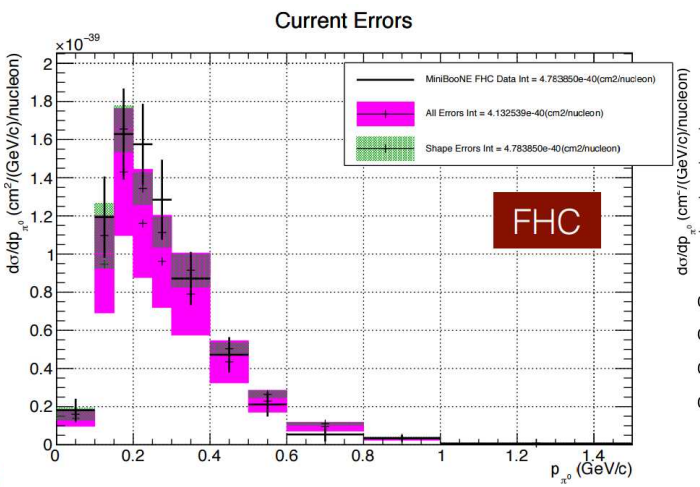
- **<u>Lots of fun for for the cross-section analyser! :D</u>**

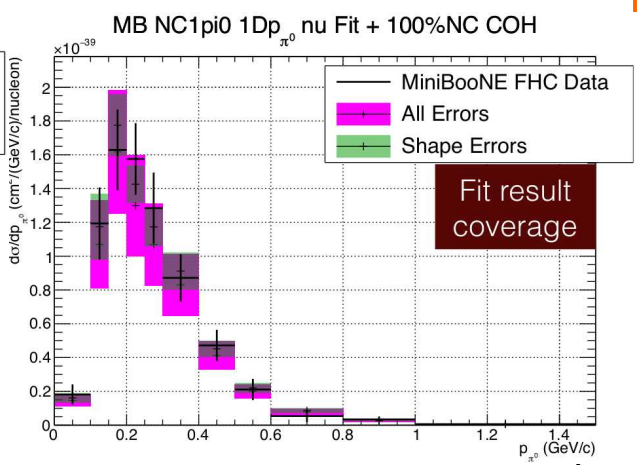
How can NUISANCE help you?

- Challenge the systematics in your analysis
  - Vary 1σ of M_A^{QE}: what does that actually mean?
  - Dominated by an interaction which you don't have a sideband? Get informed from external data?
- What do previous measurements say?
  - Tensions? Interesting distributions? Where do the models differ?
  - Become comfortable with the effects of theory on distributions



Stephen Dolan





Pierre Lasorak

This tutorial

- Patrick set up a virtual box with all the software built
- MC files are also available from us

- We'll start with NEUT and MiniBooNE FHC generated events
- Will probably only have time for simple Data/MC comparisons
- Might have time for a quick fit
- How to make flat trees
- Some details about the framework

- Callum and I will walk around and help out

- Need 10GB+ free space, 3GB+ of RAM
- Please give feedback! Always room for improvement

Setting up the virtual box

- If you don't have Oracle VirtualBox set it up:
  - Get it here
- In the meantime, download the virtual disk image:
  - Get it here
- And also download MiniBooNE FHC NEUT 5.3.3 events:
  - Get it here
- Start up the VirtualBox, give about 3GB ram, mount disk image:
  - Linux Other 64-bit
  - Instructions here
  - Password is **neutrino**
- If you finished fast, check our website

Ok, let's make some nuisance

- Make a shared folder between host and guest
  - On host: `cd ~ && mkdir Share`
  - On host in VirtualBox window: Devices → Shared folders → Settings → "+" box → Add Share folder
  - On guest: `cd ~ && mkdir Share && sudo mount -t vboxsf Share ~/Share`
  - On guest: `ls ~/Share`
- Set up a shared clipboard, Devices → Shared Clipboard → Bidirectional
- Source the environments
  - Open terminal
  - `cd ~/NUISANCEMC && source setup.sh`
- You've now sourced ROOT, GENIE, NuWro, NEUT, their reweighting libraries, T2KReWeight, NIWGReWeight and all dependencies

Getting NUISANCE

- `cd ~/NUISANCEMC/nuisance/branches`
- Get NUISANCE v2r0
  - `source getnuisance.sh v2r0`
  - This will take some time because making on VM is a little slow
- After this is done, do
  - `cd ~/NUISANCEMC/nuisance/branches/v2r0 && source setupnuisance.sh`
  - This sources all of the environment variables, from generators to T2KReWeight to ROOT
- Whilst NUISANCE is building, let's just look a little at how the build system is set up...

NUISANCE versions and build

- NUISANCE is under git version control
  - `git branch -a` and `git tag` are your friends!
- Our build system is CMake, which operates under the principle of making a build directory and generating Makefile(s) for your specific system
  - Won't need for this particular tutorial, here for completeness
  - Make the makefiles with define flags `-D`, e.g. `-DUSE_NEUT=1 -DUSE_GENIE=0`
  - `cd ~/nuisance && mkdir build && cmake ../ -DUSE_YOUR_DEFINES`
  - There should now been generated makefiles
  - `make && make install`

Running NUISANCE

- NUISANCE has a few parameter settings (`parameters/config.xml`) which can be overridden by `-q` when running executables

- Uses "card files" for user input, e.g. what experiment distributions to use, where the output MC is located, what systematics should be varied

- link explaining card files on our wiki

```
- sample SampleName Generator:Location
- parameter ParameterName cent min max step
  FREE/FIX
```

- I'll use the simple card-file format rather than xml

Card files

- Get a list of sample names

```
- nuissamples MiniBooNE
- nuissamples | wc -l
```
gives 216 samples, woop

- Let's choose MiniBooNE CCQE Q2 cross-section for neutrino

```
- vim example_ccqe.card
- sample MiniBooNE_CCQE_XSec_1DQ2_nu
  NEUT:~/Share/MB_numu_fhc_533Aut_merge.root
```

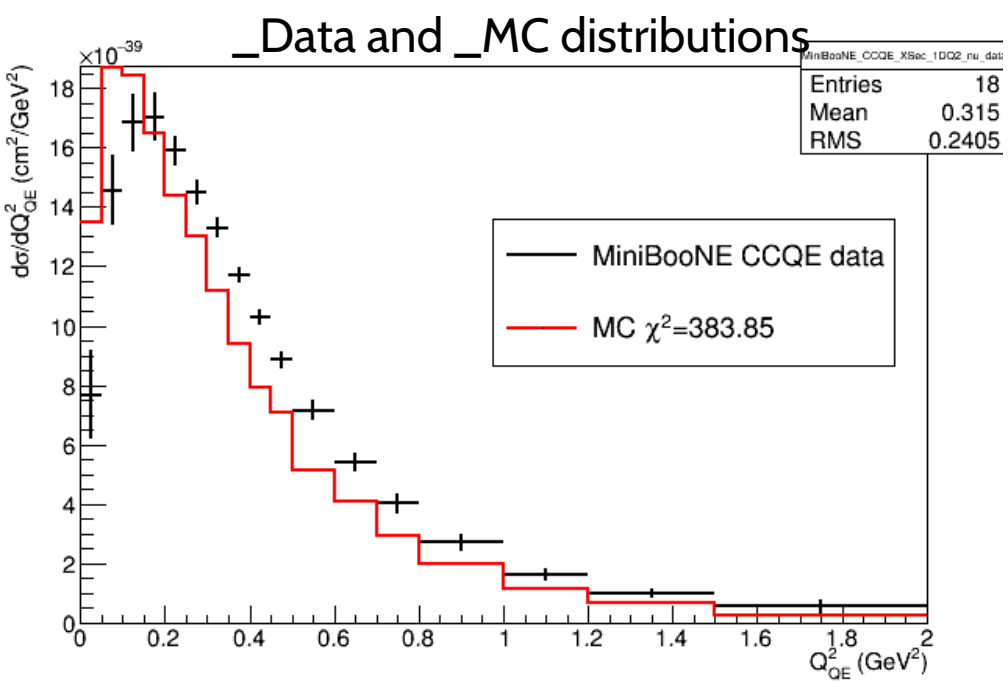
- Now let's produce the data/MC comparison

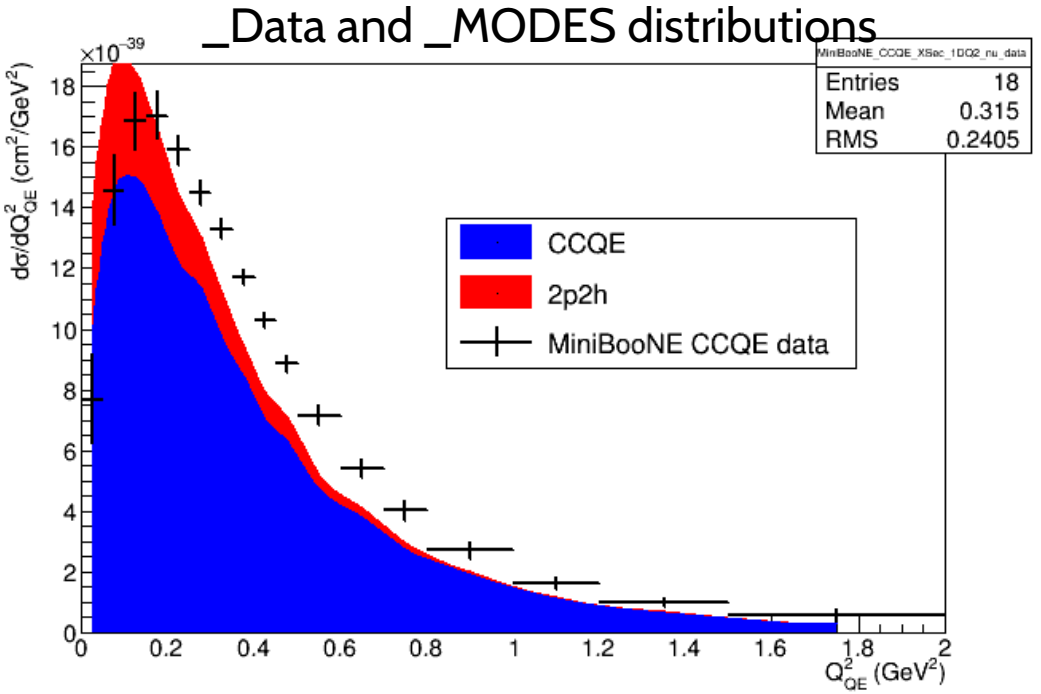
```
- ./nuiscomp -c example.card -o
  my_MiniBooNE_Q2_test.root -q
  MAXEVENTS=50000
```

- `-q MAXEVENTS=50000` says we only load the first 50k events, just a matter of speed. **Always use as many as possible!**

Inspecting the output

- `_Data` and `_MC` are the two important distributions
- Also saves the mode-by-mode distributions
- That's pretty much as simple as it gets!





Varying systematics

- Let's try varying some systematics for MiniBooNE CCQE
- This requires a bit more digging into the generators :(
- `vim ~/NUISANCEMC/neut/branches/neut_5.3.3_Eb_patch/src/reweight/NSyst.h`
- Differs from generator to generator
- Let's vary MaQE between -1 and +1 in steps of 0.5
- NEUT requires VecFFQE 2 (RFG) to be set too

Varying systematics

- Add this to the card file in any place

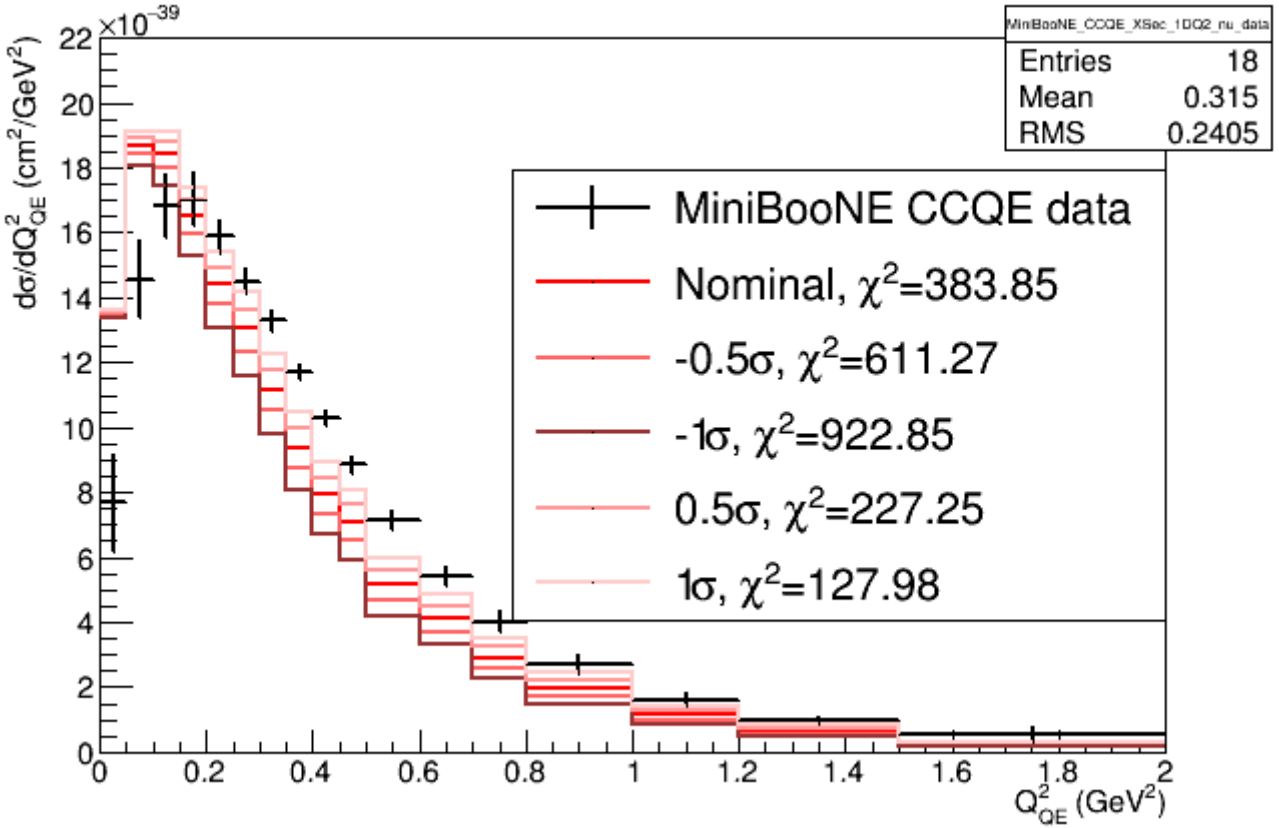
```
• neut_parameter MaCCQE 0.0 -1.0 1.0 0.5 FREE
• neut_parameter VecFFCCQE 2
```

- `neut_parameter` tells the reweight interface that we're about to feed it a NEUTReWeight parameter
  - `genie_parameter, t2k_parameter, niwg_parameter , nuwro_parameter` are the systematic parameters types
  - Also others supported (`ReWeight/WeightUtils.cxx`)
    - e.g. sample normalisations

Varying systematics

- `./nuissyst -c example_ccqe.card -o miniboone_ccqe_var.root -f PlotLimits`

- Now instead get different folders for each variation



- Seems to indicate MiniBooNE CCQE likes a higher M_A^{QE}, wow!

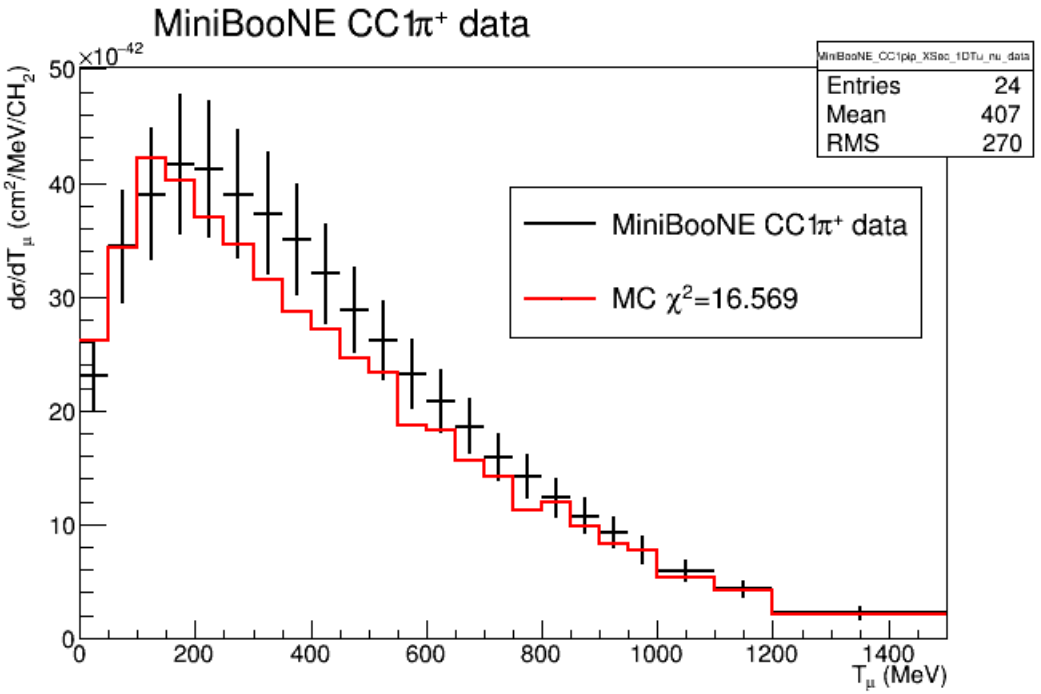
Single pion example

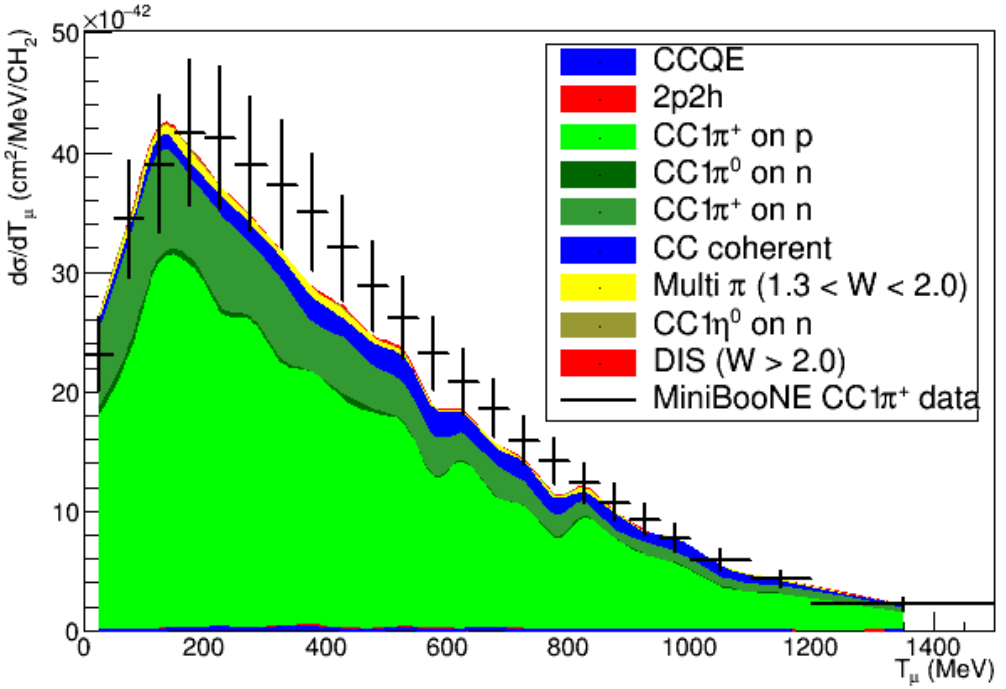
- Let's try a more complicated signal definition, MiniBooNE CC1π^+ T_μ
  - Has a topology-defined signal rather than true interaction mode
  - Will have many contributions to final state

- `vim example_cc1pip.card`
- `sample MiniBooNE_CC1pip_Xsec_1DTu_nu NEUT:~/Share/MB_numu_fhc_533Aut_merge.root`
- `./nuiscomp -c example_cc1pip.card -o miniboone_cc1pip_tmu.root -q MAXEVENTS=50000`

Inspecting the output

- Make similar plots for these distributions





- N.B. we could reuse the same generated NEUT events because
  - Generated with all interaction modes
  - Measurements use same flux on MiniBooNE

Let's start a fit in the background

- Fit MiniBooNE CCQE Q^2 distribution with M_A^{QE}
- First, give larger range to M_A^{QE} in our card file

```
• vim example.card
• neut_parameter MaCCQE 0.0 -3.0 3.0 0.5 FREE
• ./nuismin -c example.card -o
  miniboone_ccqe_q2_fit.root -q MAXEVENTS=50000
```

- Should end up with fit value 2.10 → **M_A^{QE} = 1.61 GeV**
  - This is a slightly unfortunate conversion in NEUT ReWeight and GENIE ReWeight: `real units = nominal*(1+num*1sig)`
  - `nominal` is found in `NFortFns.cc` and `1sig` is found in `NSystUncertainity.cc`

Basic structure of NUISANCE

- Load up a series of `MeasurementBase` derived classes specified by user
  - `Measurement1D, Measurement2D, JointMeas` classes controlling 1D, 2D or joint data format
- Loop over generated MC events, provided in a ROOT file `MeasurementBase::Reconfigure()`
  - Interface with the generator libraries to covert to common format `MeasurementBase::cust_event`
  - Get the dependent variable for each event, e.g. Q^2 `MeasurementBase::FillEventVariables`
  - Does event pass signal definition? `MeasurementBase::isSignal`
  - Fill the histograms, `MeasurementBase::FillHistograms()`
- Calculate a likelihhod, `MeasurementBase::GetLikelihood()`

Adding a measurement

- `Constructor`
  - Set what the sample is (e.g. naming, shape only), point to data, etc
- The dependent variable (`FillEventVariables`)
  - Are we doing $d\sigma/dQ^2$, $d\sigma/dp_\mu$, etc
  - How do we define the dependent variable
- Signal definition (`isSignal`)
  - What events in the input MC do we add to the histogram, which do we discard?

- Let's look closer at these in the implementation of `src/MiniBooNE/MiniBooNE_CC1pip_XSec_1DQ2_nu`

The constructor

Measurement descriptors

E_ν^{true} range

Target

Set data

Set scaling, e.g. /neutron, /nucleon /MeV^2 to /GeV^2

```
//*****************************************************************
MiniBooNE_CC1pip_XSec_1DQ2_nu::MiniBooNE_CC1pip_XSec_1DQ2_nu(nuiskey samplekey) {
//*****************************************************************

  // Sample overview ---------------------------------------------------
  std::string descrip = "MiniBooNE_CC1pip_XSec_1DQ2_nu sample. \n" \
                        "Target: CH \n" \
                        "Flux: MiniBooNE Forward Horn Current nue + nuebar \n" \
                        "Signal: Any event with 1 muon, any nucleons, and no other FS particles \n";

  // Setup common settings
  fSettings = LoadSampleSettings(samplekey);
  fSettings.SetDescription(descrip);
  fSettings.SetXTitle("Q^{2}_{CC#pi} (GeV^{2})");
  fSettings.SetYTitle("d#sigma/dQ_{CC#pi^{+}}^{2} (cm^{2}/MeV^{2}/CH_{2})");
  fSettings.SetAllowedTypes("FIX,FREE,SHAPE/DIAG/NORM/MASK", "FIX/DIAG");
  fSettings.SetEnuRange(0.0, 100.0); // No energy range given in v1r0
  fSettings.DefineAllowedTargets("C,H");

  // CCQELike plot information
  fSettings.SetTitle("MiniBooNE CC1pi");
  fSettings.SetDataInput(  FitPar::GetDataBase() + "MiniBooNE/CC1pip/ccpipXSec_Q2.txt" );
  fSettings.DefineAllowedSpecies("numu");

  FinaliseSampleSettings();

  // Scaling Setup -----------------------------------------------------
  // ScaleFactor automatically setup for DiffXSec/cm2/Nucleon
  fScaleFactor = GetEventHistogram()->Integral("width")*double(1E-38)/double(fNEvents)*(14.08)/TotalIntegratedFlux("width")/1E6;
  // Added /1E6. comes from Q2 being in MeV^2, not GeV^2

  // Plot Setup --------------------------------------------------------
  SetDataFromTextFile( fSettings.GetDataInput() );
  SetCovarFromDiagonal();

  // Final setup  ------------------------------------------------------
  FinaliseMeasurement();

};
```

- Other calls are to helper functions to allocate memory to e.g. TH1Ds
- Inherits from `Measurement1D` class (`MeasurementBase`)

The dependent variables

```
//*********************************************************************
void  MiniBooNE_CC1pip_XSec_1DQ2_nu::FillEventVariables(FitEvent *event){
//*********************************************************************

  if (event->NumFSParticle(211) == 0 ||
      event->NumFSParticle(13) == 0)
    return;

  TLorentzVector Pnu  = event->GetNeutrinoIn()->fP;
  TLorentzVector Ppip = event->GetHMFSParticle(211)->fP;
  TLorentzVector Pmu  = event->GetHMFSParticle(13)->fP;

  // No W cut on MiniBooNE CC1pi+
  double Q2CC1pip = FitUtils::Q2CC1piprec(Pnu, Pmu, Ppip);

  fXVar = Q2CC1pip;

  return;
};
```

Common event format

Check number of final state particles with PDG 211 (positive pion) and PDG 13 (muon)

Get the incoming neutrino

Get the highest momentum particles

Get the reconstructed Q^2

Set the dependent variable of class to be reconstructed Q^2

- The important thing is that `fXVar` gets set at the end of the `FillEventVariables` call

The signal definition

- Signal definition uses a number of helper functions

Define signal as:

incoming numu
CC (outgoing muon)
outgoing positive pion
EnuMin < Enu < EnuMin

To achieve this, we:

Check it's CC-inclusive
Check there's only one pion and that pion is the only meson
Check there's only one charged lepton which agrees with incoming numu

```
//*****************************************************************************
bool MiniBooNE_CC1pip_XSec_1DQ2_nu::isSignal(FitEvent *event) {
//*****************************************************************************
  return SignalDef::isCC1pi(event, 14, 211, EnuMin, EnuMax);
}
```

```
// Require one meson, one charged lepton. types specified in the arguments
bool SignalDef::isCC1pi(FitEvent *event, int nuPDG, int piPDG,
                        double EnuMin, double EnuMax){

  // First, make sure it's CCINC
  if (!SignalDef::isCCINC(event, nuPDG, EnuMin, EnuMax)) return false;

  int nMesons  = event->NumFSMesons();
  int nLeptons = event->NumFSLeptons();
  int nPion    = event->NumFSParticle(piPDG);

  // Check that the desired pion exists and is the only meson
  if (nPion != 1 || nMesons != 1) return false;

  // Check that there is only one final state lepton
  if (nLeptons != 1) return false;

  return true;
}
```

```
bool SignalDef::isCCINC(FitEvent *event, int nuPDG, double EnuMin, double EnuMax) {

  // Check for the desired PDG code
  if (!event->HasISParticle(nuPDG)) return false;

  // Check that it's within the allowed range if set
  if (EnuMin != EnuMax) {
    if (!SignalDef::IsEnuInRange(event, EnuMin*1000, EnuMax*1000)) {
      return false;
    }
  }

  // Check that the charged lepton we expect has been produced
  if (!event->HasFSParticle(nuPDG > 0 ? nuPDG-1 : nuPDG+1)) return false;

  return true;
}
```

Making flat trees

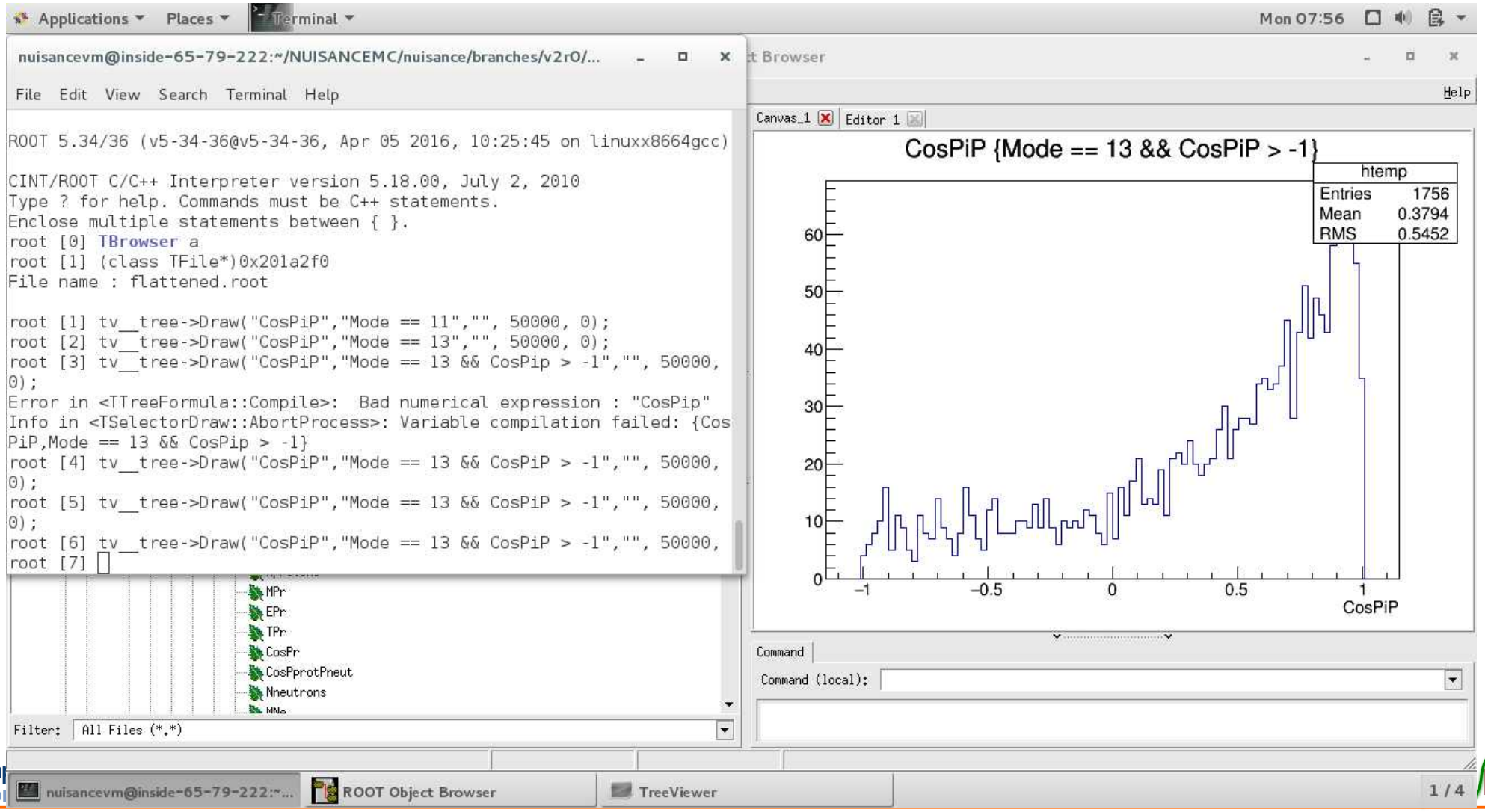
- The `GenericFlux_` class makes flat trees out of the input MC
  - Useful if you don't necessarily want to do data comparisons
  - Or want to learn what distributions are interesting
  - Compare MC with MC
    - e.g. NEUT flat trees with GENIE flat trees

```
• vim flat.card
• sample GenericFlux_Tester
  NEUT:~/Share/MB_numu_fhc_533Aut_merge.root
• ./nuiscomp -c flat.card -o myflattree.root -q
  MAXEVENTS=50000
```

- Can add in systematic parameters too, e.g.
  - `neut_parameter MaCCQE 0.7`

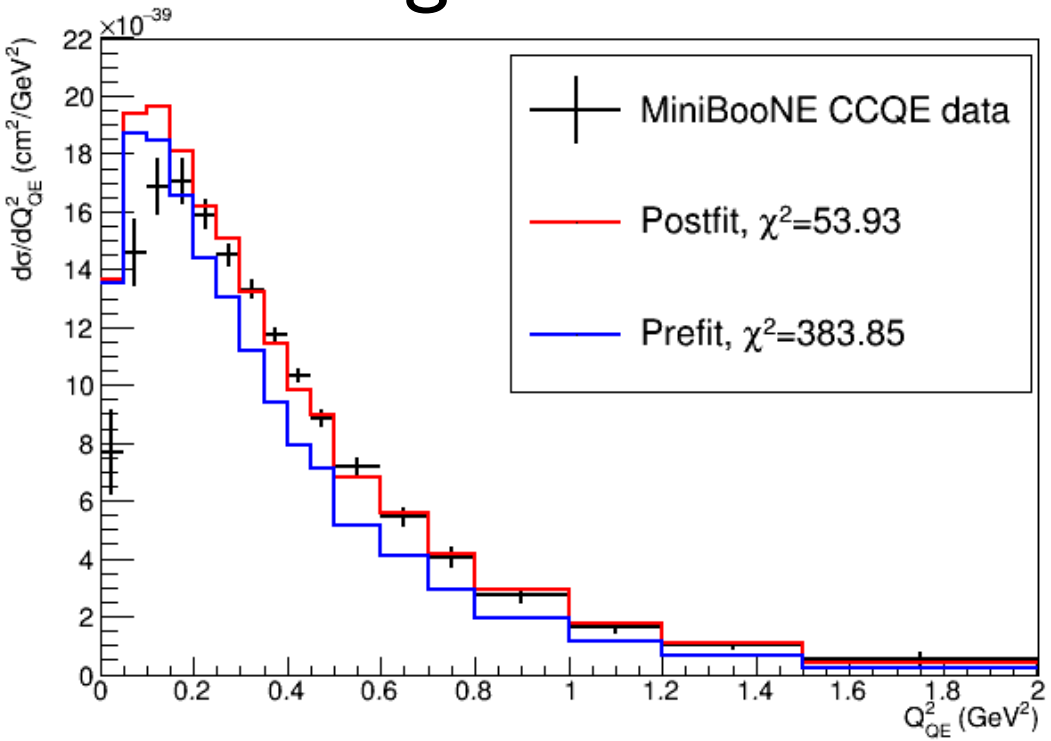
Looking at the flat trees

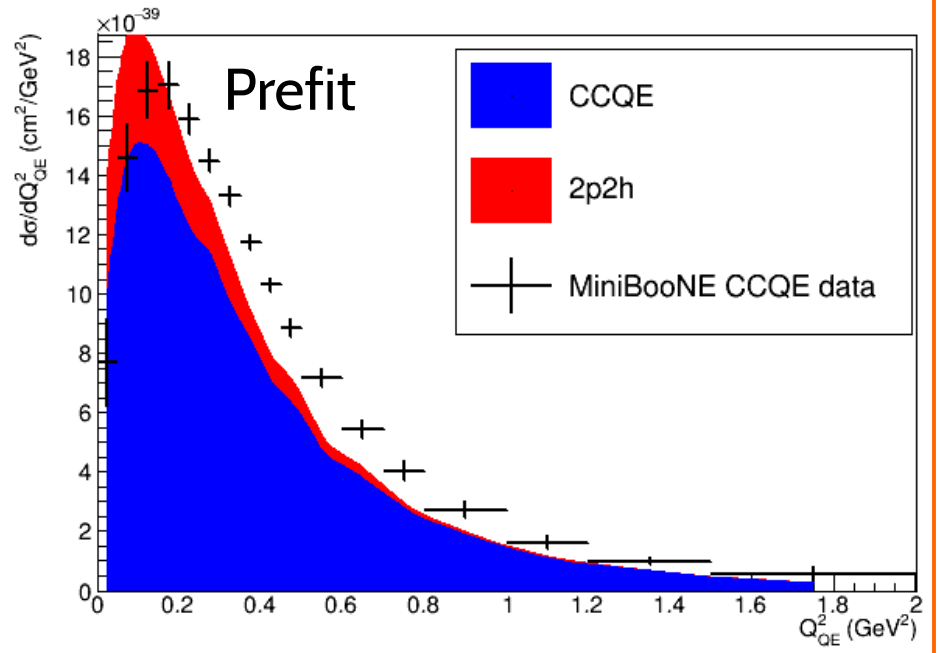
- Produces normal ROOT trees with kinematic variables, modes, etc
- Also a few generic signal definitions
- Here's $\cos\theta_\pi$ for CC1π+1n (NEUT mode 13)

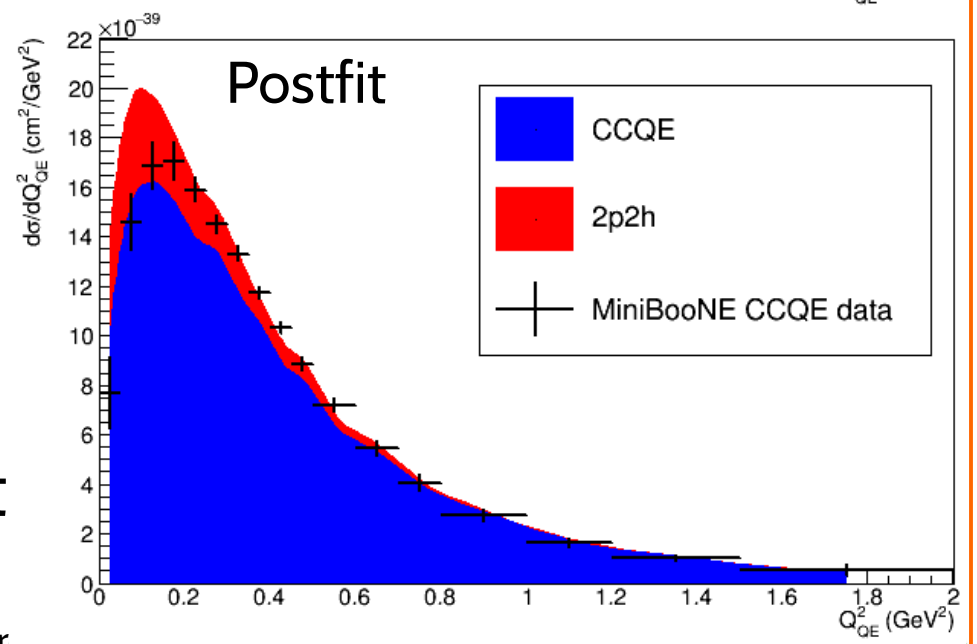


Going back to the fit we sent off

- Looking at the post-fit the low Q^2 behaviour is still not convincing





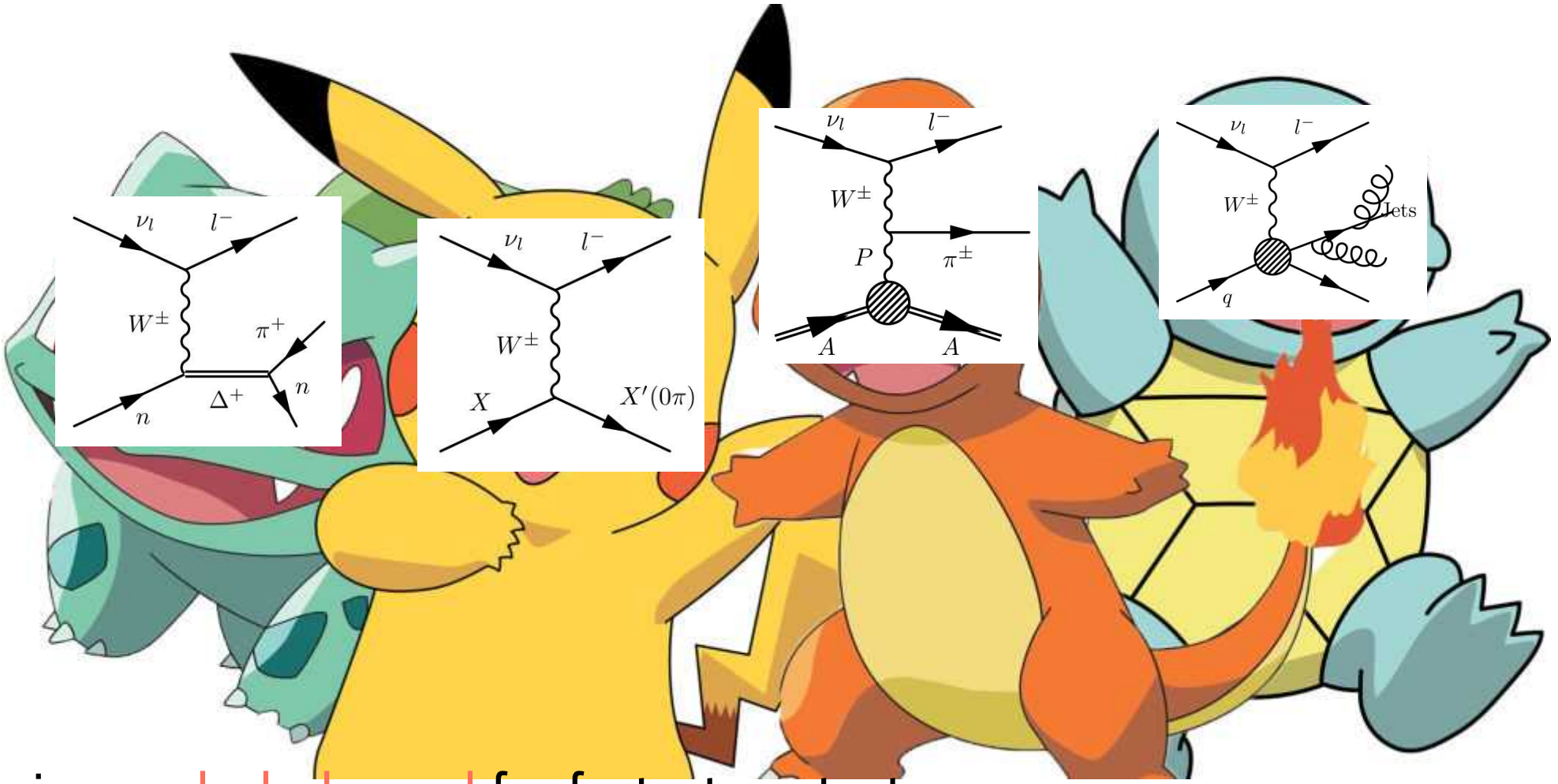


- RPA?
  - Quenches low Q^2 by ~0.6, but not included in the input file

Continuing these studies

- Have all the other generators set up in the virtual box
- Also have pre-generated events here
  - Could download GENIE, GiBUU or NuWro events too
- Try similar studies using the GENIE interface
  - Replace `NEUT:~/Share/MyNEUTfile.root` with `GENIE:~/Share/MyGENIEfile.root` in cardfile
  - Also update `neut_parameter` to `genie_parameter`
- And then you'll be fitting GENIE!
- Generate your own events with alternate models
- **Go catch em all!**

Thanks



- Join our slack channel for fastest contact
- shoot us an email
- Have a glance at our paper describing NUISANCE
- Or just bother us later!