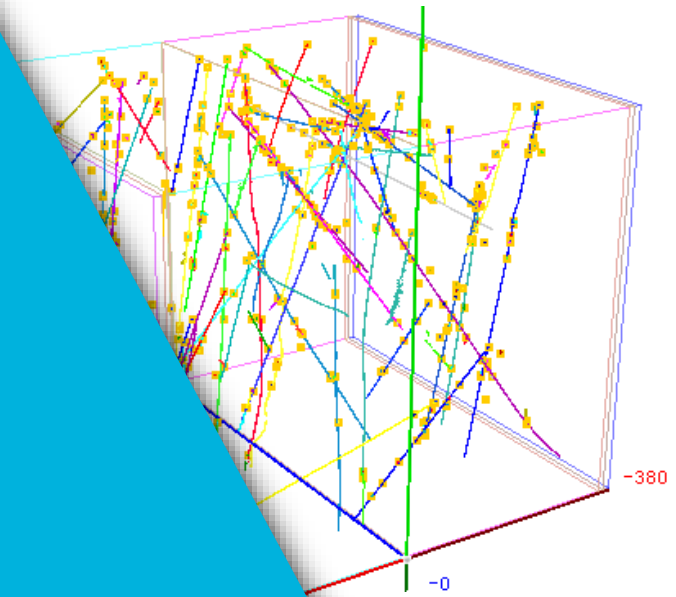
Running the reconstruction

(Exercise)

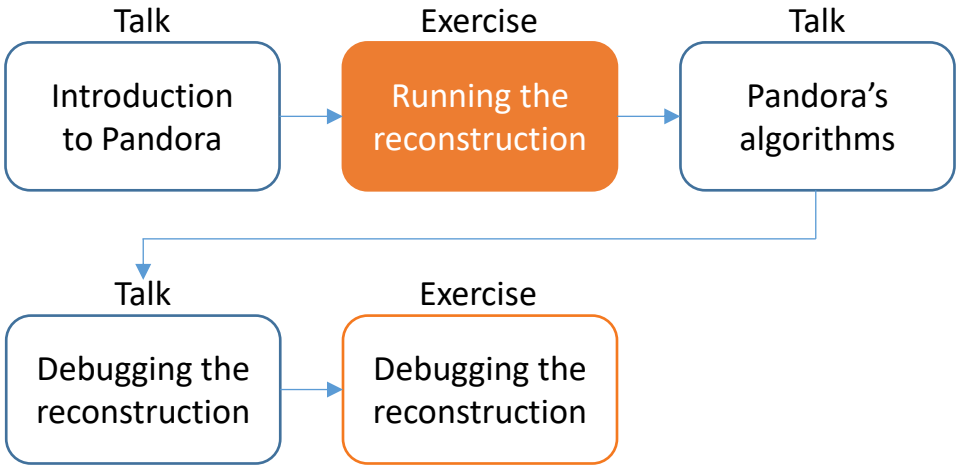


Andy Chappell and Isobel Mawby for the Pandora team

29/10/2024

9th UK LArTPC Software and Analysis Workshop

Reconstruction session



Credit: These slides are based on previous LArSoft workshop slides by Lorena Escudero and Andrew Smith-Jones

Key references:
[Pandora ProtoDUNE paper]
[Pandora MicroBooNE paper]

Goals

- This session is scheduled for 50 minutes
- Main goal 1 - Find and get to grips with the SBND reconstruction FHiCL files
  - Find the standard_reco1_sbnd.fcl and standard_reco2_sbnd.fcl configuration files
  - Look at the different reconstruction steps that we run
  - Understand what each of them do
- Main goal 2 - Run the reconstruction
  - Run the reconstruction on the files we simulated yesterday
  - This includes running Pandora
  - Dump out the new output products to confirm we produced what we wanted

Before we get started…

- Later today we'll be running the event display
- Follow the steps from yesterday to get everything set up. To check, make sure MRB_TOP points to where you expect. Let us know if you have any issues here

```
$ echo $MRB_TOP # This should print the path to your development area
```

Main Goal 1

Understanding the SBND reconstruction FHiCL files

SBND reconstruction FHiCL files

- The full reconstruction is split into two fcls:
  1) standard_reco1_sbnd.fcl
  2) standard_reco2_sbnd.fcl

  which are run in succession

WARNING:

In this tutorial, we'll be using standard_reco2_sbnd.fcl. The version of this file that we will use is **NOT** the same as that in the current release of sbndcode. They will be the same in a few weeks time.

- reco1 generally corresponds to the 'lower-level' reconstruction e.g. signal processing, hit formation, etc…
- reco2 generally corresponds to the 'higher-level' reconstruction e.g. particle creation

- Let's take a closer look at the fcls to see what these stages in more detail

reco1

- Find the location of standard_reco1_sbnd.fcl using SBND's handy fhicl finder script

```
$ find_fhicl.sh standard_reco1_sbnd.fcl
```

Didn't work? Make sure you've setup your working area!

- Open the file with your favourite code editor

```
$ emacs –nw /path/from/command/above
```

Wow, emacs is your favourite? Great choice!
To exit emacs do Ctrl-x Ctrl-C

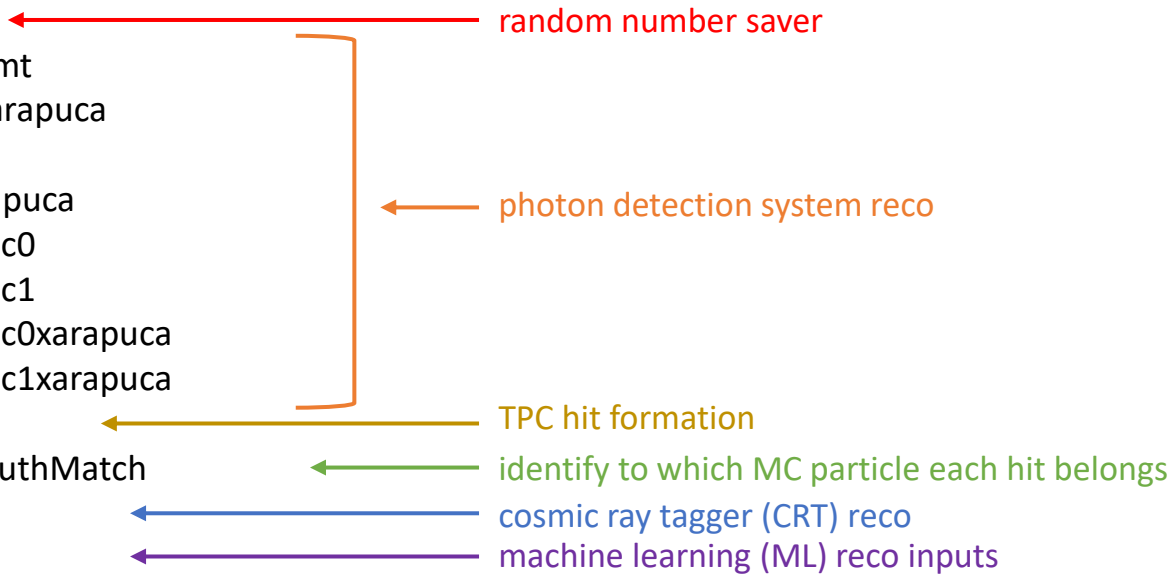
- Find the trigger_paths: [...] block
- You'll learn more in the analysis tutorial, but this block tells us the names of the 'producer' modules that will add products to our .root file
- You'll find the trigger_paths block filled by the name 'reco1'
- You'll find 'reco1' defined as @local::sbnd_reco1_producer_sequence, which itself is defined in workflow_reco1.fcl
- Repeat the above steps to search and view the reco1 producer sequence in workflow_reco1.fcl

```
sbnd_reco1_producer_sequence: [
   rns
  , opdecopmt
  , opdecoxarapuca
  , ophitpmt
  , ophitxarapuca
  , opflashtpc0
  , opflashtpc1
  , opflashtpc0xarapuca
  , opflashtpc1xarapuca
  , gaushit
  , gaushitTruthMatch
  , crtstrips
  , cluster3d
 ]
```

Following a seemingly never-ending series of fhicl files to find the thing you want is pretty standard - keep going, you'll get there eventually!

reco1 – what are these modules?

```
sbnd_reco1_producer_sequence: [
   rns
 , opdecopmt
 , opdecoxarapuca
 , ophitpmt
 , ophitxarapuca
 , opflashtpc0
 , opflashtpc1
 , opflashtpc0xarapuca
 , opflashtpc1xarapuca
 , gaushit
 , gaushitTruthMatch
 , crtstrips
 , cluster3d
 ]
```



reco2

- Follow the same procedure to examine standard_reco2_sbnd.fcl

```
$ find_fhicl.sh standard_reco2_sbnd.fcl

$ emacs –nw /path/from/command/above
```

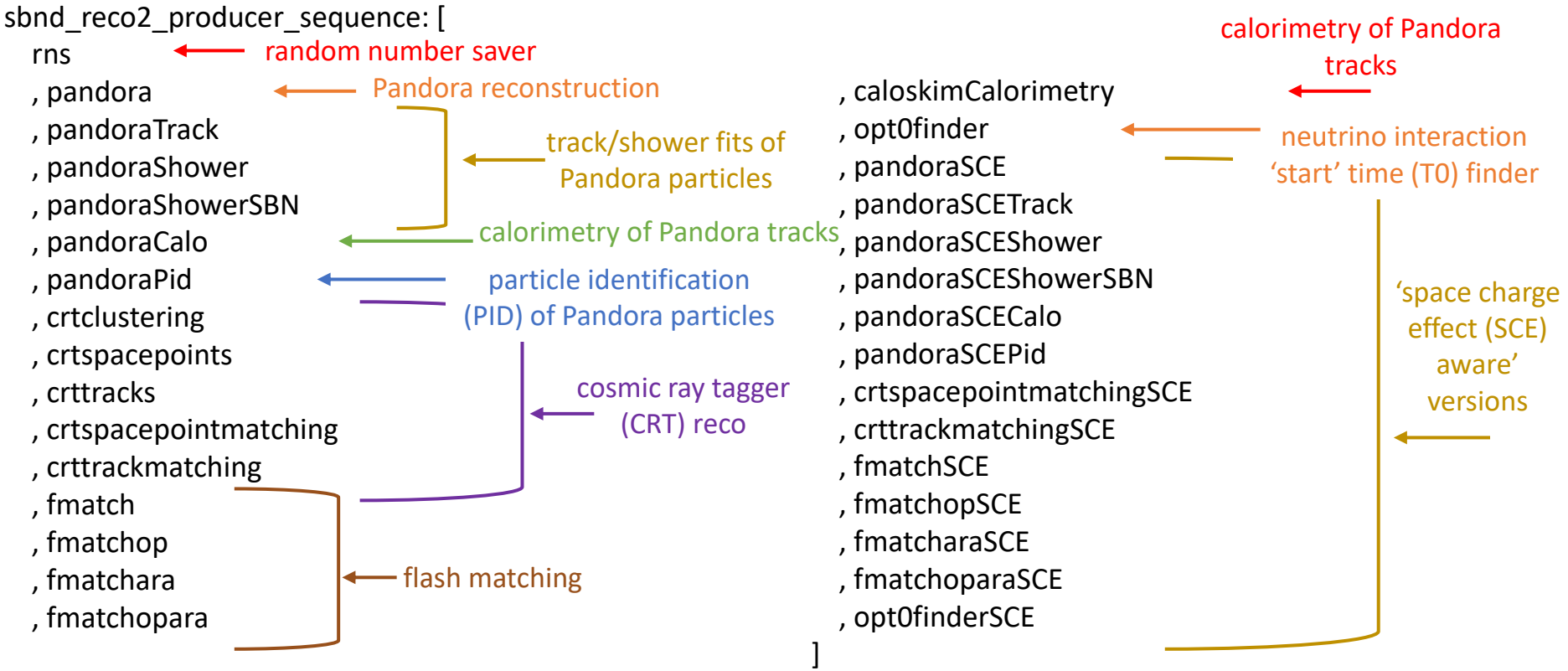
Hunting through fcls is something you'll be doing time and time again, and one day they will suddenly make sense

- Find the trigger_paths: [...] block
- You'll find the trigger_paths block filled by the name 'reco2'
- You'll find 'reco2' defined as @local::sbnd_reco2_producer_sequence, which itself is defined in workflow_reco2.fcl
- Repeat the above steps to search and view the reco2 producer sequence in workflow_reco2.fcl

```
sbnd_reco2_producer_sequence: [
  rns
  , pandora
  , pandoraTrack
  , pandoraShower
  , pandoraShowerSBN
  , pandoraCalo
  , pandoraPid
  , crtclustering
  , crtspacepoints
  , crttracks
  , crtspacepointmatching
  , crttrackmatching
  , fmatch
  , fmatchop
  , fmatchara
  , fmatchopara
  , caloskimCalorimetry
  , opt0finder
  , pandoraSCE
  , pandoraSCETrack
  , pandoraSCEShower
  , pandoraSCEShowerSBN
  , pandoraSCECalo
  , pandoraSCEPid
  , crtspacepointmatchingSCE
  , crttrackmatchingSCE
  , fmatchSCE
  , fmatchopSCE
  , fmatcharaSCE
  , fmatchoparaSCE
  , opt0finderSCE
]
```

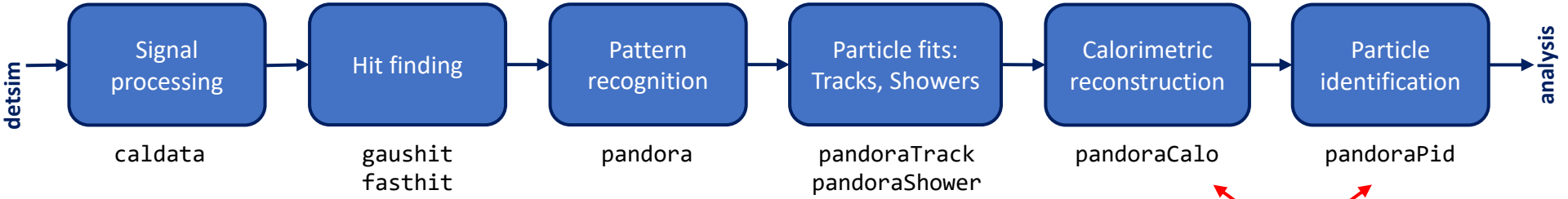
In the last session, we were introduced to pandora, but there are many other steps in the reconstruction chain too!

reco2 – what are these modules?



A cumulative reconstruction output

- There's a lot happening in the reconstruction stage
- To demonstrate how these modules accumulate a reconstruction output, let's focus on the Pandora 'workflow' i.e. ignoring the optical and CRT reconstruction



- Each experiment has its own unique needs, so expect to see some differences in the reconstruction chain if you work on MicroBooNE, ProtoDUNE, DUNE, etc.
- Similarly, Pandora is configured differently for different experiments, and different use cases (e.g. atmospheric vs beam neutrino interactions)
- Next, let's see how Pandora is configured for SBND

Historically, modules using Pandora outputs frequently have Pandora in the name.

Most of these modules aren't part of Pandora and aren't maintained by the Pandora development team.

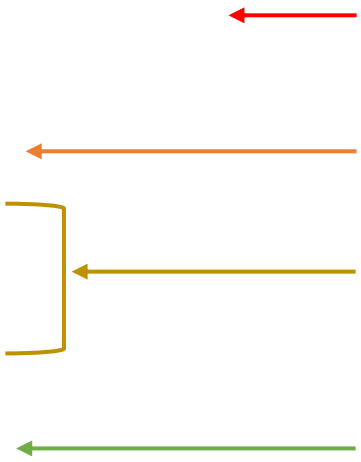
Pandora's configuration

- Practically all LArSoft modules have a configuration defined by parameters e.g. say that we had a producer module that added a number to our root file – it would likely have a parameter which defines the number that is added, and that parameter is set in our fcl files
- fcl files often *#include* many other fcl files, and it can be unclear where our parameters are set
- fhicl-dump answers this question, and follows all *#includes* to get the bottom-line configuration
- We can pipe (|) its output to less and search for a producer to learn more:

```
$ fhicl-dump standard_reco2_sbnd.fcl | less -p "pandora:"
```

Less's –p option allows us to jump straight to the part of the file that we are interested in. In less, use the ↑/↓ keys to move up and down, and 'q' to exist.

```
pandora: {
  ConfigFile: "PandoraSettings_Master_SBND.xml"
  EnableLineGaps: true
  EnableMCParticles: false
  EnableProduction: true
  GeantModuleLabel: "largeant"
  HitFinderModuleLabel: "gaushit"
  PrintOverallRecoStatus: false
  ShouldPerformSliceId: true
  ShouldRunAllHitsCosmicReco: true
  ShouldRunCosmicHitRemoval: true
  ShouldRunCosmicRecoOption: true
  ShouldRunNeutrinoRecoOption: true
  ShouldRunSlicing: true
  ShouldRunStitching: true
  SimChannelModuleLabel: "simtpc2d:simpleSC"
  UseGlobalCoordinates: true
  UseHitWidths: true
  module_type: "StandardPandora"
}
```



The settings file that contains the list of algorithms that Pandora will run

The producer module that created the hits that we are going to feed into Pandora

The steering parameters that tell Pandora which of its high-level reconstruction steps it should execute

The type of the LArSoft module to use

Main Goal 2

Running the reconstruction

Running the reconstruction

- We are now poised to run the reconstruction! Make a directory to work in, and run it:

```
$ mkdir -p $MRB_TOP/reco/work
$ cd $MRB_TOP/reco/work
$ lar -c standard_reco1_sbnd.fcl -n -1 -s /path/to/my/detsim/file.root -o reco1_events.root
$ lar -c standard_reco2_sbnd.fcl -n -1 -s reco1_events.root -o reco2_events.root
```

The -n -1 option means run over all events in the input file

Can also run on pre-made gen+g4+detsim files in:
/mnt/gridpp/poolhomes/PPEGroup/LAR24/TPCSimulation/detsim_tutorial.root

reco1 can take some time to run

```
TimeTracker printout (sec)                                  Min        Avg        Max        Median     RMS        nEvts
Full event                                                  0.515653   0.630062   0.857711   0.577501   0.122178   10
source:RootInput(read)                                      0.000233144  0.00842777  0.0686421  0.000264929  0.0204246  10
reco2:rns:RandomNumberSaver                                 2.0329e-05  4.86311e-05  0.000269584  2.2662e-05  7.37634e-05  10
...
reco2:pandora:StandardPandora                               0.264394   0.324777   0.544608   0.299226   0.0806975  10
reco2:pandoraTrack:LArPandoraTrackCreation                  0.00274014  0.00409558  0.00646157  0.00403605  0.00105976  10
reco2:pandoraShower:LArPandoraModularShowerCreation         0.0230009  0.0264048  0.0363764  0.025806  0.00366943  10
reco2:pandoraShowerSBN:LArPandoraModularShowerCreation      0.00185641  0.00215675  0.00280114  0.0021136  0.00027365  10
reco2:pandoraCalo:GnocchiCalorimetry                        0.00202068  0.00246629  0.00407539  0.00228423  0.000550001  10
reco2:pandoraPid:Chi2ParticleID                             0.00011043  0.000201841  0.000876067  0.000128414  0.000225004  10
reco2:pandoraSCEPid:Chi2ParticleID                          9.2036e-05  0.000117752  0.000145417  0.000117708  1.49363e-05  10
...
end_path:out1:RootOutput                                    2.194e-06  3.3834e-06  1.1412e-05  2.4845e-06  2.68091e-06  10
...
end_path:out1:RootOutput(write)                             0.0321085  0.0625225  0.194061   0.053615   0.0457909  10
```

We can check to see that everything we expected has been executed, and see how long each took

So… what's new?

- Run eventdump.fcl on the .root file to see the new collections we've just made

```
$ lar -c eventdump.fcl -s reco2_events.root -n 1
```

```
PROCESS NAME | MODULE LABEL..... | PRODUCT INSTANCE NAME..... | DATA PRODUCT TYPE..... | .SIZE
SingleGen... | TriggerResults..... | ..... | art::TriggerResults..... | ....1
SingleGen... | generator..... | ..... | std::vector<simb::MCTruth>..... | ....1
SingleGen... | rns..... | ..... | std::vector<art::RNGsnapshot>..... | ....1
G4..... | genericcrt..... | ..... | std::vector<sim::AuxDetSimChannel>..... | ....0
G4..... | largeant..... | LArG4DetectorServicevolTPCPlaneVert..... | std::vector<sim::SimEnergyDeposit>..... | ....0
G4..... | largeant..... | LArG4DetectorServicevolPMT..... | std::vector<sim::SimEnergyDeposit>..... | ....0
...
DetSim..... | simtpc2d..... | badchannels..... | std::vector<int>..... | ..313
DetSim..... | simtpc2d..... | gauss..... | std::vector<recob::Wire>..... | 11276
DetSim..... | simtpc2d..... | simpleSC..... | std::vector<sim::SimChannel>..... | 11276
DetSim..... | simtpc2d..... | wiener..... | std::vector<recob::Wire>..... | 11276
DetSim..... | crtsim..... | ..... | std::vector<sbnd::crt::FEBData>..... | ....0
...
Reco1..... | rns..... | ..... | std::vector<art::RNGsnapshot>..... | ....1
Reco1..... | gaushit..... | ..... | art::Assns<recob::Wire,recob::Hit,void>..... | .1791
Reco1..... | opflashtpc1..... | ..... | art::Assns<recob::OpHit,recob::OpFlash,void>..... | ....0
Reco1..... | crtstrips..... | ..... | std::
Reco1..... | gaushit..... | ..... | std::
...
Reco2..... | pandoraShower..... | ..... |
Reco2..... | pandora..... | ..... | art:
Reco2..... | pandoraSCEShower..... | ..... |
Reco2..... | pandora..... | ..... | std::vector<recob::PFParticle>..... | ....3
Reco2..... | pandoraSCEPid..... | ..... | art::Assns<recob::Track,anab::ParticleID,void>.... | ....6
...
```

You'll see a much longer list! I've cut out a lot output so that it all fits on the slide

these are the existing data products from the previous steps

these are the new data products that we produced in reco1

… and in reco2

Additional information

Configuring Pandora steps

(For reference)

- Pandora's full reconstruction workflow is designed to handle neutrino interactions in dense cosmic ray environments
- You'll see in the upcoming lecture how we've designed algorithms, and entire algorithm chains to target specific topologies and have:
  - a cosmic-ray optimised algorithm chain
  - a beam neutrino interaction optimised algorithm chain
  - an atmospheric neutrino interaction optimised algorithm chain
  - a test-beam optimised algorithm chain (for use at ProtoDUNE)
  - a low energy neutrino interaction optimised reconstruction chain (for e.g. supernova neutrinos at the DUNE FD)

- SBND is a surface detector in a neutrino beam, so sees both neutrino and cosmic ray interactions
- We want to run the cosmic algorithm chain over cosmic interactions, and the neutrino algorithm chain over neutrino interactions ⇒ we'll want to run the so called 'consolidated reconstruction' chain
- We can configure Pandora to run one, many, or all steps by modifying its steering parameters
- Let's have a go at changing how Pandora runs!

Pandora Config

(For reference)

- Firstly, what steering parameters do we have and what do they mean?

Let's do it!

(For reference)

- Make a new directory to work in during this session
- Then create a new FHiCL file with the lines below
- Then save and close the file

```
$ mkdir -p $MRB_TOP/reco/config
$ cd $MRB_TOP/reco/config       # Put your new .fcl file here
$ vim my_reco_sbnd_basic.fcl
```

Please use your favourite text editor, here we use vim. If you accidentally opened vim and want to close it type Esc, :qa, Return ↵

include the standard configuration

```
#include "standard_reco2_sbnd.fcl"

physics.producers.pandora.ShouldRunAllHitsCosmicReco:  false
physics.producers.pandora.ShouldRunStitching:          false
physics.producers.pandora.ShouldRunCosmicHitRemoval:   false
physics.producers.pandora.ShouldRunSlicing:            false
physics.producers.pandora.ShouldRunCosmicRecoOption:   false
physics.producers.pandora.ShouldRunNeutrinoRecoOption: true
physics.producers.pandora.ShouldPerformSliceId:        false
```

With these values of the steering parameters:

- we'll treat the input hits as if they belong to a single interaction
- and will reconstruct them under the neutrino algorithm chain
- performing no stitching across the central gap in SBND – JINKIES!

Pointing to a new configuration

(For reference)

- We'll need to make sure that LArSoft will know where to look for our new FHiCL file, to do this we add it to the FHICL_FILE_PATH environment variable. Start by printing it to the terminal:

```
$ echo $FHICL_FILE_PATH
```

- You will see many many directories, all separated by a ':'.
- To add our reco/config folder to this list run the following command:

```
$ export FHICL_FILE_PATH=$MRB_TOP/reco/config:$FHICL_FILE_PATH
```

- echo the FHICL_FILE_PATH again to check that everything worked (it should be the first in the list)
- Now run fhicl-dump again to make sure our new configuration file is set up as we want

```
$ fhicl-dump my_reco_sbnd_basic.fcl | less -p "pandora:"
```

- Now try to run reco2 (on your reco1 output) with your new fcl file!

```
$ lar –c my_reco_sbnd_basic.fcl -n -1 -s /path/to/my/reco1/file.root -o alternative_reco2_events.root
```