ProtoDUNE Science Workshop
CERN, 29/06/2016

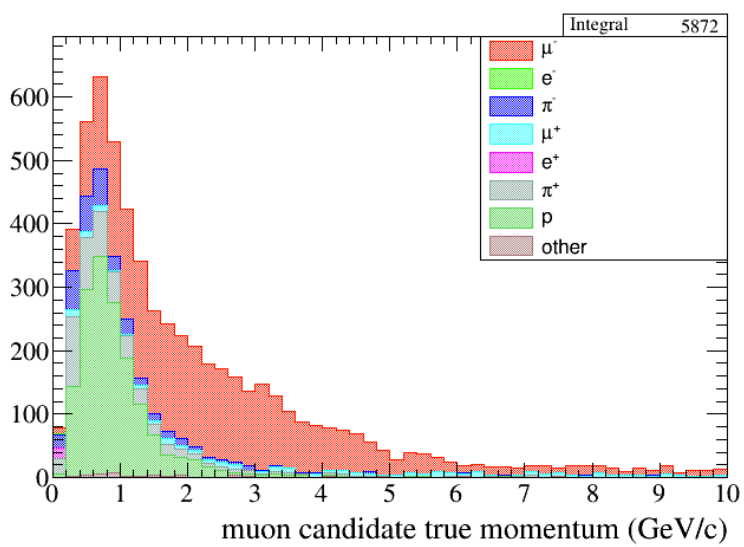
HighLAND
analysis framework

Alexander Izmaylov, A. Cervera, J.J. Gomez-Cadenas, P. Novella, M. Sorel

IFIC-Valencia

Introduction

- **HighLAND**, **High L**evel **A**nalysis at a **N**eutrino **D**etector
- HighLAND has been crucial for T2K near detector analyses
  - Has decreased considerably the learning curve and speed up analysis development
- **Highly optimized**, **thread safe, compiled c++ code** and run on the shell command line (not as root macro)
- Adapted to DUNE from the T2K near detector
  - A working prototype exists (see later)
  - Here the first DUNE HighLAND plot



What HighLAND provides

- **General analysis tools**
  - Event loop
  - Tools for multiple simultaneous event selections
  - Tools for numerical systematic error propagation
- **Tools for drawing the analysis results**
- **Data Reduction functionality. Example:**
  - LArSoft —> MiniTree —> MicroTree —> NanoTree
- **Tools for incorporating specific analyses into the framework**
  - Extensible event data model
  - Hierarchy of analyses depending on each other

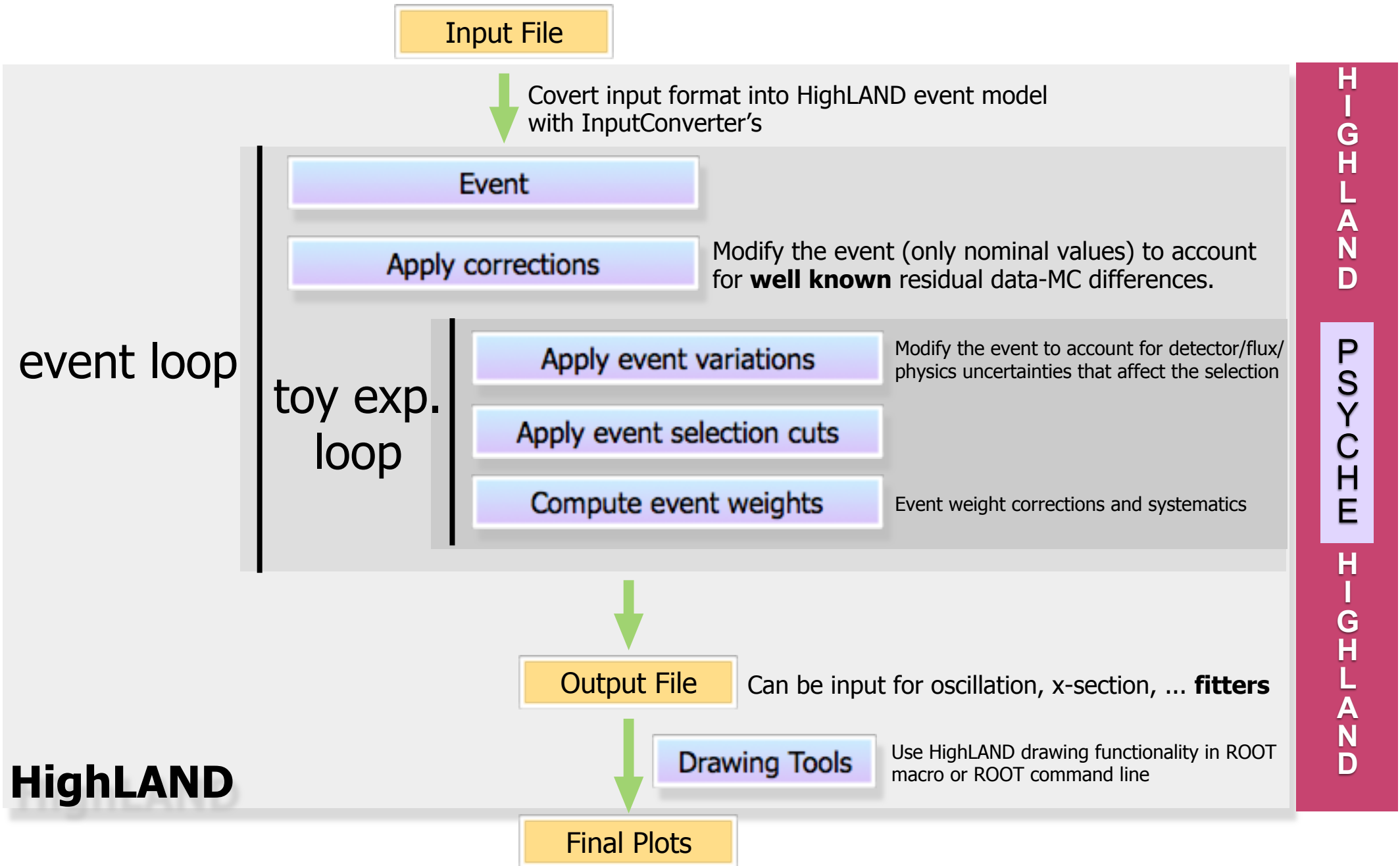
HighLAND for DUNE

- Ideally the same framework for all project components:
  - **FD, ND, prototypes**: optimization, input for oscillation analyses, x-section, proton decay, other new physics, performance studies, etc.
- This is possible because:
  - **HighLAND** can accept **any input format**
  - The basic **event model can be extended** by the user to match the requirements of its particular analysis
- Benefits of common framework
  - Moving from one group to another should be easier
  - **Correlated systematics** between near and far detector
  - People from different groups would speak the same language when talking about selections, systematics and their associated technicalities

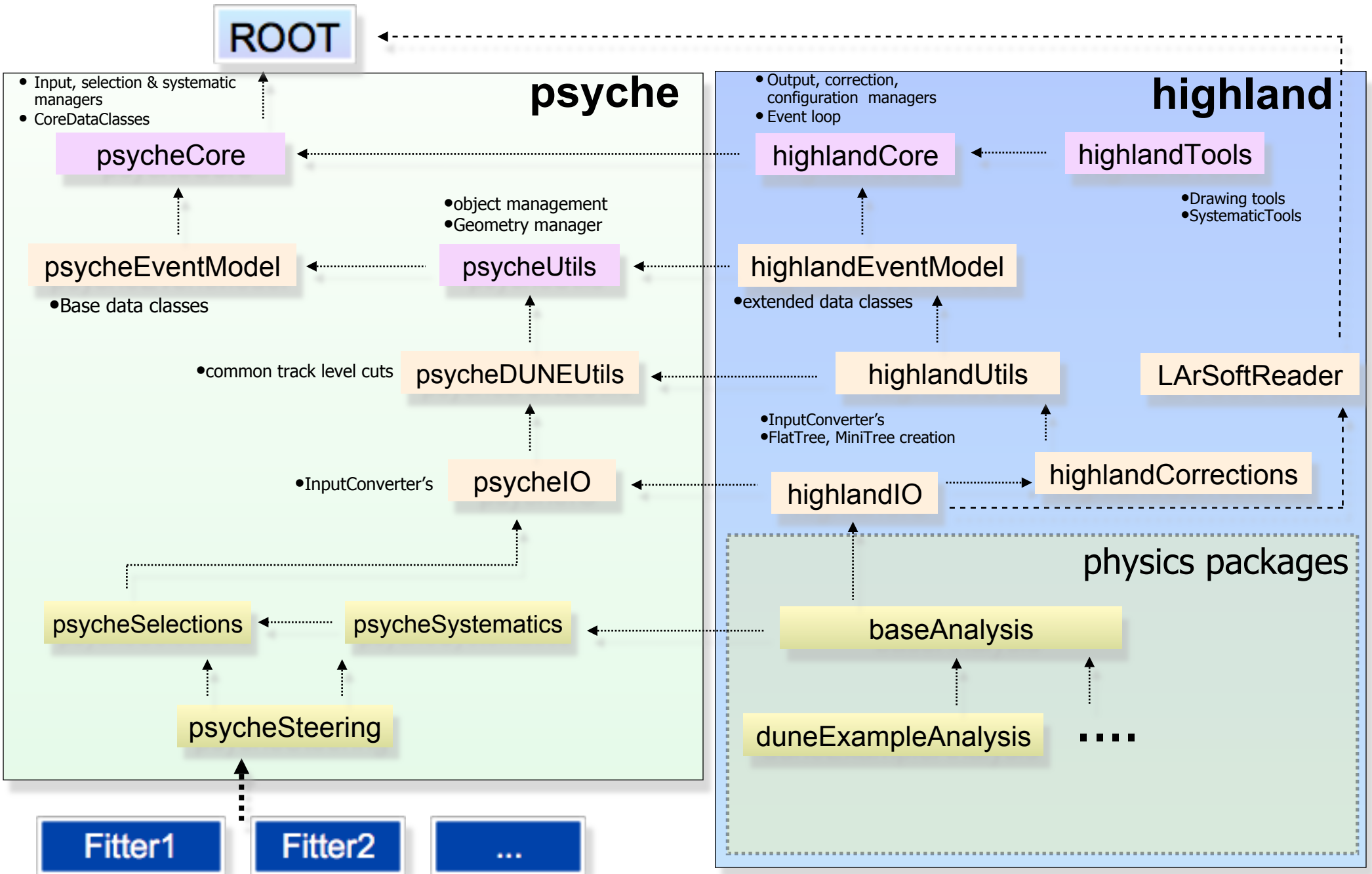
HighLAND for ProtoDUNE

- Highland might be specially useful for ProtoDUNE
  - Given the time constraints
  - Suitable for both prototypes
- Could start prototyping test beam analyses immediately: we need reconstruction output files
- The same analysis could be performed in both prototypes, facilitating the comparisons
- …..

Flow and structure



Package hierarchy



From T2K to DUNE

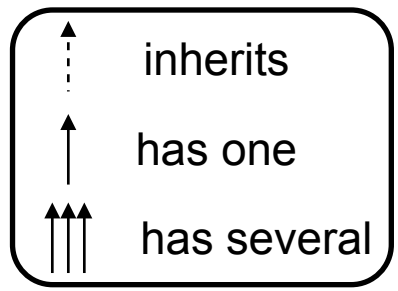
- The first thing we have done is to remove any detector dependent code from the basic packages
  - psycheCore, psycheUtils, highlandCore, highlandTools
  - Those packages are now common to T2K and DUNE
- We have setup a Event Model for DUNE, to be reviewed and optimized with people's feedback (psycheEventModel and highlandEventModel)
- InputConverter's for DUNE reconstruction output file:
  - We are in the process of understanding the file format (**Art event** or **AnalysisTree**) such that it can be converted into the Analysis Event Model
  - As explained later, several converters already exist under highlandIO
- There is also a analysis example duneAnalysisExample

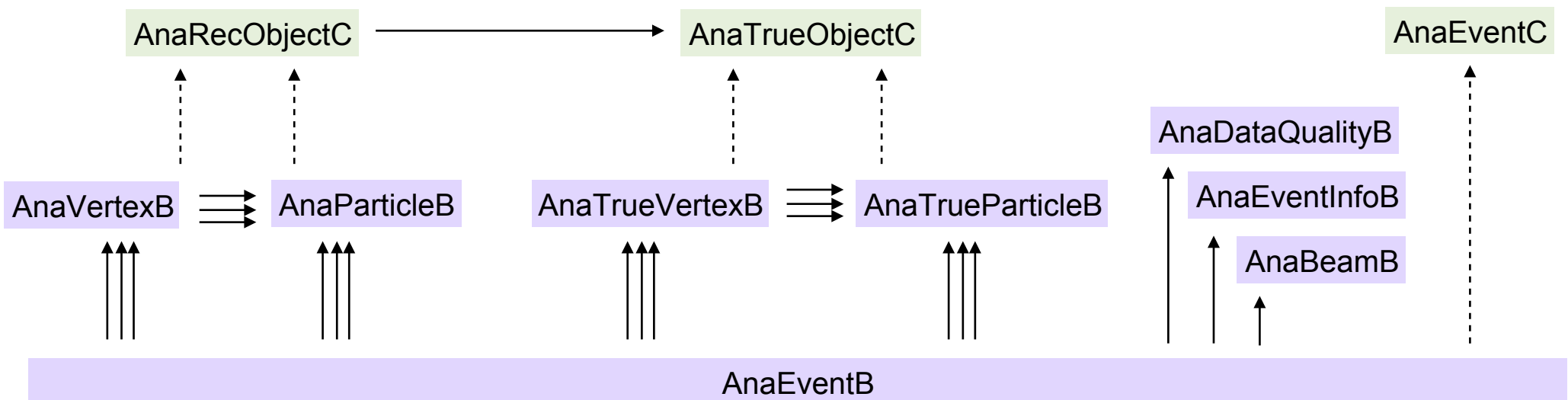
DUNE event model

- A preliminary event model, sufficient to dump most info from the DUNE Analysis tree

classes in psycheCore/v0r0/CoreDataClasses.hxx

classes in psycheEventModel/v0r0/BaseDataClasses.hxx





extended event model

- with hits, clusters, showers, etc

DUNE InputConverters

- These objects convert DUNE root files into the HighLAND event model. At the moment we have three converters:
  - **LArSoftTreeConverter:** recon output with **art** format
  - **anaTreeConverter**: 35t analysis format
  - **nueAnaTreeConverter**: 10kt analysis format
- The **LArSoft** converter needs the art event model headers. Those can be obtained with TFile::MakeProject from a root LArSoft file. A package **LArSoftReader** has been added to the DUNE HighLAND distribution, such that users don't have to worry about that
  - It runs but needs to improve the true-reco association
- A converter for ProtoDUNE files will be developed soon (anaTreeConverter could be probably used)

Event Selection I

- It's a collection of "steps" (cuts and actions)
  - Each step inherits from the base class **StepBase**
  - It has a single method **Apply**, which returns true or false (only relevant for cuts)
- Each selection inherits from **SelectionBase**, which has a main mandatory method **DefineSteps**

```
//********************************************************************
void numuCCSelection::DefineSteps(){
//********************************************************************

    // Cuts must be added in the right order
    // last "true" means the step sequence is broken if cut is not passed (default is "false")
    AddStep(StepBase::kCut,    "event quality (good beam/detector)", new EventQualityCut(),           true);
    AddStep(StepBase::kCut,    "> 0 tracks ",                        new TotalMultiplicityCut(),      true);
    AddStep(StepBase::kAction, "find lepton candidate",              new FindCandidateAction());
    AddStep(StepBase::kAction, "find vertex",                        new FindVertexAction());
    AddStep(StepBase::kCut,    "track quality + fiducial volume",    new TrackQualityFiducialCut(),   true);
    AddStep(StepBase::kAction, "find veto track",                    new FindVetoTrackAction());
    AddStep(StepBase::kCut,    "external veto",                      new ExternalVetoCut());
    AddStep(StepBase::kCut,    "muon PID",                           new MuonPIDCut());

    // This is a selection with a single branch, but each branch should have am alias
    SetBranchAlias(0,"trunk");
```

Event Selection II

- Example of action (fills the box ...)

```
//*****************************************************
bool FindCandidateAction::Apply(AnaEventB& event, ToyBoxB& box) const{
//*****************************************************

    // Find leading tracks with good quality and only in FGD FV
    cutUtils::FindLeadingTracks(event,box);

    // For this selection the LeptonCandidate track is the HMN (Highest Momentum Negative) track
    box.LeptonCandidate = box.HMNtrack;
    return true;
}
```

- Example of cut (uses the filled box ...)

```
//*****************************************************
bool MuonPIDCut::Apply(AnaEventB& event, ToyBoxB& box) const{
//*****************************************************

  // LeptonCandidate must exist
  if (!box.LeptonCandidate) return false;

  // And it should have a valid momentum value
  if (box.LeptonCandidate->Momentum < 0.) return false;

  // Apply the PID cut in the utilities namespace to the LeptonCandidate
  return cutUtils::MuonPIDCut( box.LeptonCandidate );
}
```

The box is used to pass info from one step to another

Systematics

- HighLAND provides full systematic propagation functionality
- Probably too early to talk about that, **see back slides**

DrawingTools

- This is one of the framework classes which can be accessed from a ROOT macro or command line
- It is initialized with a micro-tree file (HighLAND output)
- When opening a root session the HighLAND classes are already visible so you just do

```
root [1] DrawingTools draw("microtree.root")
```

- Now you can start doing plots
  - Some examples later when talking about the duneAnalysisExample
  - Much more examples in the backup

T2K & DUNE maintenance

- We have to maintain the code for DUNE and T2K
- How can we optimize resources ? Options:
  - **an external library** (as ROOT or GEANT4): **discarded**
  - **two versions separately maintained: discarded**
  - **a common repository for T2K and DUNE**: is that possible provided that T2K uses **CVS** and DUNE uses **git** ?
- The current approach is a **git** repository (with CVS structure for the common packages, that can be ignored by DUNE users)
- We are discussing within T2K a possible migration to git (of HighLAND only). The CVS structure will be kept (DUNE users can ignore it) until T2K moves to git

Elements and support

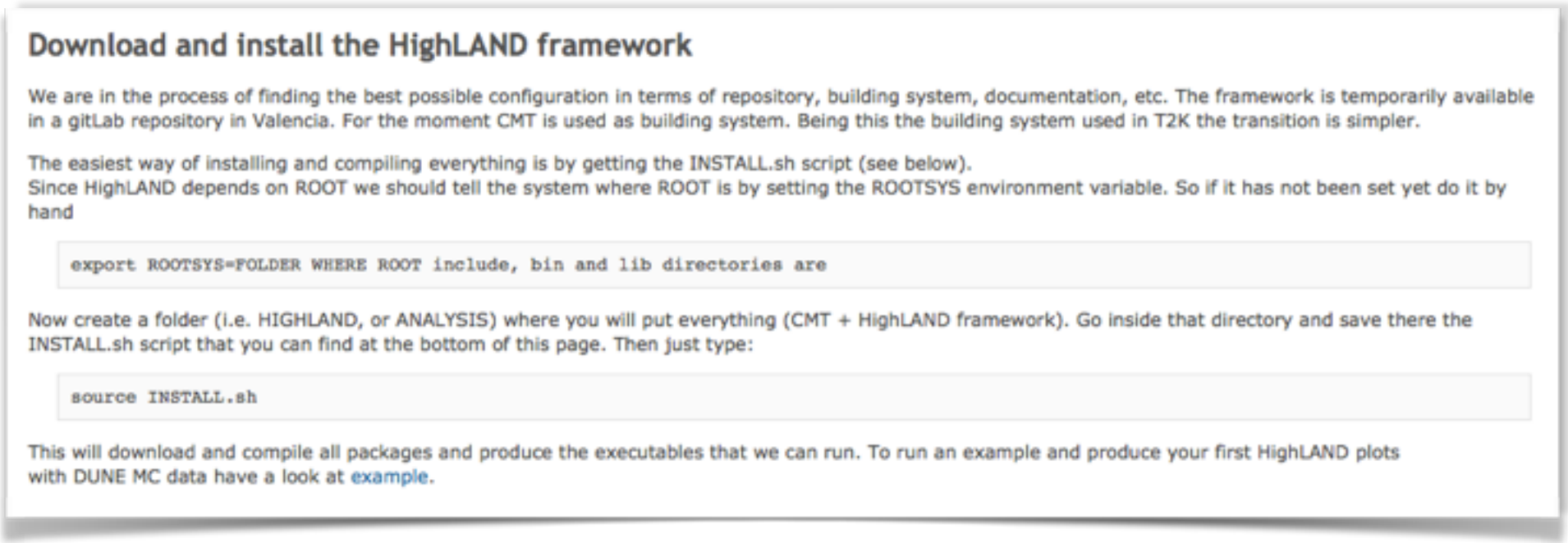
- **Repository**: A prototype version has been installed in a **gitLab** repository in Valencia
- **Building system:** To have a smooth transition from T2K we keep **CMT** for the moment. This is a very light package that can be obtained from the same git repository
- **Documentation:** highland comes with some **doxygen** documentation, which would have to be installed at some Fermilab URL. For the moment we are using simplified documentation in **redmine**

  https://cdcvs.fnal.gov/redmine/projects/highland/wiki
- **Bug tracking**: also redmine. Nothing there yet !!!
- **Validation**: **Jenkins** continues integration tests will be used at some point

HighLAND installation

- This is an screen capture of

https://cdcvs.fnal.gov/redmine/projects/highland/wiki/install



Download and install the HighLAND framework

We are in the process of finding the best possible configuration in terms of repository, building system, documentation, etc. The framework is temporarily available in a gitLab repository in Valencia. For the moment CMT is used as building system. Being this the building system used in T2K the transition is simpler.

The easiest way of installing and compiling everything is by getting the INSTALL.sh script (see below).
Since HighLAND depends on ROOT we should tell the system where ROOT is by setting the ROOTSYS environment variable. So if it has not been set yet do it by hand

```
export ROOTSYS=FOLDER WHERE ROOT include, bin and lib directories are
```

Now create a folder (i.e. HIGHLAND, or ANALYSIS) where you will put everything (CMT + HighLAND framework). Go inside that directory and save there the INSTALL.sh script that you can find at the bottom of this page. Then just type:

```
source INSTALL.sh
```

This will download and compile all packages and produce the executables that we can run. To run an example and produce your first HighLAND plots with DUNE MC data have a look at example.

- Please try, and if you have any problems email us (acervera@ific.uv.es, izmaylov@ific.uv.es) or submit an issue to **redmine**

https://cdcvs.fnal.gov/redmine/projects/highland/issues

duneExampleAnalysis

- A HighLAND package containing an example for DUNE has been committed to git. It contains:
  - A very simple event selection with few cuts
  - Few systematic propagation algorithms
  - The AnalysisAlgorithm that configures the analysis and produces a root file with interesting information about the analysis
  - A macro to make few plots
- Few plots using a 10kt analysis tree file will be shown

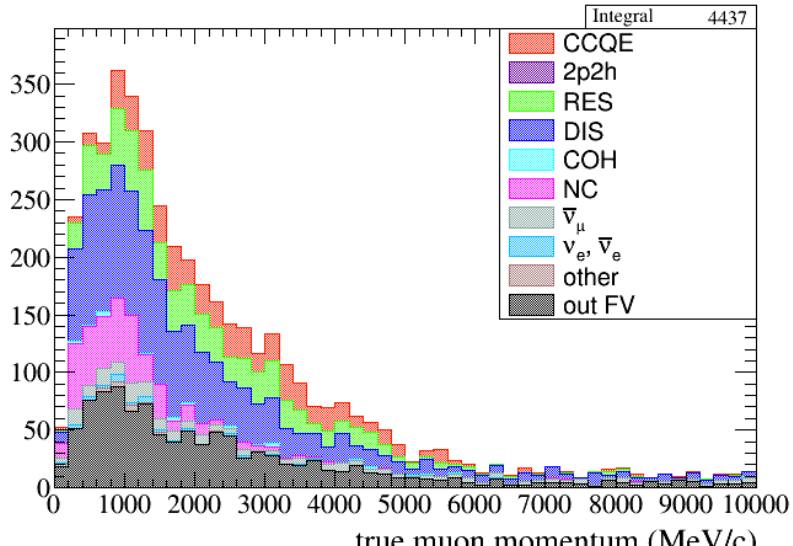
Event selection

- The event selection contain few simple cuts, which can be analyzed using the DrawingTools

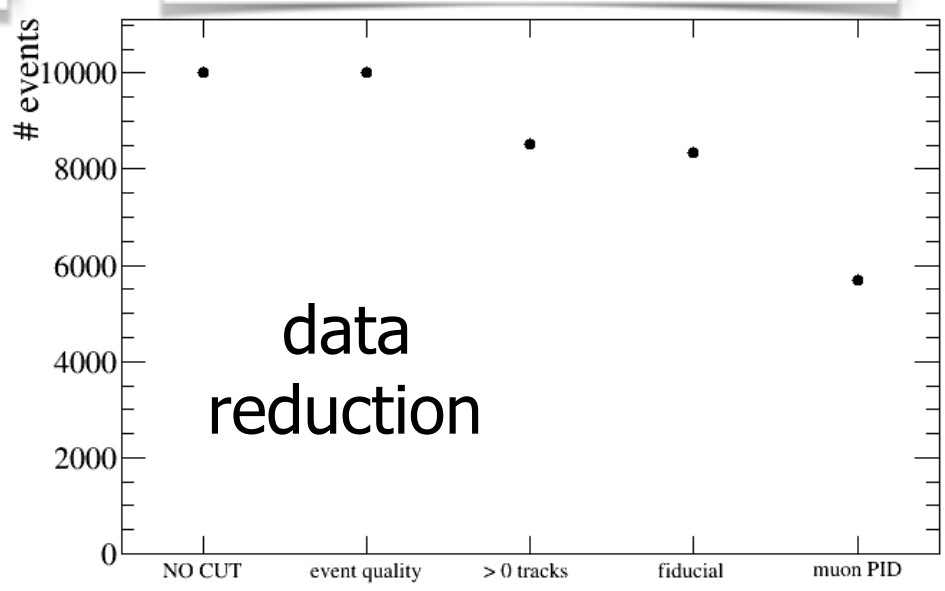
```
root [1] DrawingTools draw("test.root")
root [2] draw.DumpCuts()
------------------------------------------------------------------
      Cuts for selection 'duneExample'  with no branches
------------------------------------------------------------------
  #: type       title                        break  branches

  0: cut        event quality                1      0
  1: cut        > 0 tracks                   1      0
  2: cut        fiducial                     0      0
  3: cut        muon PID                     0      0
------------------------------------------------------------------
```

```
root [7]  draw.SetTitleX("true muon momentum (MeV/c)");
root [8]  draw.Draw(default,"selmu_truemom",50,0,10000,"reaction","accum_level>3");
```

after all cuts




```
root [2] draw.DrawEventsVSCut(default)
```



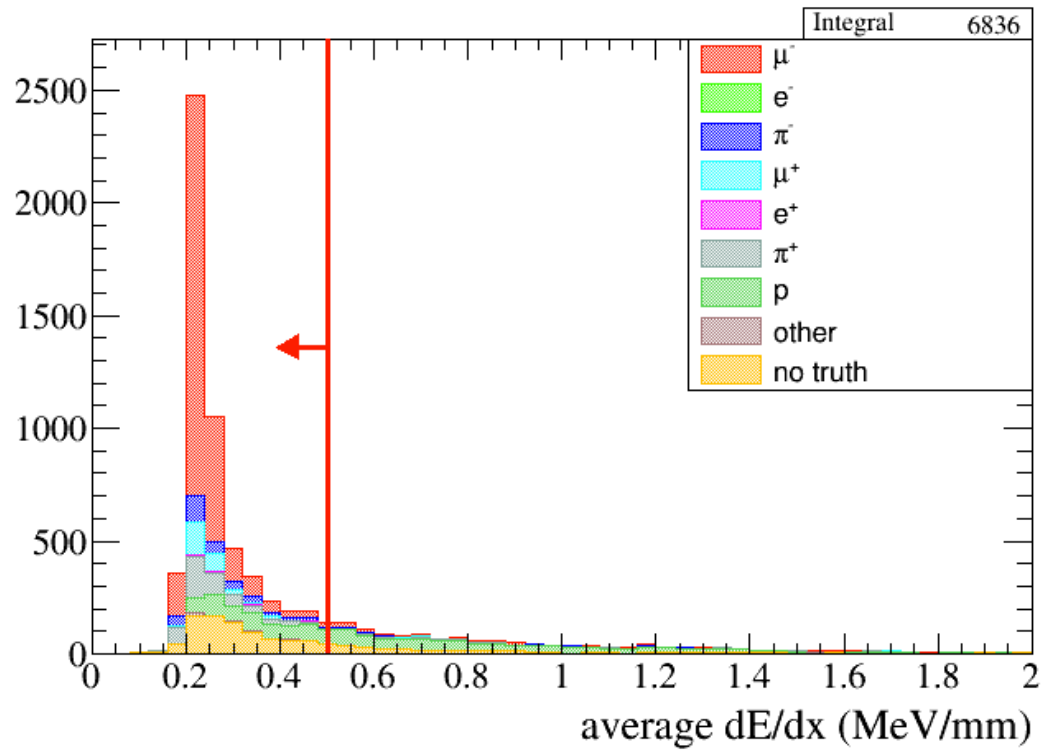
Example of cut

```
root [6] draw.SetTitleX("average dE/dx (MeV/mm)");
root [7] draw.Draw(default,"selmu_dedx",50,0,2,"particle","accum_level>2"
root [8] draw.DrawCutLineVertical(0.5,true,"l");
```

accum_level>2 means events passing cut 2
(fiducial, before the PID cut)

MuonPIDCut is a class inheriting from StepBase and implementing a single method Apply

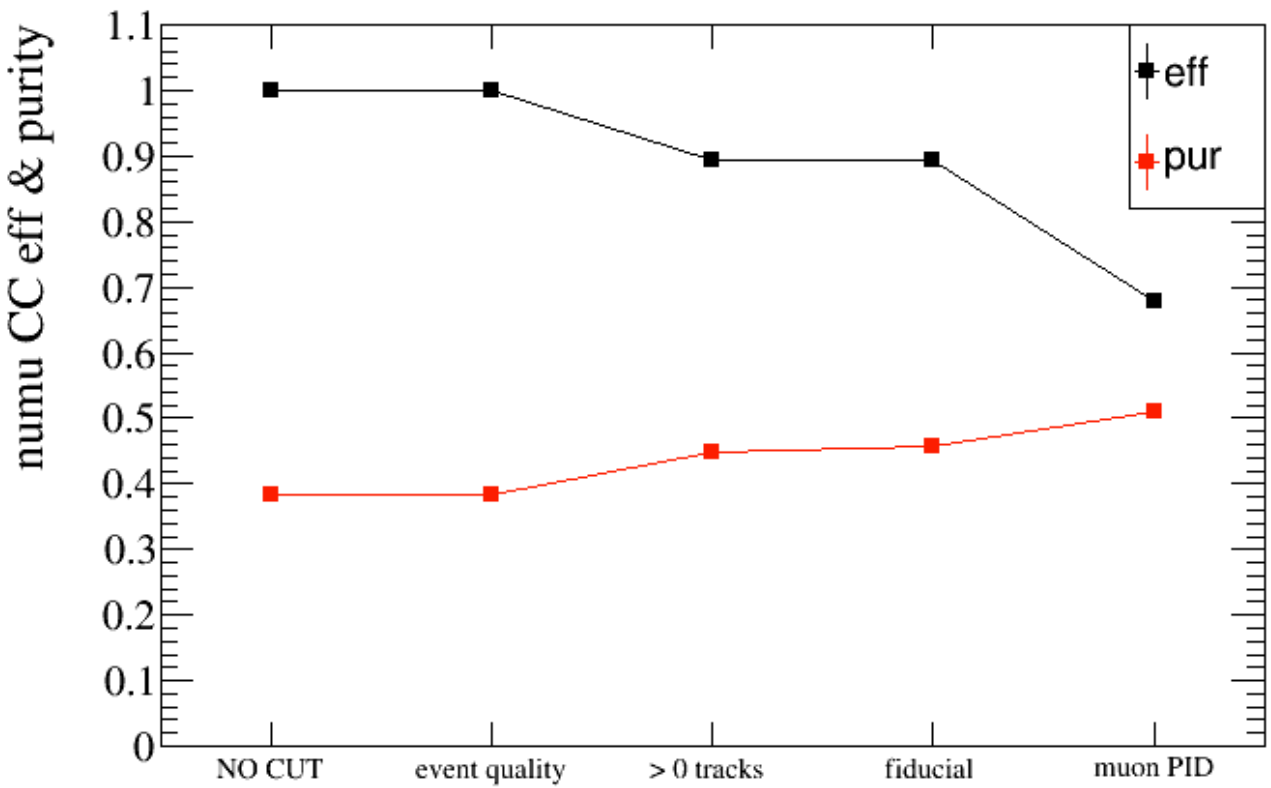
```
//********************************************************
bool MuonPIDCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//********************************************************

    (void)event;

    // Cast the ToyBox to the appropriate type
    ToyBoxDUNE& box = *static_cast<ToyBoxDUNE*>(&boxB);

    if (!box.MainTrack) return false;
    if (static_cast<AnaParticle*>(box.MainTrack)->AveragedEdx>0.5) return false;
    return true;
}
```



Efficiency and purity

- One can also plot the efficiency and purity of the selection

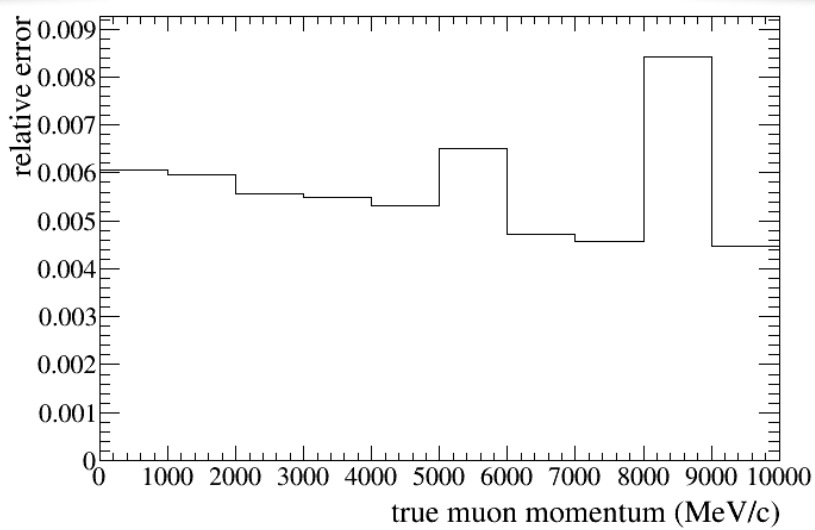
```
root [20] DataSample mc("test.root");
root [21] draw.SetTitleY("numu CC eff & purity");
root [22] draw.DrawEffPurVSCut(mc,"reactionCC==1");
```



Systematics

- We have added two systematics (see backup) as example:
  - a weight systematic: **detector mass**
  - a variation systematic: **dE/dx resolution**
- They are run by default in the example
- To produce a plot with the relative systematic error

```
root [3]   draw.SetTitleX("true muon momentum (MeV/c)");
root [4]   draw.DrawRelativeErrors(all_syst,"selmu_truemom",10,0,10000,"accum_level>3","","SYS");
average differencial error = 0.00581702
```



Next steps for ProtoDUNEs

- The first thing is to get a set of ProtoDUNE-SP AnaTree files from MCC6
  - Make sure the existing converter (developed for 35t files) works for ProtoDUNE-SP. If not develop a new one
  - Develop a simple selection for those files as an example
- What about the DP prototype ?
  - Similar files will be only produced in MCC7
  - Is there any other format we can use for the moment ?
  - We could easily develop a converter for those files
- Guess it is very important to get the system working for both prototypes ASAP

Outlook

- HighLAND has been crucial for T2K ND
  - Has decreased considerably the learning curve and speed up analysis development
- It seems it could be useful for DUNE as well, including its prototypes
- We have now a prototype for DUNE
  - Feedback from DUNE users will be essential
- Next steps are:
  - Improve converter for LArSoft files
  - Improve documentation
- A tutorial could be given at some point

backup

Framework structure

- **HighLAND** framework is divided in two sets of packages:
  - **PSyChE** is the core of **HighLAND**, devoted to event selection and systematic error propagation for
    - External fitters (osc., x-section, ...) and HighLAND
  - **HighLAND**, which extends **PSyChE** with
    - Event loop, extended event model, more input converters, corrections and drawing tools
    - Physics Analysis packages with specific selection/systematics and customized output trees

- **HighLAnD = High Level Analysis in Dune**
- **PSyChE = Propagation of Systematics and Characterization of Events**

Input Data

- The Input data for **HighLAND** can have any format (in T2K we use root files)
  - For **DUNE** either **Art event** or **AnalysisTree** or ...
- The input file information is dumped into the HighLAND data classes (event model) by **InputConverter**'s, one for each input file type
- Once the information is propagated to those data classes, all analyses are independent of the input format
- Input files should be as small as possible to gain in speed and portability
  - **HighLAND** provides a new level in **data reduction**

Output file

- HighLAND produces an output tree called micro-tree file, which contains several root trees
  - **default**: standard analysis tree containing reco+truth info for all events passing a given cut. This is the nominal selection
  - **syst1, syst2, ..., all_syst :** as the default tree but containing info for each toy experiment for one or several systematics enabled
  - **truth**: tree used to compute efficiencies, containing truth info for all signal events (passing or not the selection)
  - **config**:(Single entry) how the analysis was run
    - Systematics/corrections enabled, input file name, software version, documentation about variables in the analysis tree, etc
  - **header**: (Single entry) POT info

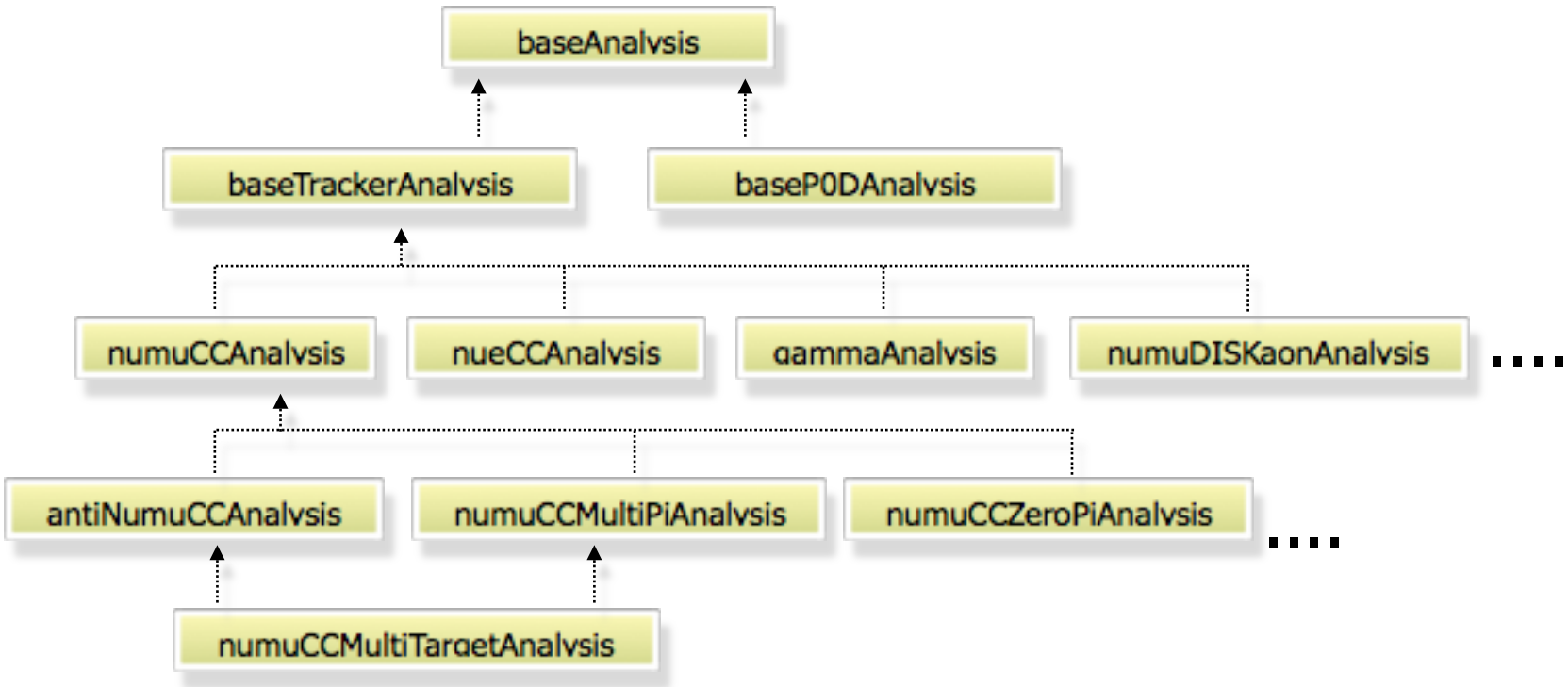
The code

- There are 157 header files, most of them with an associated source file
- The main framework packages are psycheCore and highlandTools (57 header files)
- Utils and IO packages contain some general code but also detector specific code. We could probably split them

package name	# header files	detector dependent
psycheCore	34	**NO**
psycheUtils	17	some
psycheIO	2	YES
psycheSelections	17	YES
psycheSystematics	31	YES
psycheSteering	2	some
highlandTools	23	**NO**
highlandUtils	3	some
highlandCorrections	12	YES
highlandIO	8	some
baseAnalysis	8	YES
TOTAL	157	

A hierarchy of analyses

Analysis hierarchy in T2K ND

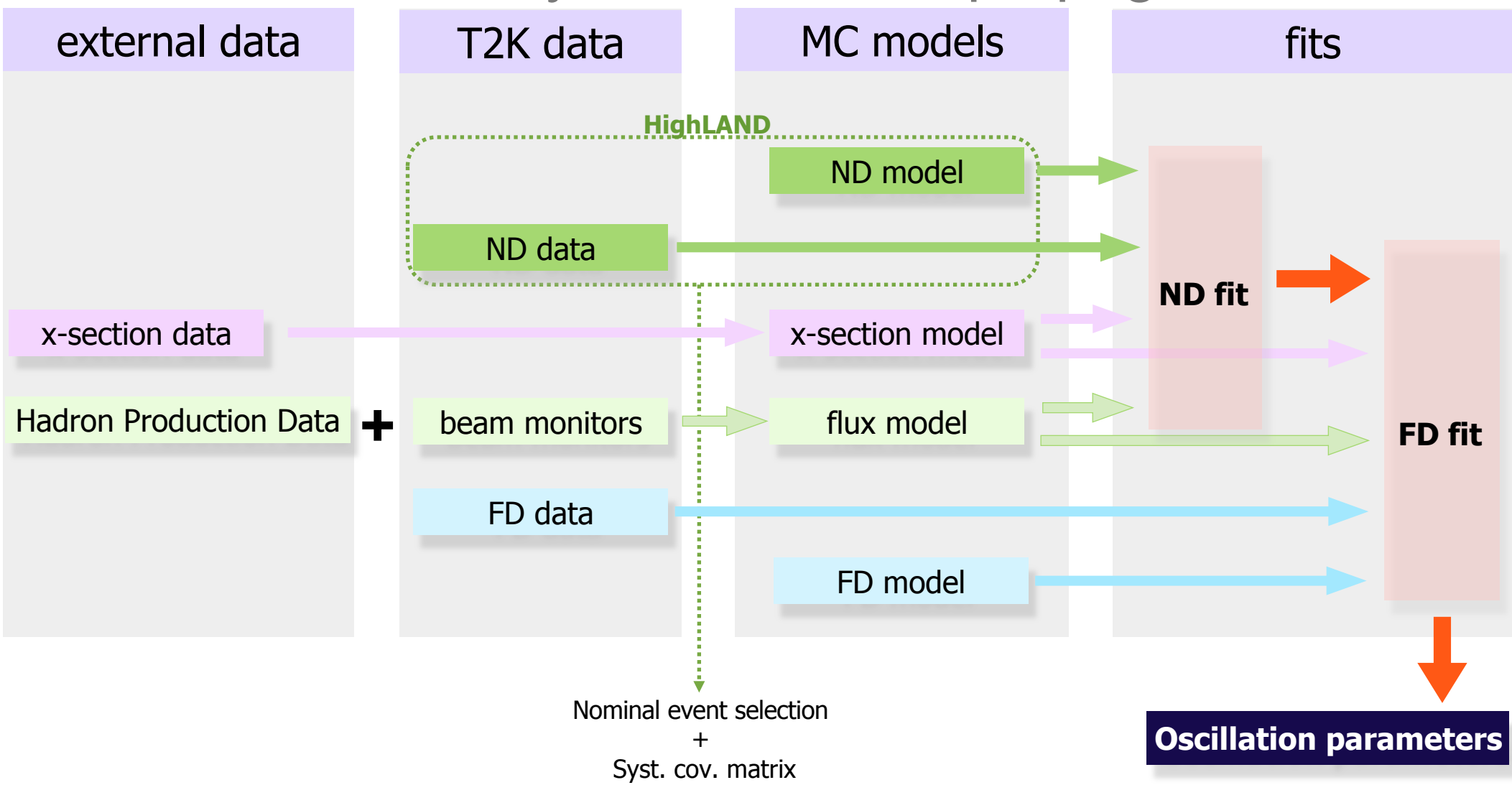


only few packages shown here (>20)

- In most cases packages down in the hierarchy perform selections that are subsamples of the packages above
- But this is not mandatory, you can just use functionality from another package

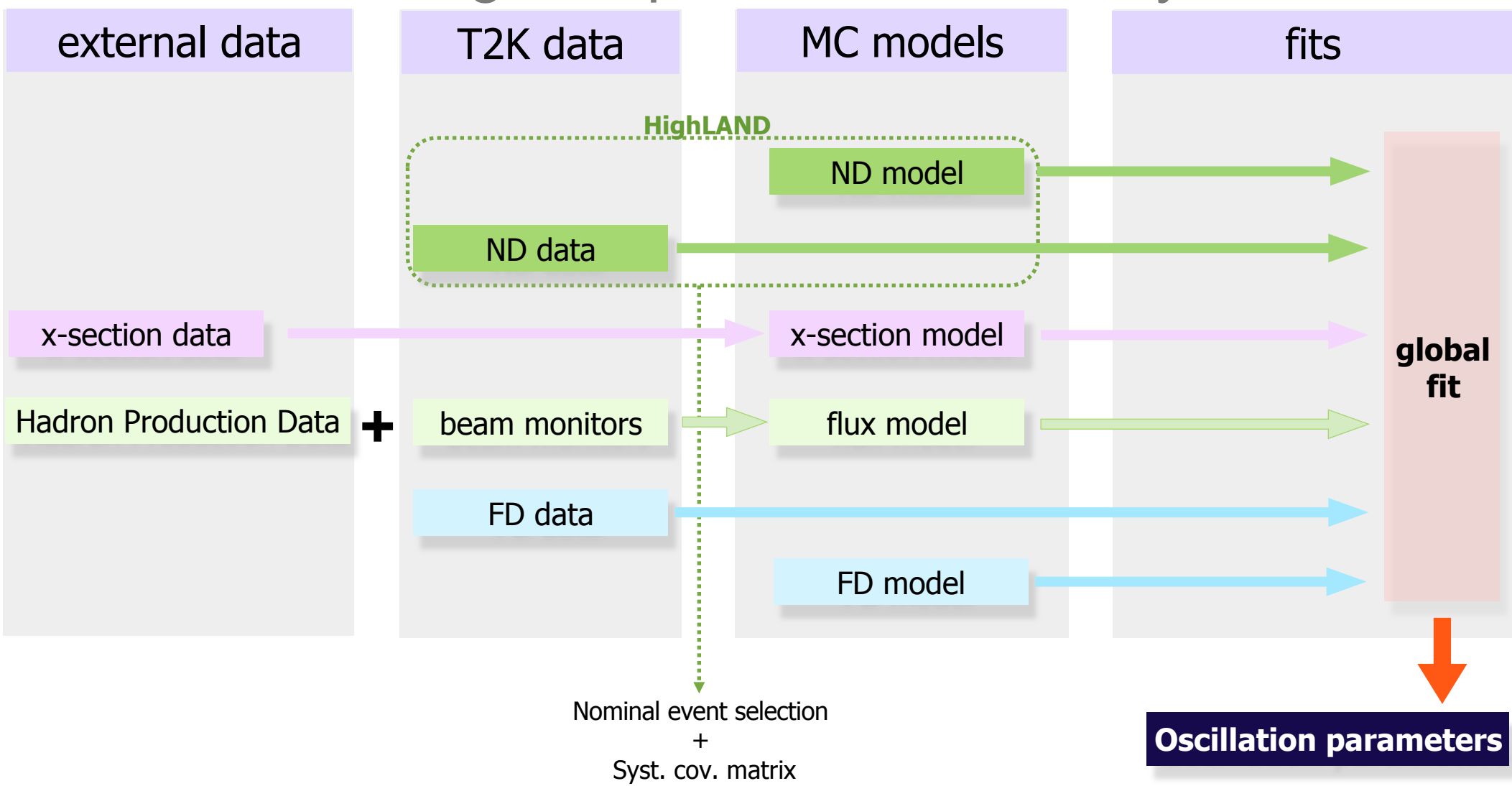
VaLOR oscillations in T2K

- In T2K HighLAND is only used for ND event selection and systematic error propagation



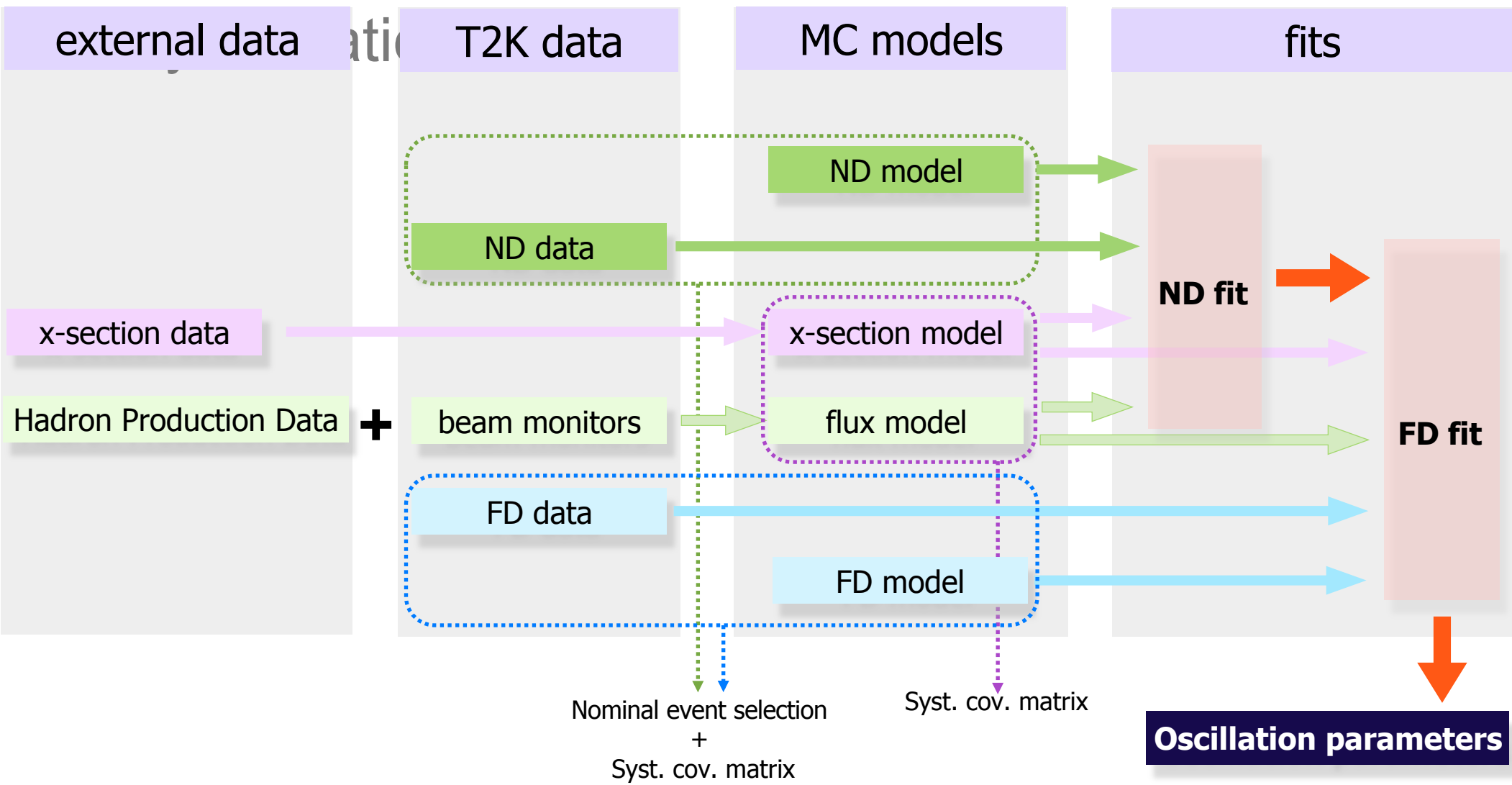
MaCh3 fit in T2K

- We have another fitter using a Markov Chain MC method using all inputs simultaneously



In DUNE

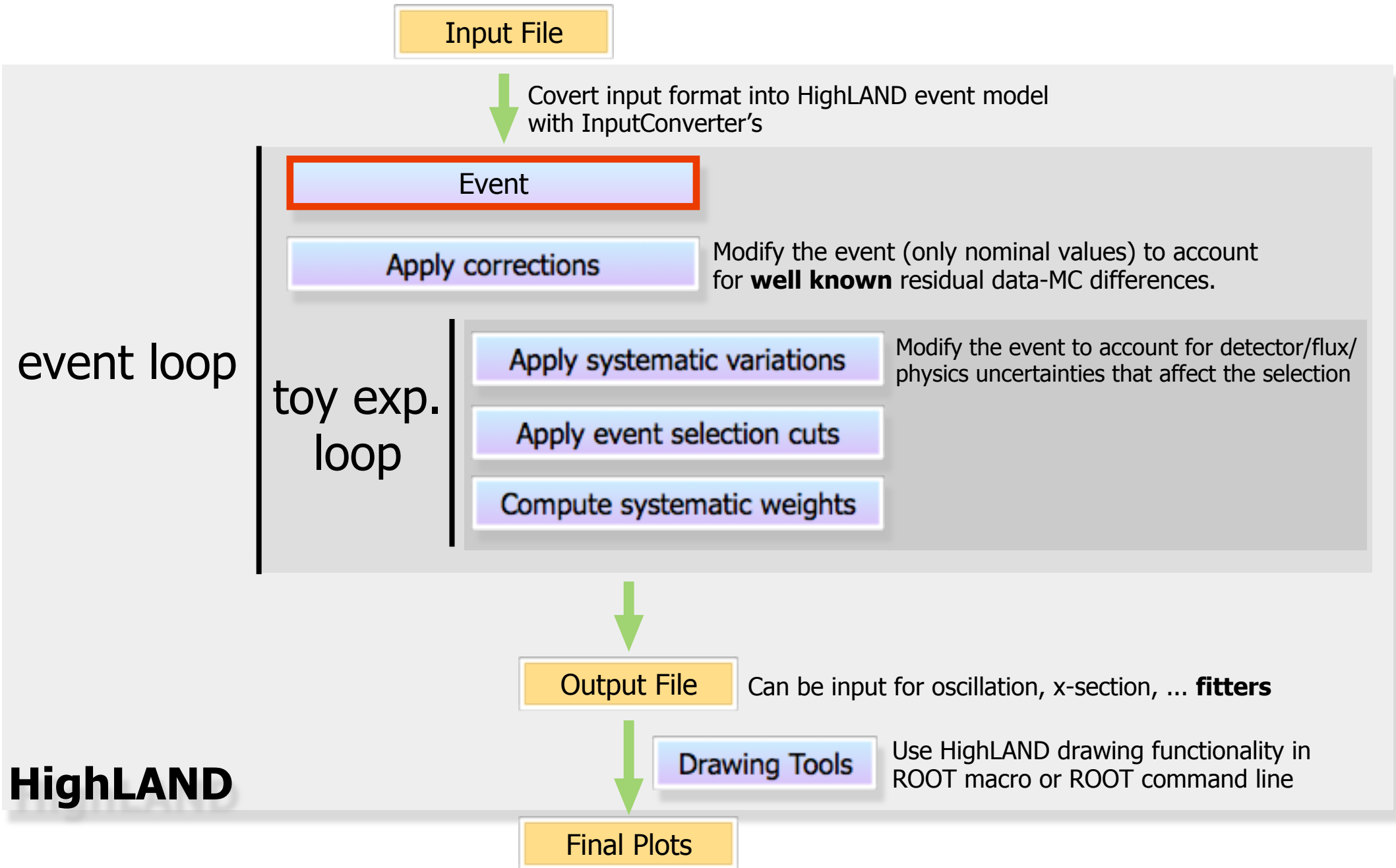
- We could also use it for FD selection/systematics + correlations with ND, x-section and flux



Two ways of using HighLAND

- In T2K we have to ways of propagating systematics with **HighLAND** for oscillation analyses
  1. HighLAND can be used to produce a nominal selection + a covariance matrix, which will be later used by fitters
     - Toy Experiments (random throws) are generated internally by HighLAND
     - A cov. matrix assumes gaussian errors !!!!
  2. Or can be used directly by the fitters:
     - Toy experiments generated by fitters. For each toy HighLAND is called by the fitter to get the results of the selection
     - In T2K we want to move to this

Analysis flow

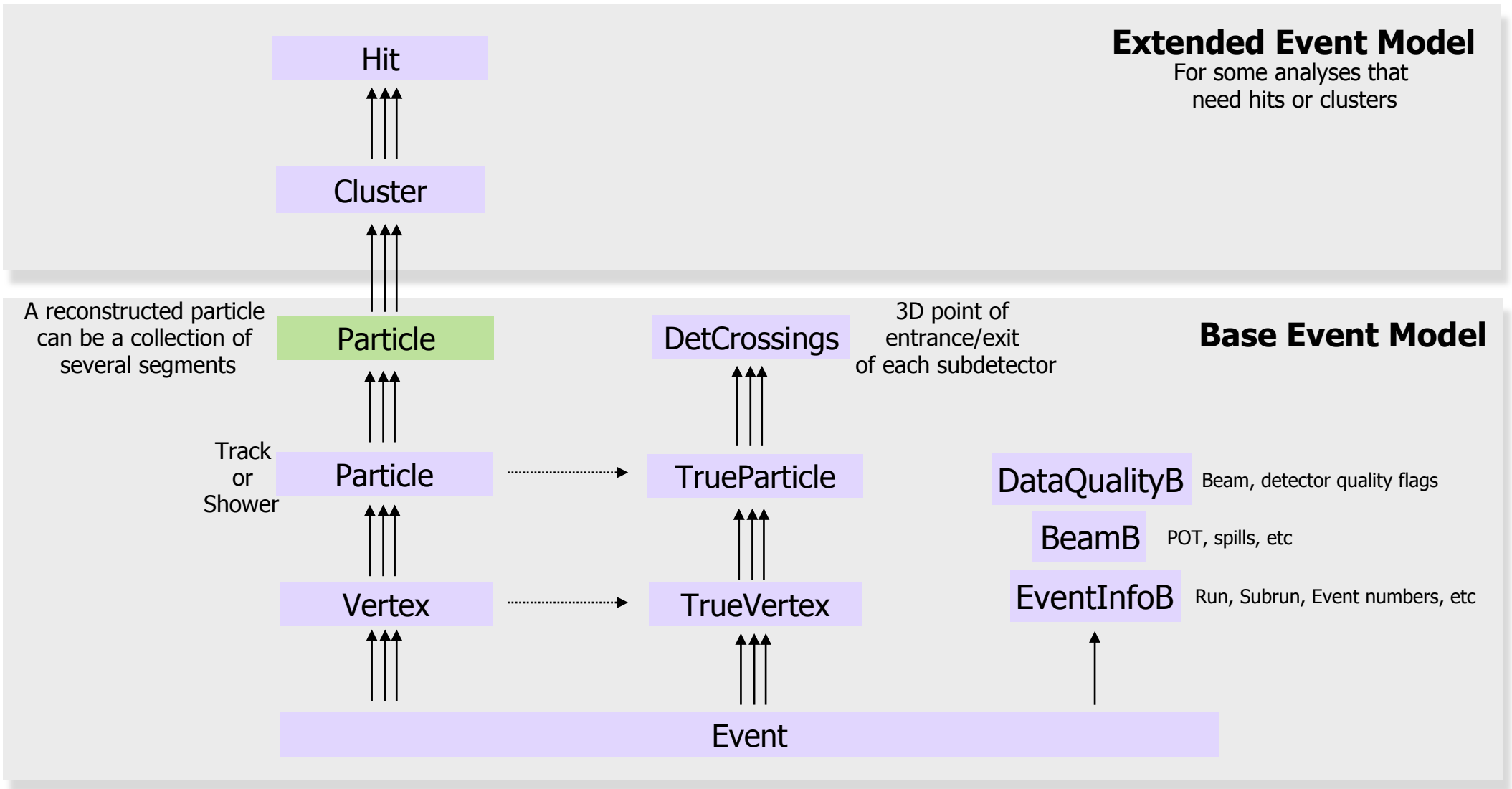


Event Model

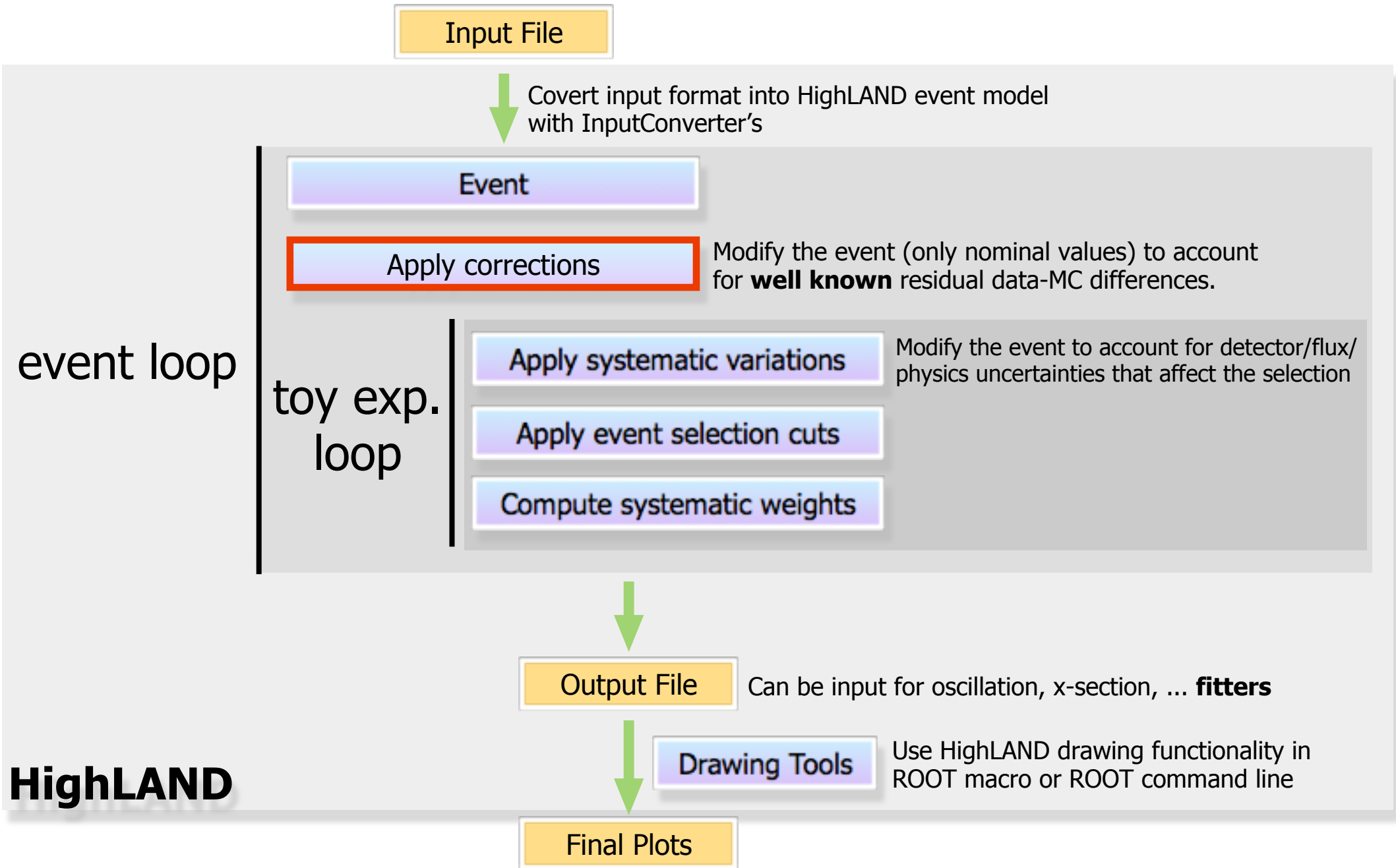
- There is a set of basic Data Classes common to all inputs that define the HighLAND event model
- But we can have an extensible data structure inheriting from the base one
  - Various types of analyses needing more complex objects
    - In the same detector, in different detectors, ...
  - etc
- Conversion from base to derived classes should be done with **static_cast** (it's fast)

T2K-HighLAND event model

In T2K we have something like this
but it could be different for DUNE



Analysis flow



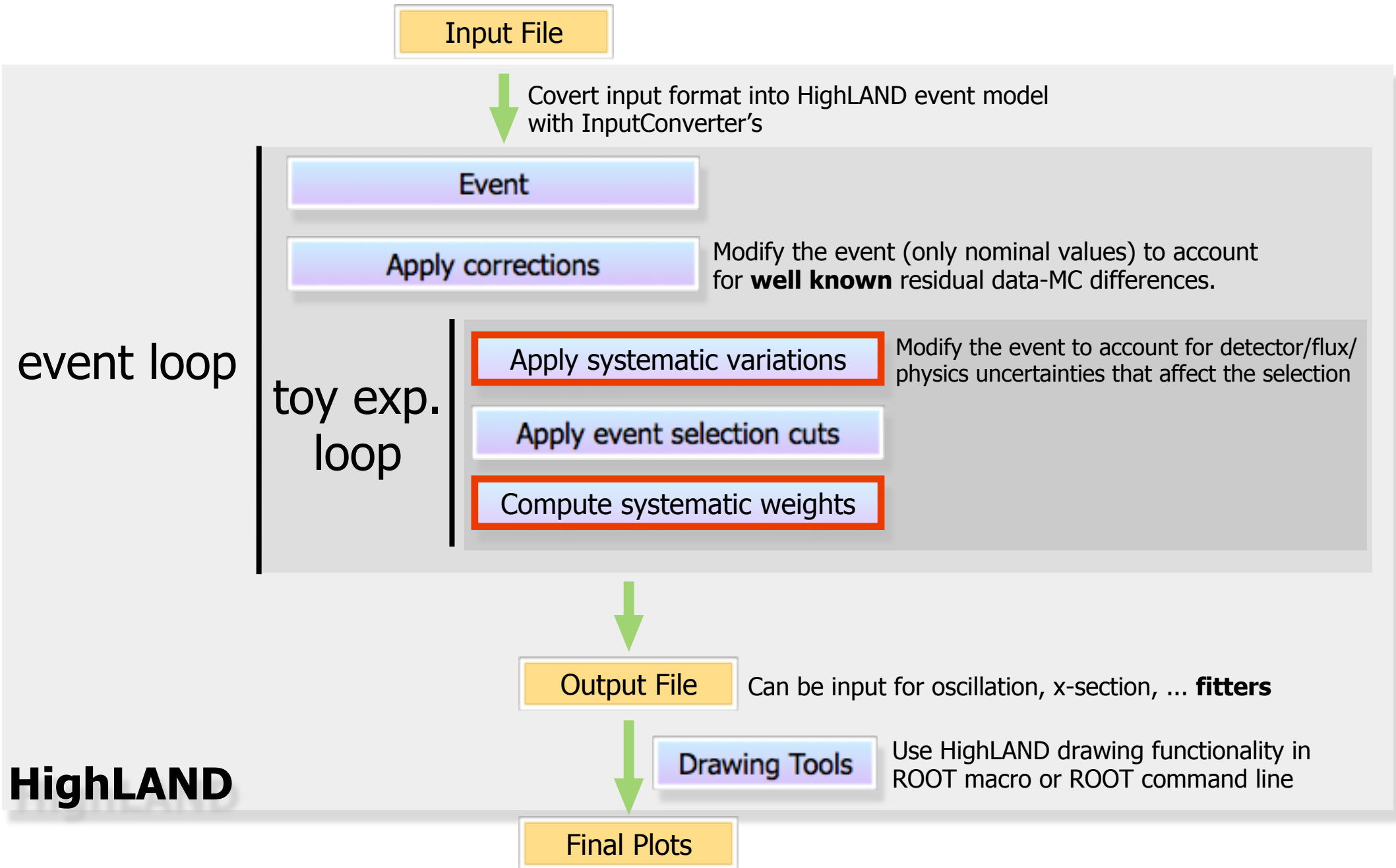
Corrections

- In this step data and/or MC are corrected such that they match each other in detector performance:
- As an example let's imagine that **momentum scale** is different in data and MC
  - Imagine we have a way to quantify this difference
  - We can either propagate this difference as a systematic or correct for it introducing as a systematic only the error on the correction
  - The correction would consist in scaling the momentum of all tracks in the MC to match the momentum scale in data. So we change the **nominal value** of the momentum for each track

Detector/reco optimization

- We can use the **correction** functionality to tweak the output of the reconstruction and perform analysis (selection +systematics) without rerunning the reconstruction
  - Change point or momentum/energy resolution
  - Change momentum/energy scale
  - Change PID information
  - ...
- In that way we can **optimize the detector or reconstruction parameters** without rerunning the reconstruction (at least 3 orders of magnitude slower)

Analysis flow



Systematics

- Systematics are propagated numerically using toy-experiments (pseudo-experiments or virtual analyses)
- Each toy-experiment is defined by a set of random throws (one for each systematic parameter)
- The covariance of the number of events selected in a given bin is computed in the usual way:

$$C_{ij} = \frac{1}{N_{toys}} \sum_{t=1}^{N_{toys}} (N_i^t - \overline{N}_i)(N_j^t - \overline{N}_j)$$

events in bin i for toy t

$$N_i^t = \sum_{e=1}^{N_{events}} W_{e,i}^t$$

average over toys

$$\overline{N}_i = \frac{1}{N_{toys}} \sum_{t=1}^{N_{toys}} N_i^t$$

Two types of systematics

- **Variations**: The event is modified taken into account the set of systematic parameters for a particular toy experiment. Then the entire analysis proceeds on the modified event. For example:
  - Momentum scale (smear the momentum of all tracks in MC around the nominal, see slide 16)
- **Weights**: a weight (which is one by default) is assigned to each event. This weight is computed using event truth/reco info and the systematic parameters for the current toy. This is done in two cases:
  - when the variation method is not possible
    - Imagine for example the track finding efficiency in one of the TPCs. If the efficiency is larger in data than in MC we can't easily add a new track into the MC
  - for global normalization parameters (flux, target mass, etc.)
- In HighLAND they inherit from base class **EventWeightBase**

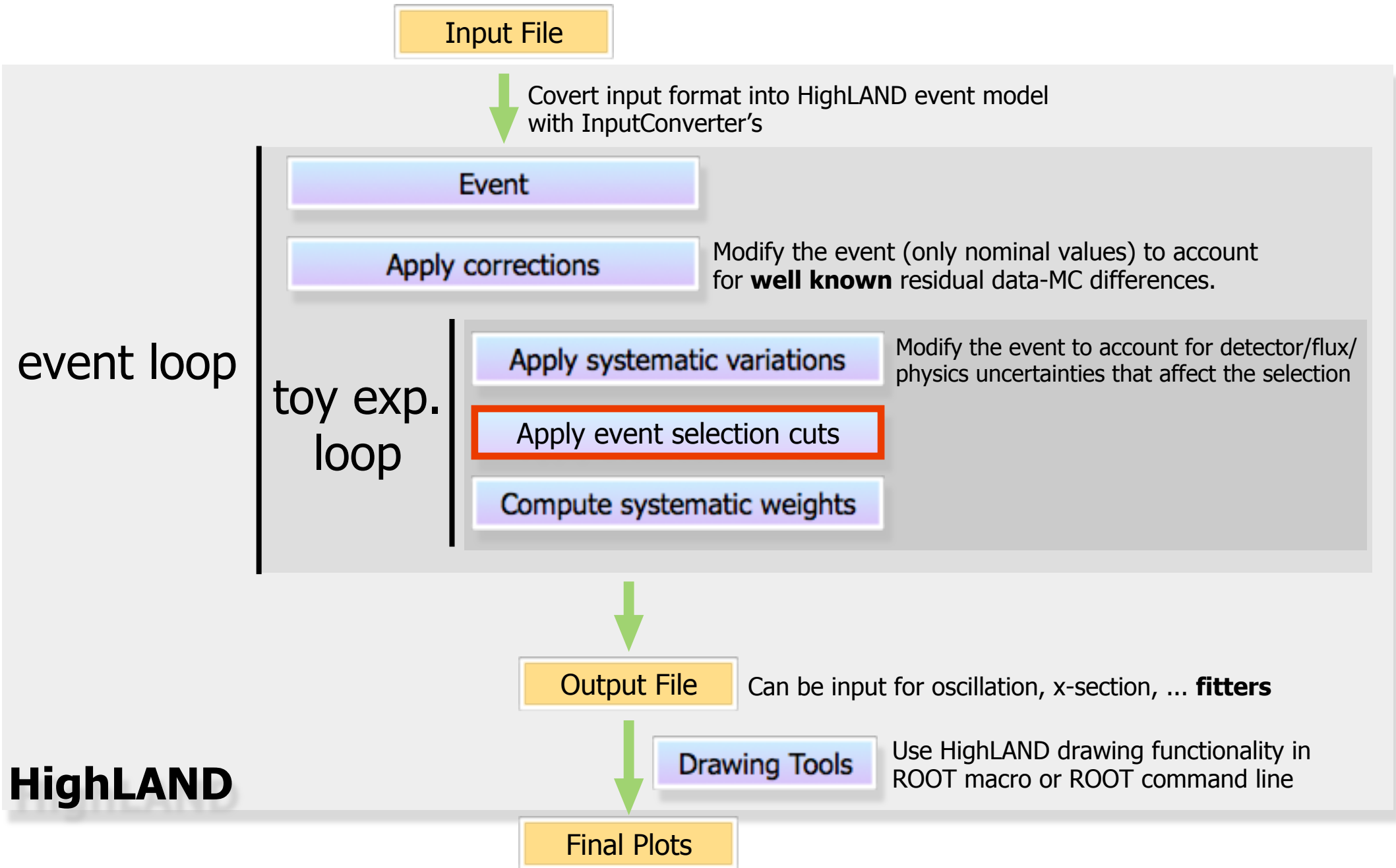
List of systematics in T2K

- This is the list of 31 detector systematics implemented in HighLAND for T2K
- Not all selections use the same systematics but most of them are common to many selections

```
BFieldDistortionSystematics.hxx
ChargeIDEffSystematics.hxx
ECalEMEnergyResolSystematics.hxx
ECalEMEnergyScaleSystematics.hxx
ECalEMEnergySystematicsBase.hxx
ECalPIDSystematics.hxx
ECalTrackEffSystematics.hxx
FGDECalMatchEffSystematics.hxx
FGDECalSMRDMatchEffSystematics.hxx
FGDHybridTrackEffSystematics.hxx
FGDMassSystematics.hxx
FGDPIDSystematics.hxx
FGDTrackEffSystematics.hxx
FluxWeightSystematics.hxx
MichelElectronEffSystematics.hxx
MomRangeResolSystematics.hxx
MomentumResolSystematics.hxx
MomentumScaleSystematics.hxx
```

```
OOFVSystematics.hxx
PileUpSystematics.hxx
SIPionSystematics.hxx
SIProtonSystematics.hxx
SandMuonsSystematics.hxx
TPCClusterEffSystematics.hxx
TPCECalMatchEffSystematics.hxx
TPCFGDMatchEffSystematics.hxx
TPCP0DMatchEffSystematics.hxx
TPCPIDSystematics.hxx
TPCTrackEffSystematics.hxx
TPCVariationSystematics.hxx
ToFResolSystematics.hxx
```

Analysis flow



Event Selection I

- It's a collection of "steps" (cuts and actions)
  - Each step inherits from the base class **StepBase**
  - It has a single method **Apply**, which returns true or false (only relevant for cuts)
- Each selection inherits from **SelectionBase**, which has a main mandatory method **DefineSteps**

```
//********************************************************************
void numuCCSelection::DefineSteps(){
//********************************************************************

    // Cuts must be added in the right order
    // last "true" means the step sequence is broken if cut is not passed (default is "false")
    AddStep(StepBase::kCut,    "event quality (good beam/detector)", new EventQualityCut(),           true);
    AddStep(StepBase::kCut,    "> 0 tracks ",                        new TotalMultiplicityCut(),      true);
    AddStep(StepBase::kAction, "find lepton candidate",              new FindCandidateAction());
    AddStep(StepBase::kAction, "find vertex",                        new FindVertexAction());
    AddStep(StepBase::kCut,    "track quality + fiducial volume",    new TrackQualityFiducialCut(),   true);
    AddStep(StepBase::kAction, "find veto track",                    new FindVetoTrackAction());
    AddStep(StepBase::kCut,    "external veto",                      new ExternalVetoCut());
    AddStep(StepBase::kCut,    "muon PID",                           new MuonPIDCut());

    // This is a selection with a single branch, but each branch should have am alias
    SetBranchAlias(0,"trunk");
```

Event Selection II

- Example of action (fills the box ...)

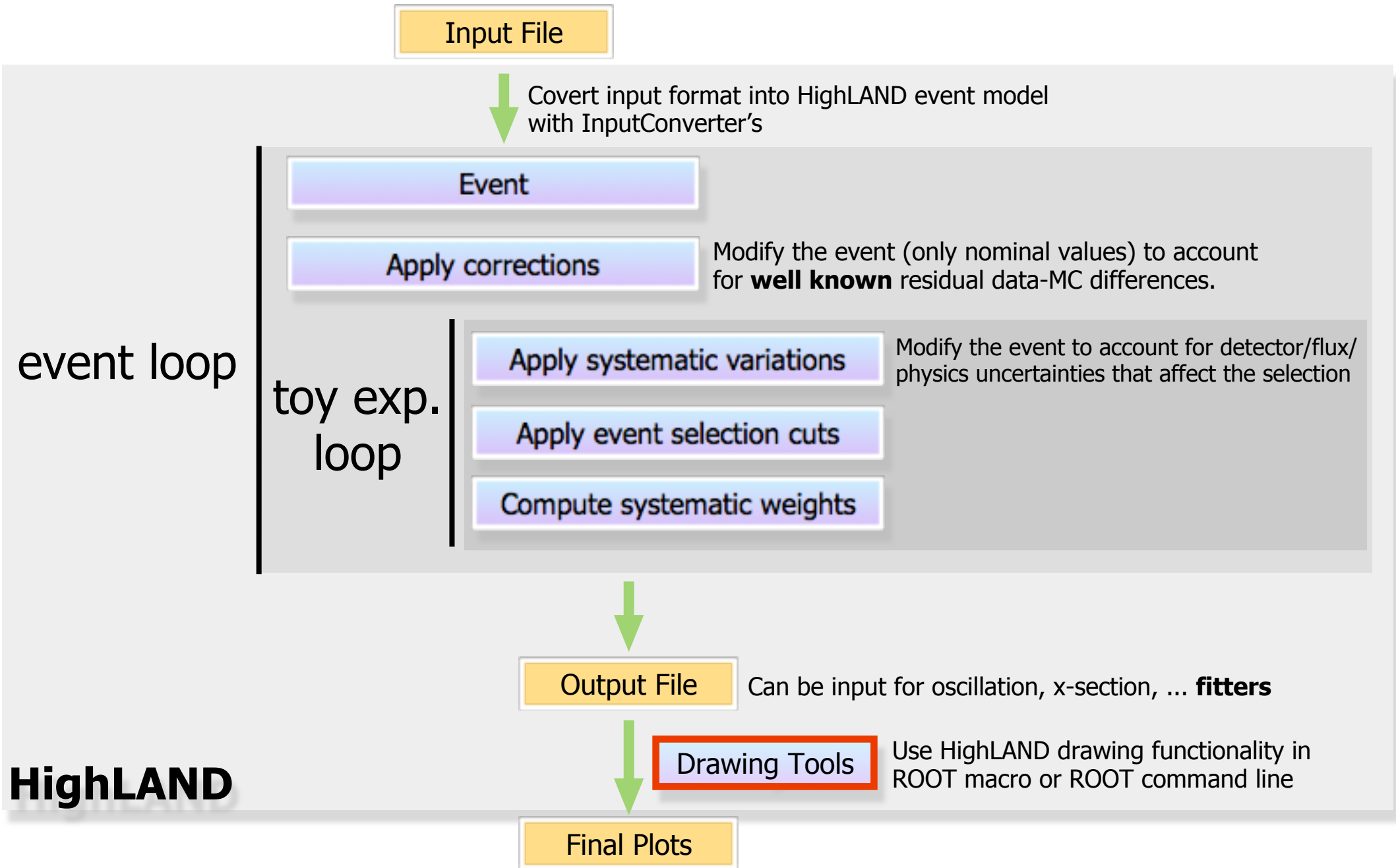
```
//*****************************************************
bool FindCandidateAction::Apply(AnaEventB& event, ToyBoxB& box) const{
//*****************************************************

    // Find leading tracks with good quality and only in FGD FV
    cutUtils::FindLeadingTracks(event,box);

    // For this selection the LeptonCandidate track is the HMN (Highest Momentum Negative) track
    box.LeptonCandidate = box.HMNtrack;
    return true;
}
```

- Example of cut (uses the filled box ...)

```
//*****************************************************
bool MuonPIDCut::Apply(AnaEventB& event, ToyBoxB& box) const{
//*****************************************************

  // LeptonCandidate must exist
  if (!box.LeptonCandidate) return false;

  // And it should have a valid momentum value
  if (box.LeptonCandidate->Momentum < 0.) return false;

  // Apply the PID cut in the utilities namespace to the LeptonCandidate
  return cutUtils::MuonPIDCut( box.LeptonCandidate );
}
```

The box is used to pass info from one step to another

Analysis flow



DrawingTools

- This is one of the framework classes which can be accessed from a ROOT macro or command line
- It is initialized with a micro-tree file (HighLAND output)
- When opening a root session the HighLAND classes are already visible so you just do

```
root [1] DrawingTools draw("microtree.root")
```

- Now you can start doing plots

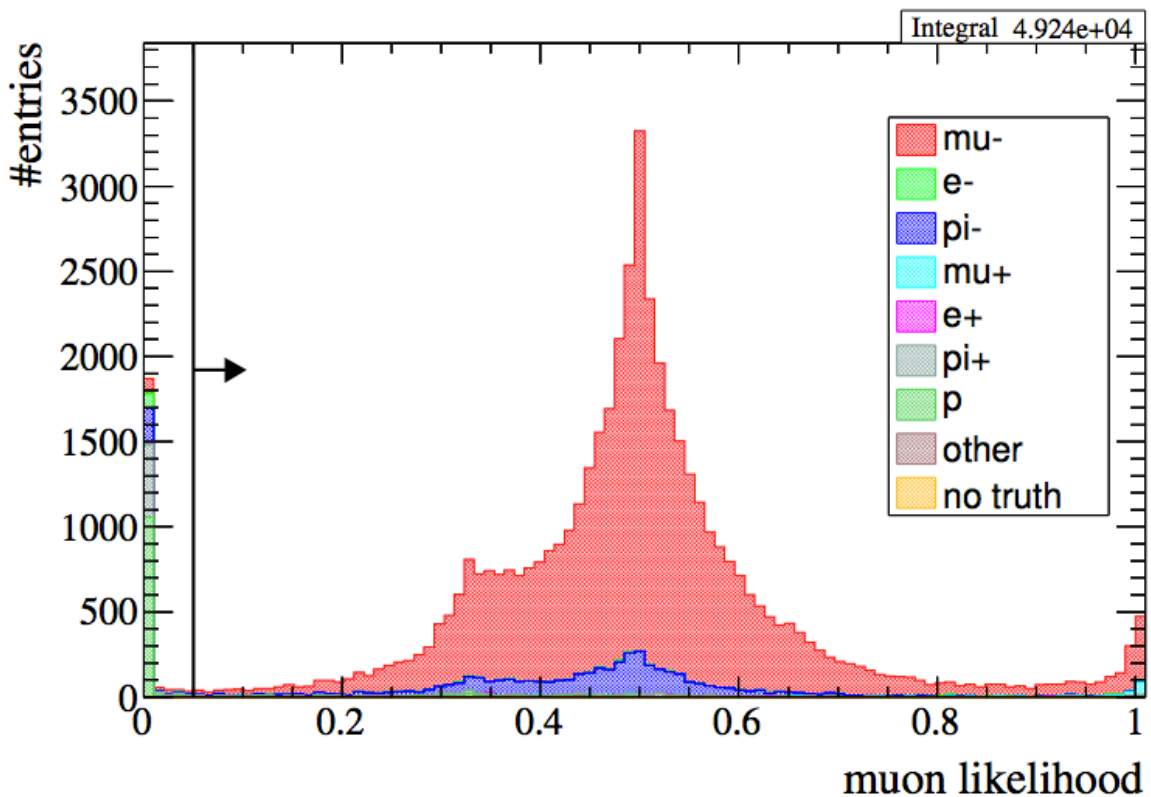
Distributions

- This plot shows the muon PID likelihood before the muon PID cut, broken down in “particle” categories

```
root [2] draw.SetTitleX("muon likelihood")
root [3] draw.SetTitleY("# entries")
root [4] draw.Draw(default,"selmu_likemu",100,0,1,"particle","accum_level>4")
```

tree name | variable to plot | binning | color category | events passing cut 4

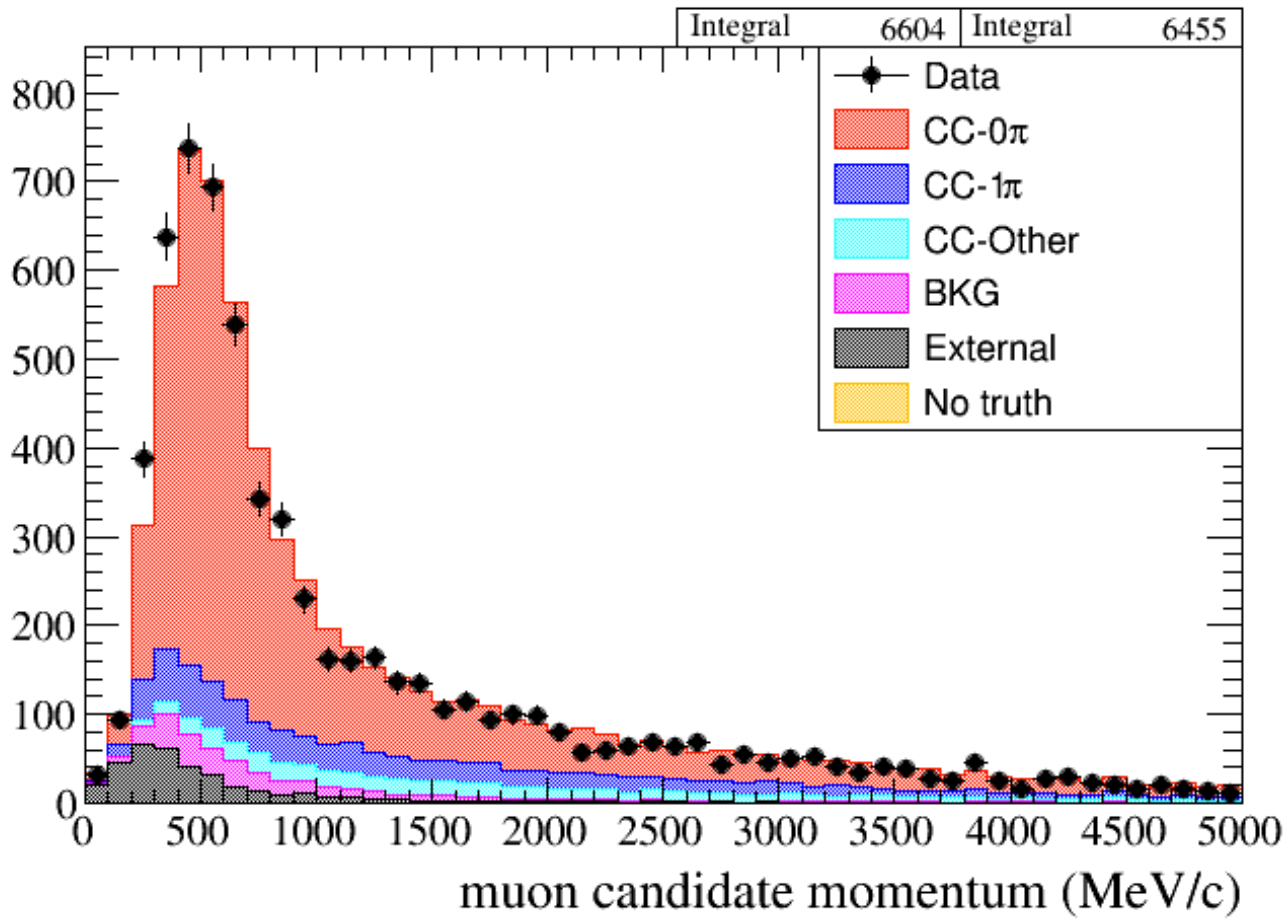
```
root [5] draw.DrawCutLineVertical(0.05,true,"r")
```



Data/MC comparisons

- We initialize a DataSample class with a micro-tree file

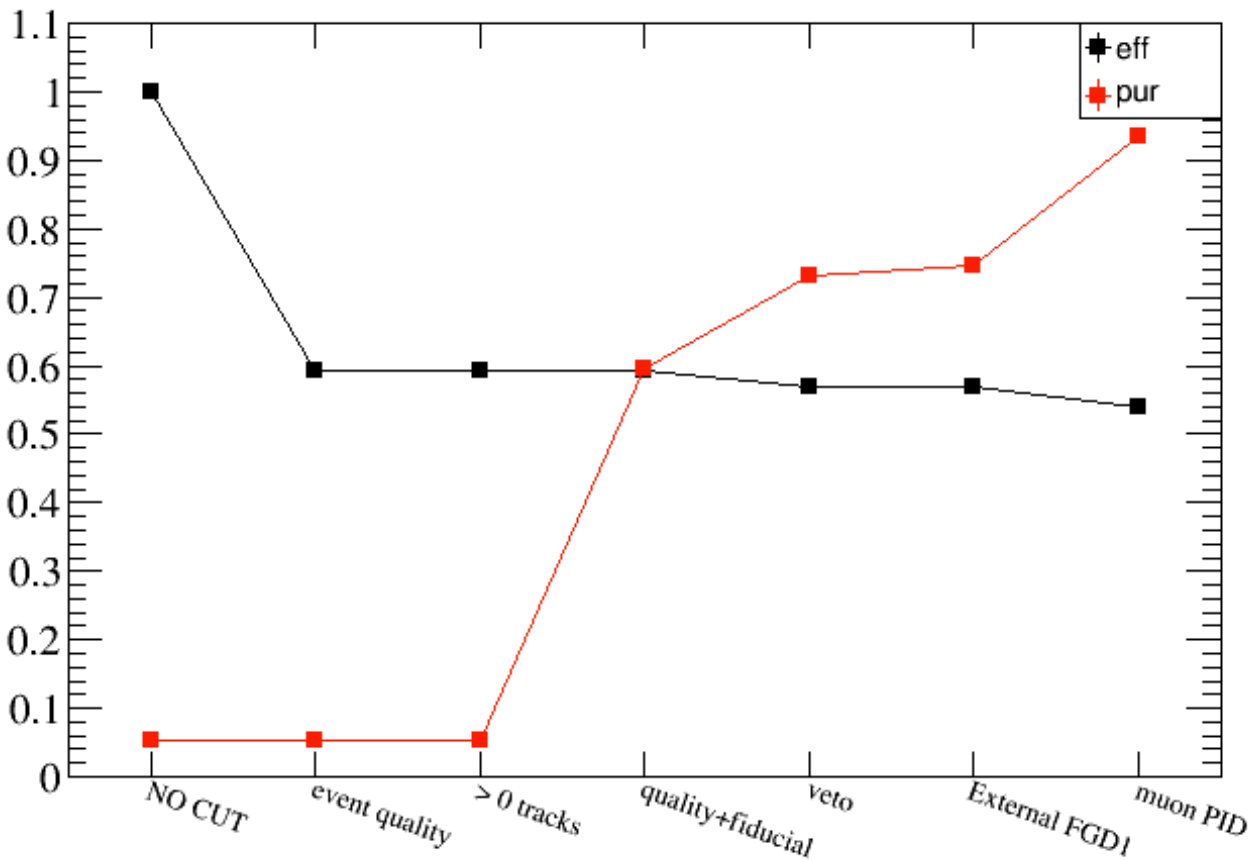
```
root [1] DrawingTools draw("data.root")
root [2] DataSample mc("mc.root")
root [3] DataSample data("data.root")
root [4] draw.SetTitleX("muon candidate momentum (MeV/c)")
root [5] draw.Draw(data,mc,"selmu_mom",50,0,5000,"topology","accum_level>5")
```



Efficiencies & purities

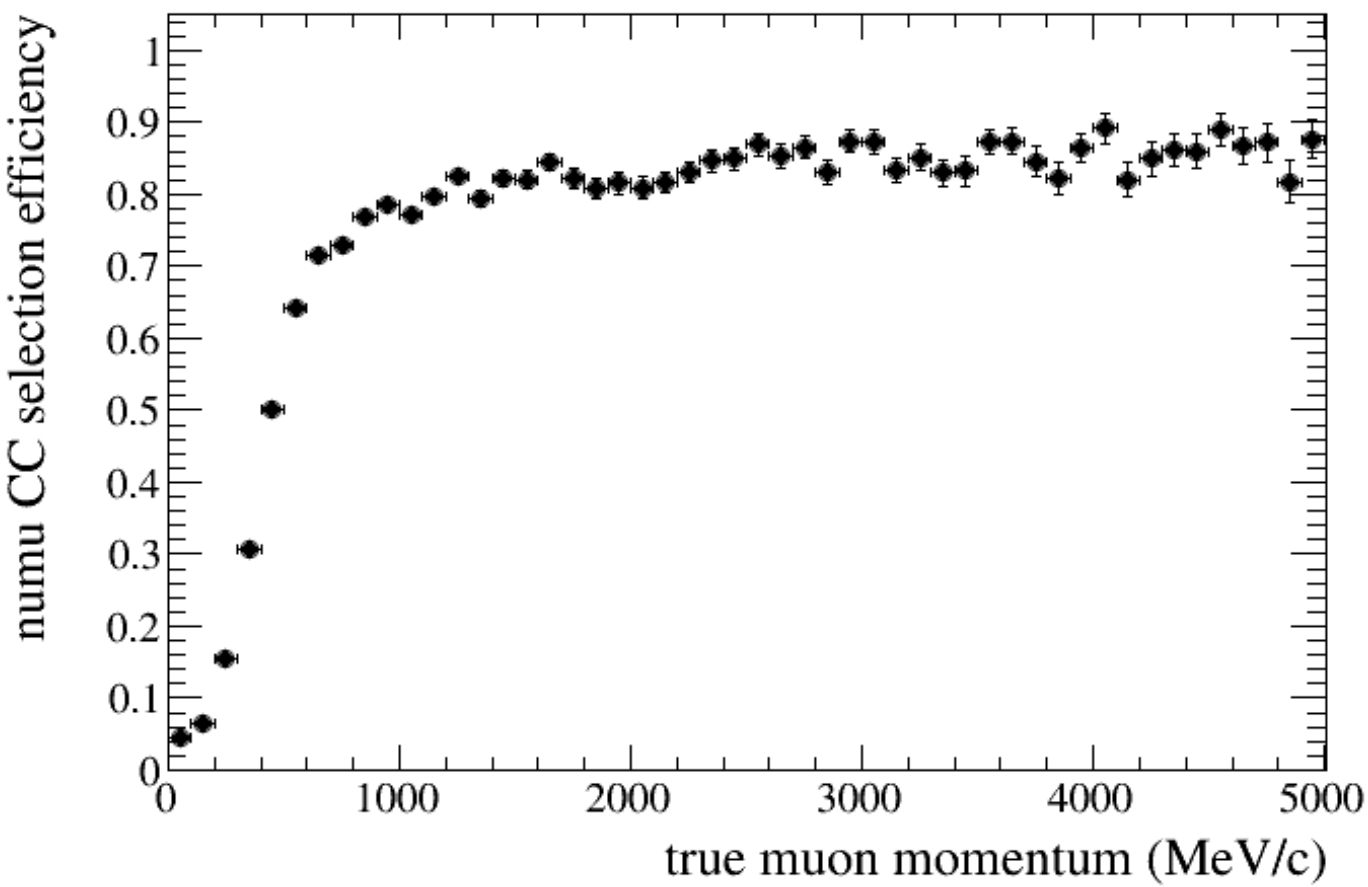
- Efficiency and purity after each cut in the selection

```
root [27] DataSample mc("mc.root")
root [28] draw.SetTitleY("")
root [29] draw.DrawEffPurVSCut(mc,"reaction>=0 && reaction<=4")
```



- Efficiency as a function of true muon momentum

```
root [18] draw.SetTitleX("true muon momentum (MeV/c)")
root [19] draw.SetTitleY("numu CC selection efficiency")
root [20] draw.DrawEff(truth,"truemu_truemom",50,0,5000,"accum_level>5","reaction>=0 && reaction<=4")
```



Using Experiment class

```
Experiment exp("t2k");
```

Create Experiment

```
DataSample* data2a  = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/data2a_5F.root");
DataSample* data2w  = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/data2w_5F.root");
DataSample* data3b  = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/data3b_5F.root");
DataSample* data3c  = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/data3c_5F.root");
DataSample* data4w  = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/data4w_5F.root");
DataSample* data4a  = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/data4a_5F.root");

DataSample* mc2a    = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/mc2a_5F_allsyst.root");
DataSample* mc2w    = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/mc2w_5F_allsyst.root");
DataSample* mc3a    = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/mc3a_5F_allsyst.root");
DataSample* mc4a    = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/mc4w4a_5F_allsyst.root");
DataSample* mc4w    = new DataSample("/data3/T2K/DataDir/MicroTrees/numuCCQExsec/mc4w_5F_allsyst.root");
```

Create DataSamples for data and MC

```
SampleGroup run2a("run2a");
run2a.AddDataSample(data2a);
run2a.AddMCSample("sys",mc2a);

SampleGroup run2w("run2w");
run2w.AddDataSample(data2w);
run2w.AddMCSample("sys",mc2w);

SampleGroup run3("run3");
run3.AddDataSample(data3c);
run3.AddMCSample("sys",mc3a);

SampleGroup run4a("run4a");
run4a.AddDataSample(data4a);
run4a.AddMCSample("sys",mc4a);

SampleGroup run4w("run4w");
run4w.AddDataSample(data4w);
run4w.AddMCSample("sys",mc4w);
```

Create SampleGroups one per period

```
exp.AddSampleGroup("run2a", run2a);
exp.AddSampleGroup("run2w", run2w);
exp.AddSampleGroup("run3",  run3);
exp.AddSampleGroup("run4a", run4a);
exp.AddSampleGroup("run4w", run4w);
```

Add SampleGroups to the Experiment

Final plots with all runs

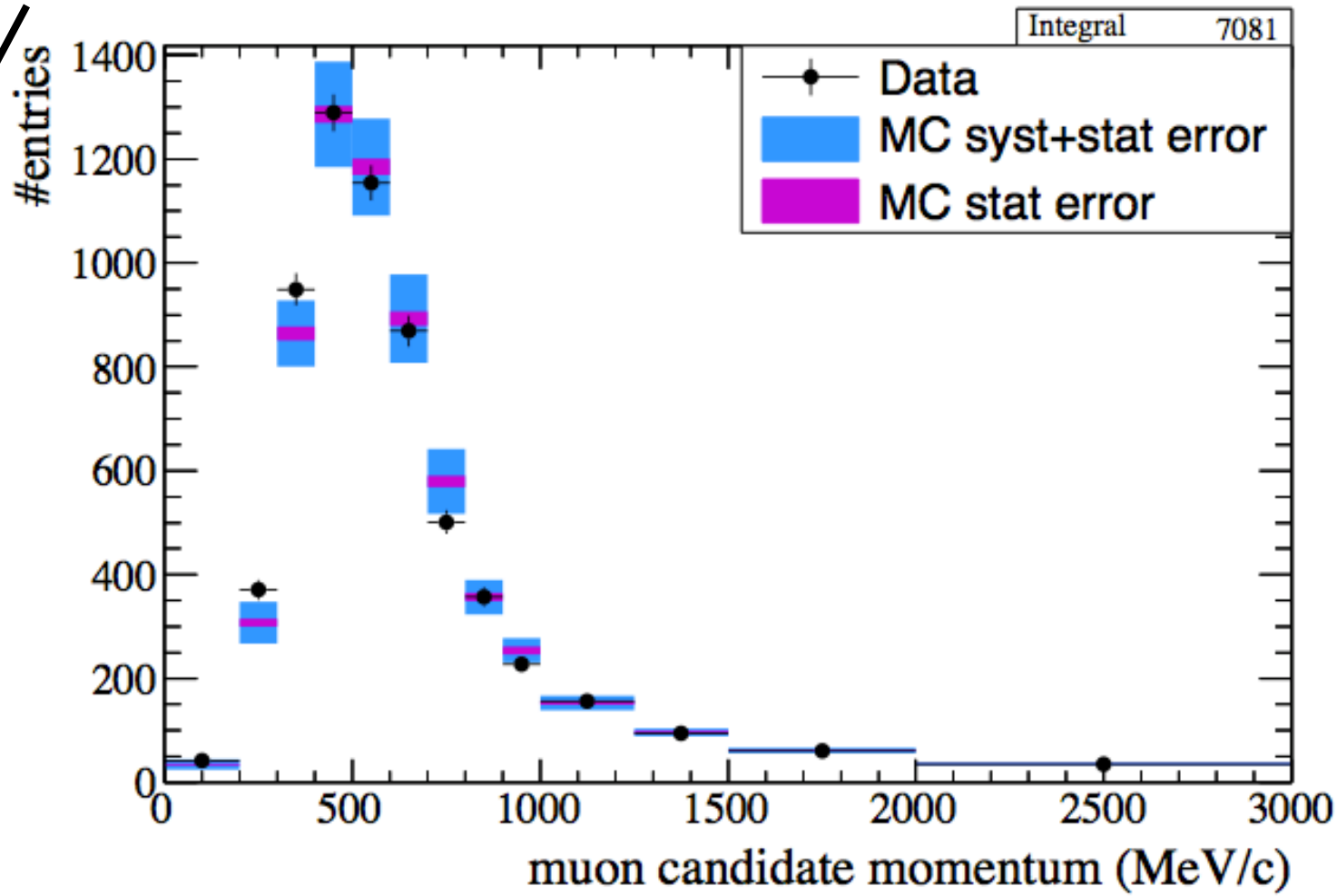
```
draw.SetTitleX("muon candidate momentum (MeV/c)");
draw.SetTitleY("#entries");
draw.SetAllMCLabel("MC stat error");
draw.SetAllMCStatLabel("MC stat+syst error");
draw.SetMCErrorColor(kAzure);
draw.Draw(exp,"selmu_mom",14,pbins,"all","accum_level[][0]>7","","SYS E2");
```

draw systematic error bars

error style for MC

using Experiment class

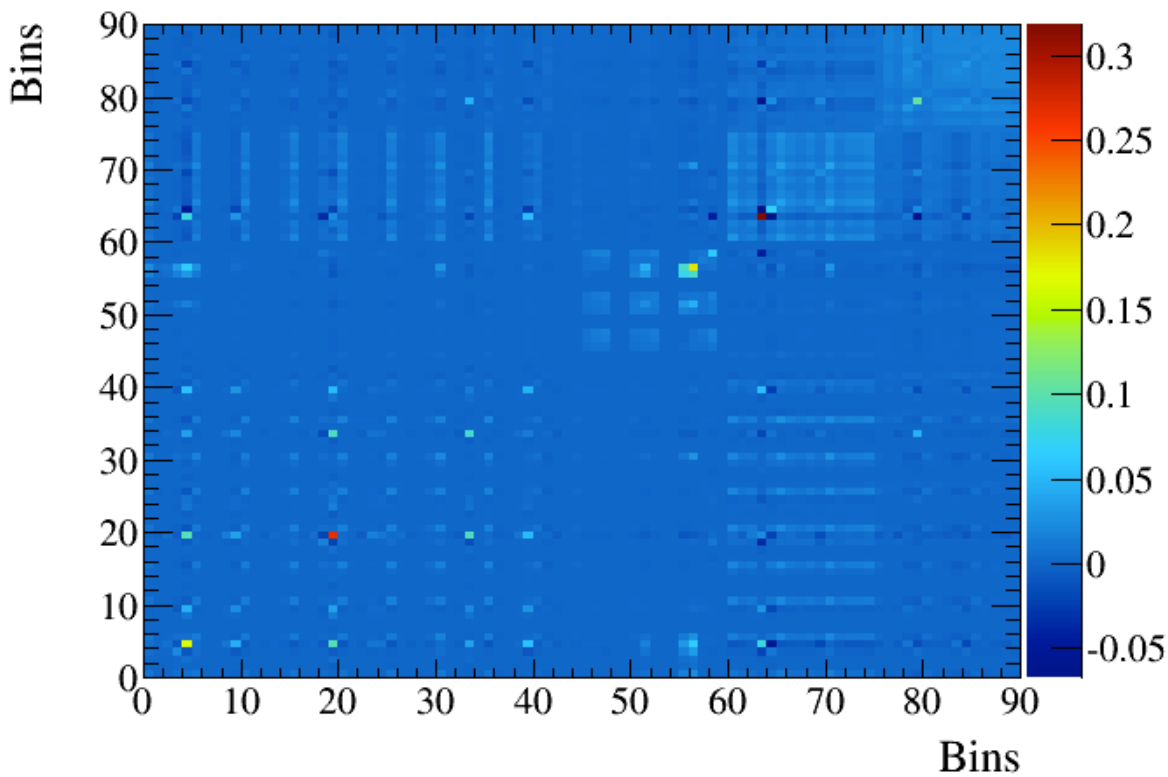
variable binning



Covariance Matrix

- binning: 3 theta x 5 momentum x 6 samples = 90 bins
- Cov matrix is **computed at plotting time** (all info in the tree). Thus the user can change cuts, binning, etc

1000 throws



official T2K ν_μCC-0π x-section

- Detailed talks at previous DUNE meetings:
  - FD sim/reco 23/11/215: https://indico.fnal.gov/conferenceDisplay.py?confId=10882
  - LBL 24/11/2015: https://indico.fnal.gov/conferenceDisplay.py?confId=10861
  - S&C 15/12/2015: https://indico.fnal.gov/conferenceDisplay.py?confId=11030