A data summary file structure and analysis tools for neutrino oscillation analysis at the NOvA experiment

C. Backhouse
Lauritsen Laboratory, California Institute of Technology, Pasadena, CA 91125, USA
E-mail: bckhouse@caltech.edu

D. Rocco
Tate Laboratory, University of Minnesota, Minneapolis, MN 55455, USA
E-mail: rocco@physics.umn.edu

Abstract. The NuMI Off-axis Neutrino Experiment (NOvA) is designed to study neutrino oscillations in the NuMI beam at Fermilab. Neutrinos at the Main Injector (NuMI) is currently being upgraded to provide 700 kW for NOvA. A 14 kt Far Detector in Ash River, MN and a functionally identical 0.3 kt Near Detector at Fermilab are positioned 810 km apart in the NuMI beam line. The fine granularity of the NOvA detectors provides a detailed representation of particle trajectories. The data volume associated with such granularity, however, poses problems for analyzing data with ease and speed. NOvA has developed a data summary file structure which discards the full event record in favor of higher-level reconstructed information. A general-purpose framework for neutrino oscillation measurements has been developed for analysis of these data summary files. We present the design methodology for this new file format as well as the analysis framework and the role it plays in producing NOvA physics results.

1. NOνA
Neutrinos at the Main Injector (NuMI) is a neutrino beam at Fermilab which will soon reach 700 kW through upgrades. The beam can run in either $\nu_\mu$ mode or $\bar{\nu}_\mu$ mode by reversing the current in its focusing horns. The NuMI Off-axis $\nu_e$ Appearance (NOvA) experiment places two functionally identical detectors in the NuMI beamline. [2] Placing the detectors off-axis at 14 mrad provides a narrow band $\nu$ energy spectrum near 2 GeV. NOvA detectors are composed of extruded PVC cells filled with liquid scintillator. A strand of wavelength shifting fiber runs down and back each cell to capture scintillation light and transmit it to an avalanche photodiode (APD). The cells are oriented into planes, with alternating planes orthogonally rotated to provide separate, interleaved $x-z$ and $y-z$ views. Fermilab is home to the Near Detector, placed 1 km from the source, while the Far Detector is located 810 km away near Ash River, Minnesota. The 14 kton Far Detector instrumented with 344,064 channels massively dwarfs the Near Detector at 300 tons and 20,192 channels. [1]

2. Common Analysis Format
The fine granularity of the NOνA detectors provides a detailed representation of particle trajectories. NOνA reconstruction algorithms are modules (plugins) in the art framework.
The data volume associated with such granularity, however, poses problems for analyzing data with ease and speed; the art files are large and their structure is intricate.

In order to provide a convenient means for physics analysis, NOνA created its Common Analysis Format (CAF) to summarize the results of reconstruction. This format discards hit-by-hit information in favor of higher-level reconstructed variables. Information is stored in a basic ROOT tree. [3] In order to resolve interactions from NOνA's min-bias readout, entries in the tree are reconstructed “slices”. Slicing is essentially an offline re-triggering based on correlated activity. The tree is highly segmented and hierarchical; tracks, showers, etc. are stored in separate branches. Event records in the tree (corresponding to slices) are defined by the StandardRecord class:

```cpp
class StandardRecord {
    SRHeader    hdr; // Header branch: run, subrun, etc.
    SRSpill     spill; // Beam spill branch: pot, beam current, etc.
    SRSlice     slc;  // Slice branch: nhit, extents, time, etc.
    SRTrackBranch  trk; // Track branch: nhit, len, etc.
    SRShowerBranch shw; // Shower branch: nhit, len, etc.
    ...
};
```

Figure 1. Schematic of the NOνA particle detector.
Some branches contain similar output from multiple algorithms – for instance alternative tracking or classification algorithms – which are stored in sub-branches. An example of a branch with only leaves and no sub-branches is `SRSlice`:

```cpp
class SRSlice {
  unsigned int nhit; ///< number of hits
  unsigned int firstplane; ///< first plane
  unsigned int lastplane; ///< last plane
  float calE; ///< sum of uncalibrated ADC
  ...
};
```

Objects for which multiple are present within a slice, like tracks, are stored using `std::vector`.

### 3. CAF Analysis Framework (CAFA2a)

#### 3.1. Motivation and Design

NOvA has sought to streamline the process of oscillation analysis by designing framework for filling histograms and creating physics results. The framework aims to be flexible enough to maintain compatibility across analysis groups. Common tasks (e.g. file handling, event loops, etc.) are encapsulated to avoid inconsistencies across users. Flexibility is achieved by allowing arbitrary manipulation of `StandardRecord` objects using function pointers and lambdas. The framework tracks exposure as events are handled so that histograms can always be scaled appropriately.

#### 3.2. Basic Example

Within the CAFA2a framework, the event loop is obfuscated from the user; instead, the user creates `Cut` and `Var` objects. Each of these objects wraps around a function pointer (or lambda) which takes a `StandardRecord` object as an argument and returns a result which is determined by the user. In the case of `Cut`, the return value should be a `bool`, while `Var` expects a float. Examples of `Cut` and `Var` objects can be seen in figure 3. Boolean operators are overloaded for `Cut` and arithmetic operators are overloaded for `Var`.

Users define `Binning` objects for either fixed or variable width binning for their histograms.

```cpp
const Binning bins = Binning::Simple(100, 0, 1000);```

![Figure 2. Reconstructed “slice” in NOvA. The x-z and y-z views are displayed separately. Two tracks are shown, one red and one blue.](image)
const Cut kCrudeMuonSel("trk.nkalman", "trk.kalman.len"),
    [](const caf::StandardRecord* sr)
    {
        if(sr->trk.nkalman == 0) return false;
        return sr->trk.kalman[0].len > 200;
    };

const Var kTrackLen("trk.kalman.len", "trk.nkalman"),
    [](const caf::StandardRecord* sr)
    {
        if(sr->trk.nkalman == 0) return 0.0;
        return sr->trk.kalman[0].len;
    };

Figure 3. Cut and Var serve as the primary user interface to the event loop in CAFAna.

The SpectrumLoader class handles files and looping. Input files can be specified singly or as
glob-style wildcards.

SpectrumLoader loader("reconstructed_events.root");
Spectrum builds a histogram out of Binning, SpectrumLoader, Var and Cut.

Spectrum len("Track length (cm)", bins, loader,
    kTrackLen, kCrudeMuonSel);

All Spectrum objects track exposure so that histograms can be rescaled. Users can define as
many spectrum objects as they like. Calling SpectrumLoader::Go() initiates the loop over files
and fills all associated spectra.

3.3. Extensions

3.3.1. Systematic shifts and reweighting  Both systematic shifts and event weights are optional
parameters for Spectrum. Weights are implemented similar to Var, a function which returns
the weight. Systematics can alter the events prior to Var and Cut evaluation, i.e. downstream
analysis.

3.3.2. Oscillation calculators  An oscillation calculator computes neutrino flavor transition
probability as a function of true energy. Oscillation parameters are configurable.

    osc::OscCalculatorPMNSOpt calc;
    calc.SetL(810); // Set baseline
    calc.SetDmsq21(7.6e-5); // Set Delta_M_{21}^2
    calc.SetDmsq32(2.35e-3); // Set Delta_M_{32}^2
    calc.SetTh12(asin(sqrt(.87))/2); // Set theta_{12}
    calc.SetdCP(0); // Set delta_cp

The OscillatableSpectrum class bins true energy distribution as an additional histogram
dimension. With that information, the histograms can be reweighted later according to any
oscillation calculator.
OscillatableSpectrum len("Track length (cm)", bins, loader,
kTrackLen, kCrudeMuonSel);
loader.Go();
Spectrum lenOsc = len.Oscillated(&calc, 14, 14); // numu -> numu

3.3.3. Fitting and sensitivity contours  Users can define variables for fitting through derived version of FitVar base class. The Surface class varies fit parameters to produce likelihood surfaces. Collaborators have built many variations around OscCalculator to serve various contour generation purposes. An example fitted contour can be seen in figure 5.

![Figure 4. Example OscillatableSpectrum plotted with two alternative sets of oscillation parameters](image1)

![Figure 5. Fitted contours for each of the predictions at left using the Surface class](image2)

4. Conclusions

NOνA has designed its Common Analysis Format to streamline physics analysis. The format discards hit-by-hit detector readout in favor of reconstructed objects like tracks and showers. An analysis framework has been built to simplify event selection, plotting and likelihood fitting. Flexibility is achieved through use of function pointers and derivation of base classes. Such a general framework allows users to spend less time writing and debugging redundant code.

Acknowledgements

The authors thank the NOνA collaboration for use of its Monte Carlo simulation, reconstruction and related tools. Support for this research was carried out by the Fermilab scientific and technical staff. Fermilab is Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy

References

[1] D. S. Ayres et al. (NOνA), FERMILAB-DESIGN-2007-01 (2007); R. B. Patterson, for NOνA, Nucl. Phys. Proc. Suppl. 235-236, 151 (2013).
[2] K. Anderson et al., FERMILAB-DESIGN-1998-01 (1998).
[3] R. Brun et al. (ROOT), Nucl. Inst. & Meth. in Phys. Res. A 81-86, 389 (1997).