MaCh3 Binned Fitter

June 17, 2024

Need for a fast/approximate fitter

MaCh3 fits currently take a long time - for ~100 million steps, can take 10,000+ CPU hours.

Can run on clusters, use many chains in parallel, use GPUs.

But a fit still typically requires many days.

In some cases, one may want to run a number of fits with different detector parameters before running a “final” fit.

It would be useful to have a fast fitter for this.

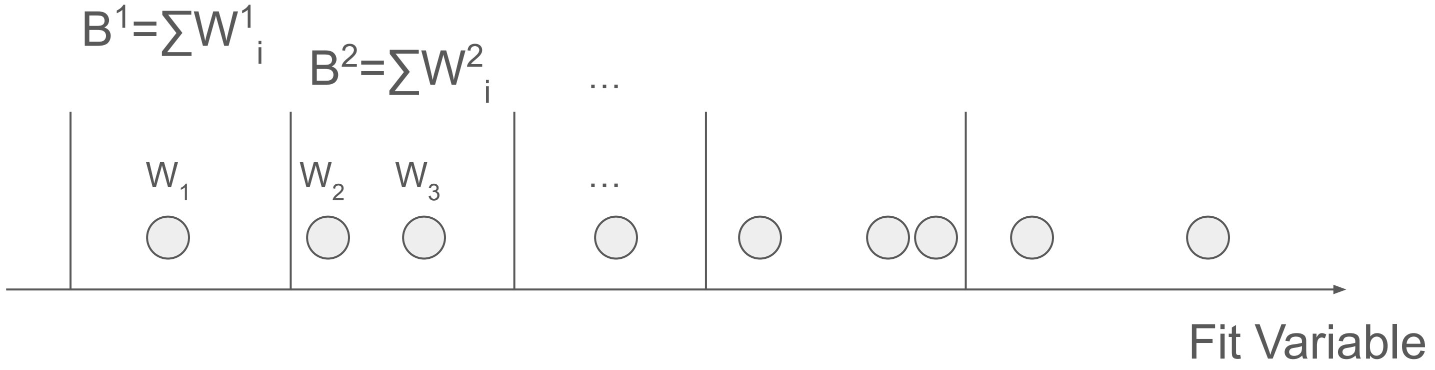
Guang and I would like to develop such a fitter for our the Theia LBL study.

Event-by-event

Currently: Loop over every event, then bin.

Binning then assigns bin weights as sum of event weights in bin.

Advantage: Each event can have a different weight.



This takes a long time due, in part, to looping over every event.

From Daniel Barrow

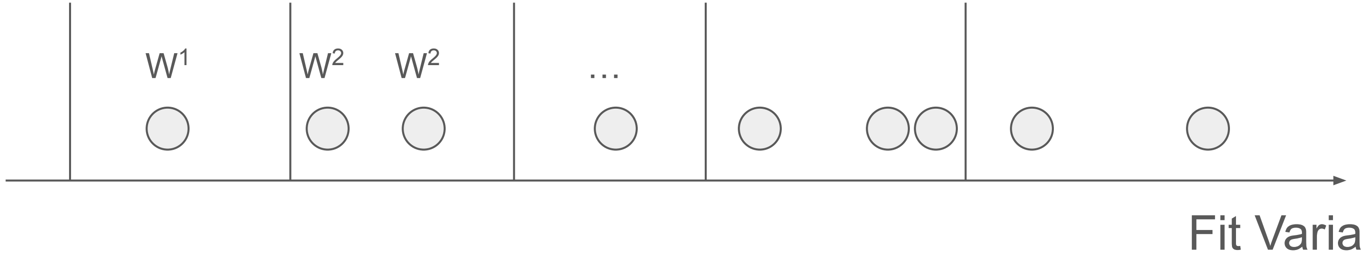
Binned Distribution

Assumes all events in a bin have the same weight.

Now bin weights are just a constant times number of events in bin.

$B^1 = \sum W^1 = N^1 \times W^1$

$B^2 = \sum W^2 = N^2 \times W^2$ …



From Daniel Barrow

Binned Distribution

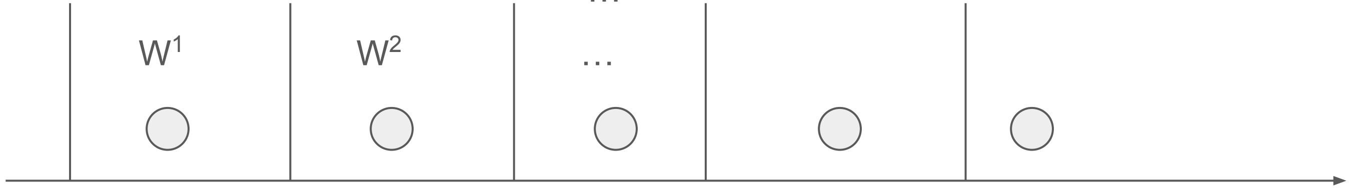
Assumes all events in a bin have the same weight.

Therefore equivalent to using one event per bin.

$B^1 = \sum W^1 = N^1 \times W^1$

$B^2 = \sum W^2 = N^2 \times W^2$

…



Fit Variable

Now loop over nBins rather than nEvents, fundamentally using the same code machinery

Problem: Justifying the binning choices

From Daniel Barrow

Implementation

- Currently:
    - Read MC files and store events in fdmc_base struct.
    - In each step, there is a loop over events in fdmc_base struct.

- New flow for binned fit:
    - Read MC file and store events in fdmc_base struct.
    - At start time, call new function binData()
    - binData() loops over events and fills a binned array.
    - binData() creates a single event for each bin with $w = \sum w_i$
    - binData() writes these new events to disk as binned_MC file
    - Then fitter reads binned_MC file and stores events in fdmc_base struct.
    - Now each step is looping over just one event per bin.

Implementation

MaCh3 consists of MaCh3_core plus experiment-specific build (MaCh3_dune).

Experiment-specific code includes a configuration file that defines which (up to two) variables to fit in, and specifies binning for each of these.

MaCh3_core contains a struct with generic x_var and y_var and reads config file to determine which variables these are and what the bins are.

We use an analogous system for the binData() step.

Experiment-specific config file defines which variables to bin in and specifies the bins (in principle *different* from the bins for the fit).

Add generic binVar_1, binVar_2, … to MaCh3_core, which will read config file to determine which variables these are.

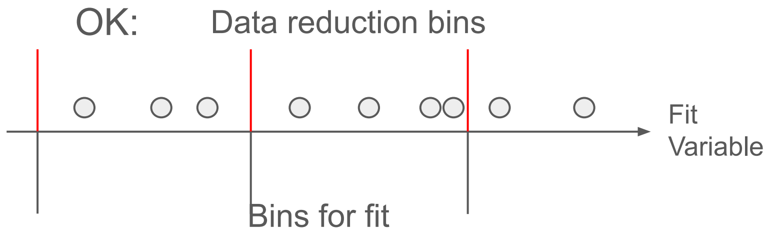
Requirements on configuration

Some obvious requirements to be enforced:

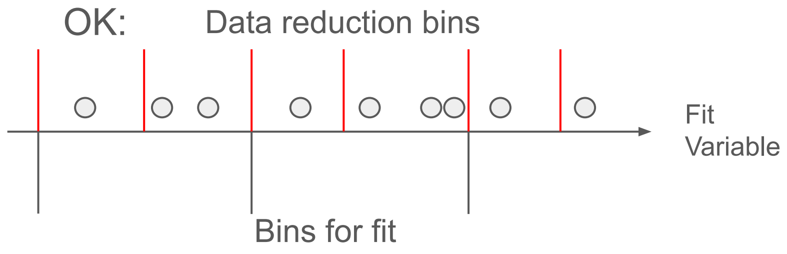
- Bin variables must include fit variables and E_true
  - Must be binned in fit variables for fit to work
  - Must be binned in E_true for energy scale systematics (see next slide)

- For the fit variables, the bins used for binData() should divide the bins used for fitting evenly.

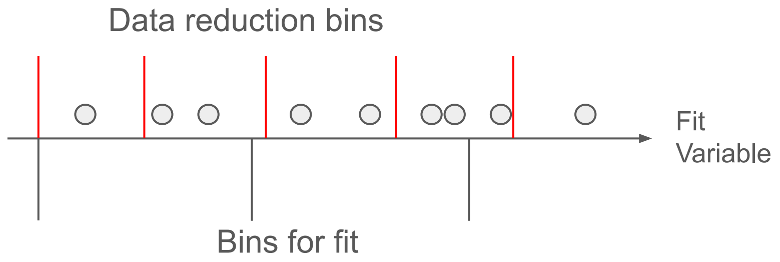
OK:



OK:



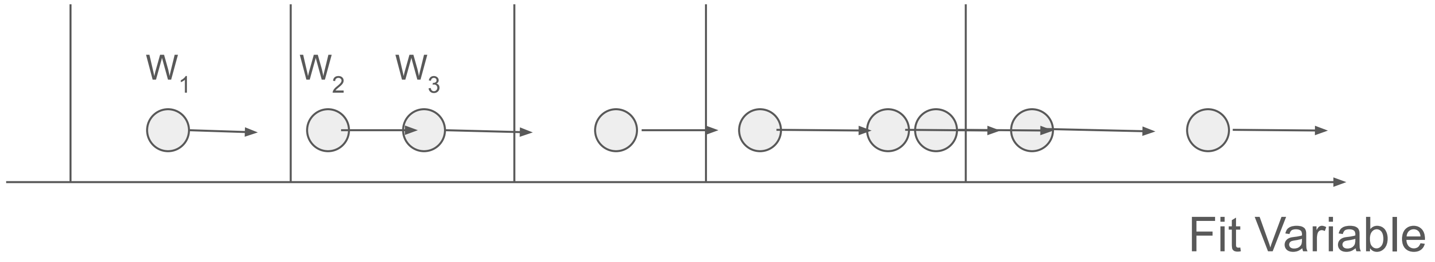
Not OK:



Energy scaling

Energy shift is applied event by event, meaning some events will change bins.

With one event per bin, this doesn't work.



Solution: In binData() step, generate splines to simulate energy shifting.

Splines already used in MaCh3 for other systematics, just need to implement for energy scale.

Choice of variables/binning

Choice of variables and binning requires thought.

- High number of binned variables results in high-dimensional splines.
- Proposal: Start with just three variables (E_true and the two fit variables), as well as interaction channel (doesn't add to dimensionality because discrete).

- May run into problems if the number of events in a bin is small or zero.
- Proposal: Start with same binning as for fit in fit variables, throw a warning if number of events in a bin is low, and adjust E_true binning to avoid this.

If we are happy with the results given this binning, we're done. If not, then we think about adding more variables or reducing bin size.

Validation scheme

- Choose some target chi^2 (corresponding to a chosen confidence interval, e.g. 95%).
- Run fits on a small number of events, using both fitters, and varying the bin selections.
- From the bin widths tested, choose a set of bin selections for which the average of the relative difference in chi squared is less than the target resolution in all parameters.
- Using the chosen bin widths, run a fit with a large number of events, again using both fitters.
- Compare chi^2 distributions in all relevant parameters again. If these still show the desired accuracy, and the speed is acceptable, these plots validate the fast fitter. If these do not show the desired accuracy, or if the fit is too slow, go back to step two, choose new binning, and try again.

Status

Done:

- Compile, learn to run MaCh3
- Write function to bin data

In progress:

- Enforce requirements on binning choices
- Write binned data to disk (instead of just saving in the struct)
- Implement splines for energy scale

Not yet done:

- Test binData() function
- Choose binning and validate

Ideas not yet given serious thought:

- Machine learning for choice of step (Guang’s idea)
- Alternatives to Markov chain (Mike’s idea)