CDF/DOC/MONTECARLO/CDFR/7981
Version 1.0
January 12, 2006

Re-weighting the Longitudinal Luminosity Profile in Gen5 MC Samples

Jason Galyardt

Abstract

The β^* model of the longitudinal luminosity distribution offers improved data-MC matching over the gaussian model when used as the generating PDF in Monte Carlo. In particular, Gen5 MC samples which have been generated using the gaussian model can be re-weighted in order to take advantage of this improved matching. This note describes an ntuple-level tool, the `RwZVtx` class, which provides this functionality on a run-by-run basis.

1 Introduction

The β^* model of the longitudinal luminosity distribution ($\frac{dL}{dz}$) has been implemented in Gen6 simulation code [1], and it has been shown, on the whole, to match the data better than the simple gaussian model employed in the beamline fits. These gaussian fits to the longitudinal luminosity distribution under-represent the data in the peak and in the tails, where the discrepancy can be as much as a factor of two.

The Gen5 simulation code (class `GenPrimVertModule`) uses the gaussian parameters found in the beamline database to generate the primary vertex z coordinate. The quality of the matching between MC and data can be improved (at least as far as the primary vertex z distribution is concerned) by re-weighting a Gen5 MC sample such that the effective primary vertex z distribution for a particular run matches the β^* parameterization used in Gen6 simulation. The analytical re-weighting function described in this note is as follows,

$$f(z) = N_1 \frac{e^{\frac{-(z-z_0)^2}{2\sigma_z^2}}}{1 + \frac{(z-z_{min})^2}{\beta^{*2}}} \frac{1}{N_2 e^{\frac{-(z-\mu)^2}{2\sigma^2}}}, \tag{1}$$

where the β^* model parameters, z_0, σ_z, z_{min}, β^*, are obtained from fits to min-bias COT primary vertex z distributions, and the gaussian parameters, μ and σ, are obtained from fits to similar distributions (either SVX or COT beamline histograms). The parameters N_1 and N_2 are calculated such that the two pieces of this function, the β^* function and the gaussian, respectively, are normalized on [-60.0, 60.0] cm (this is the default; the normalization domain can be changed by the user, see section 4.2). This re-weighting function is implemented on a run-by-run basis; it is therefore the user's responsibility to handle the situation where the MC sample run range does not match the data run range.

2 Gaussian Parameters

The gaussian beamline parameters are obtained from the database via the `BeamDatabase` executable [2], which is part of the `Alignment` package. The run range requested was [141597, 203799]; no good run list was applied, and no runs were excluded.

Since the goal of this re-weighting procedure is to "undo" the gaussian parameterization, then apply a β^* parameterization, the "correct" beamline must be chosen for each run: SVX or COT. The algorithm used in the simulation code to choose between the two (potentially) available beamlines is as follows. For each run, the SVX beamline is tried first, and if the database access is successful (no matter whether the parameters are valid), that beamline is used. If there's no SVX beamline available, the COT beamline is tried. This same logic is applied to the gaussian parameters used by the `RwZVtx` class.

However, there is a large list of runs for which no fitted SVX beamline is available but which still have an SVX beamline entry in the database. The longitudinal gaussian parameters are set to zero for these runs, though the transverse distributions have "dummy" entries. All events for these runs are generated at $z = 0$ in Gen5 simulation, so no re-weighting can be applied. This behavior has since been fixed in Gen6 simulation, where the SVX beamline is required to be fitted.

3 β^* Model Parameters

The $\beta^* \frac{dL}{dz}$ parameters were taken from revision 1.4 of `generatorMods/src/GendLdz.cc`, which are themselves fits to min-bias data (see [1] for a detailed description of the methodology). The parameter sets available span the run range [141597, 203799], though no fits were made for the COT compromised period.

4 Implementation

The implementation of this re-weighting function is in the self-contained class, `RwZVtx`. This class is suitable for use at the ntuple analysis level. Currently, this class is not in the code repository, but it can be found at the URL,

`http://www-cdf.fnal.gov/~galyardt/internal/zVtx/RwZVtx_norm60.tgz`.

An attempt was made to make the class user friendly so that one need only feed an instance of the class the current run number and ask for an event weight, given a generator-level primary vertex z coordinate. This also provides for uniform use across analyses.

The parameters for the re-weighting function are written to a ROOT `TTree` by the class `BeamPosAna`, and this ROOT file is then read by the `RwZVtx` class. The latter is the only class meant to be employed by users. The separation of the code into a class which gets the parameters for each run (`BeamPosAna`) and a user interface class was done for logistical and performance reasons.

Figures 1 and 2 illustrate the re-weighting functions given by the `RwZVtx` class for four specific runs, 152954, 165063, 178758, and 199143. The β^* and gaussian functions which went into Figure 1 were normalized on [-100.0, 100.0] cm, while those contributing to the re-weighting functions shown in Figure 2 were normalized on [-60.0, 60.0] cm. The normalization domain can be changed by the user as shown in section 4.2.

Note that the scale of the re-weighting function for run 199143 is much larger than for the earlier runs in both figures. This is partially due to significantly different beam conditions in the newer data-taking period, but it is also caused by differing beam conditions with respect to the other runs in the same run range, [198882, 199620). The re-weighting functions for run 199143 together with four other runs in the same run range (199067, 199076, 199146, and 199148) can be seen in Figure 3. From this figure it is clear that run 199143 cannot be taken as a representative run for its run range. The re-weighting functions of the other four runs, however, are quite consistent with one another.

Figures 4 through 11 display the β^* and gaussian fits overlaid upon `MINBIAS_1HZ` data for the same runs as in Figures 2 and 3. The β^* and gaussian function parameters are obtained from the `RwZVtx` class. The two functions are normalized to the integral of the data histogram for $|z| < 60.0$ cm in each figure.

4.1 Usage

The instance of the `RwZVtx` class must be available throughout an analysis module's lifetime, so it should be created as a private data member of the analysis class,

```
RwZVtx *_myRwZVtx;
```

Memory for the object should be allocated during the `MyAnaModule::beginJob()` function of the analysis class,

```
_myRwZVtx = new RwZVtx;
```

During the `MyAnaModule::beginRun()` function of the analysis class, the `RwZVtx` object should be given the current run number,

```
Bool_t success = _myRwZVtx->loadRun(currentRun);
```

With the generator-level primary vertex z coordinate as input, the event weight can then be obtained as follows,

```
Float_t obsvZ;  // the generator-level primary vertex z coordinate
Float_t wgt = _myRwZVtx->getEvtWgt(obsvZ);
//RwZVtx::getEvtWgt() returns 1.0 if run is not valid
```

During `MyAnaModule::endJob()` or `MyAnaModule:: MyAnaModule()`, delete the `RwZVtx` to avoid memory leaks,

```
delete _myRwZVtx;
```

4.2 Utility Functions

The following utility functions are provided in the interest of flexibility.

```
// Test that the current run's parameters were loaded properly.
Bool_t validity = _myRwZVtx->isValid();

// Get the currently valid run number
Int_t currentRun = _myRwZVtx->getCurrentRun();

// Get the re-weighting function parameters for the current run.
// First create the array; note the size!
Double_t parms[8];
// Now get the parameters.
Bool_t psuccess = _myRwZVtx->getRwParms(Double_t *parms);

// Get a histogram filled from the weighting function.
// First initialize the histogram (x-axis range and binning are not
// restricted).
TH1F *hzrw = new TH1F(``hzrw'',''hzrw'', 100, -100.0, 100.0);
Bool_t errFlag = kFALSE; // Do not calculate bin errors.
// Now Fill the histogram; don't forget to 'delete' it when done.
Bool_t hsuccess = _myRwZVtx->getRwHist(TH1F *hzrw, const Bool_t errFlag);

// Change the normalization domain of the betaStar and gaussian
// functions; note that RwZVtx::loadRun() must be called
// after the normalization domain is changed.  The default
// domain is [-60.0, 60.0] cm.
t->setNormDomain(loZ, hiZ);
// Get the current normalization domain
t->getNormDomain(loZ, hiZ);
```

5 Summary

A tool for re-weighting Gen5 MC samples in order to transform the longitudinal luminosity profile from the generated gaussian into a β^* function has been developed. It is implemented in the class `RwZVtx` on a run-by-run basis in a user-friendly way at the ntuple level. Validation of this tool is on-going.

This tool should enable Gen5 analyses to obtain better agreement between data and MC with respect to the longitudinal luminosity distribution.

References

[1] J. Galyardt, "Simulation of the Longitudinal Luminosity Profile Using the β^* Model", CDF Note-7521.

[2] The default algorithm for choosing the best SVX and COT beamlines available was used.

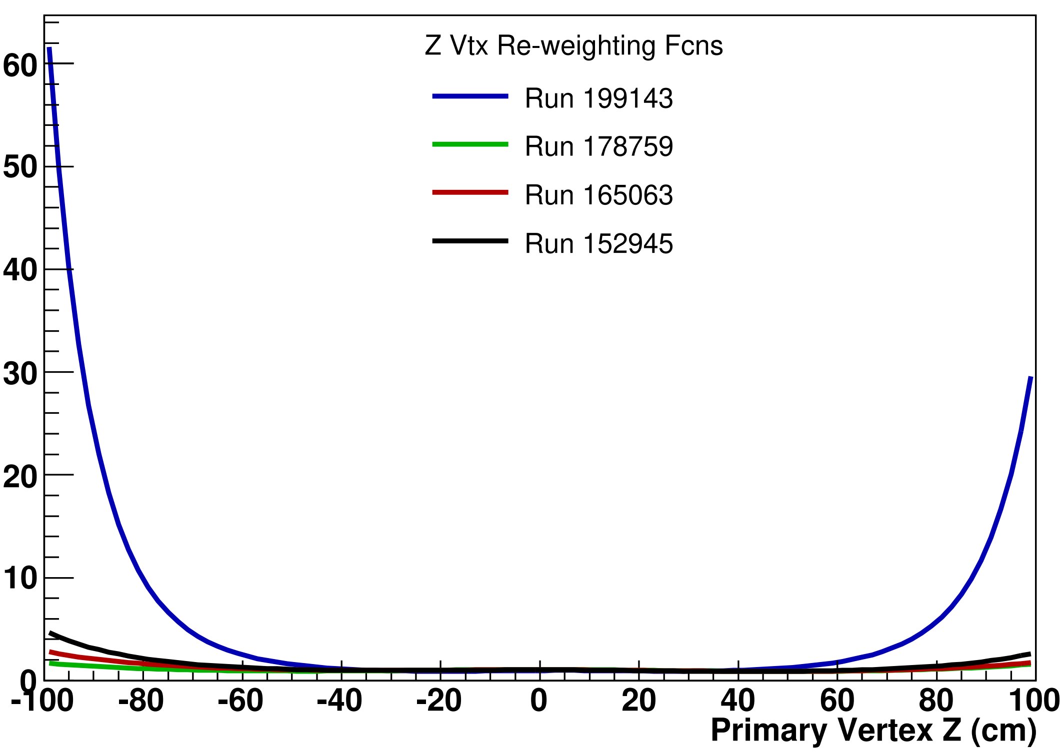

Figure 1: $\frac{dL}{dz}$ re-weighting functions for four runs; the normalization of β^* and gaussian functions is on [-100.0, 100.0] cm.

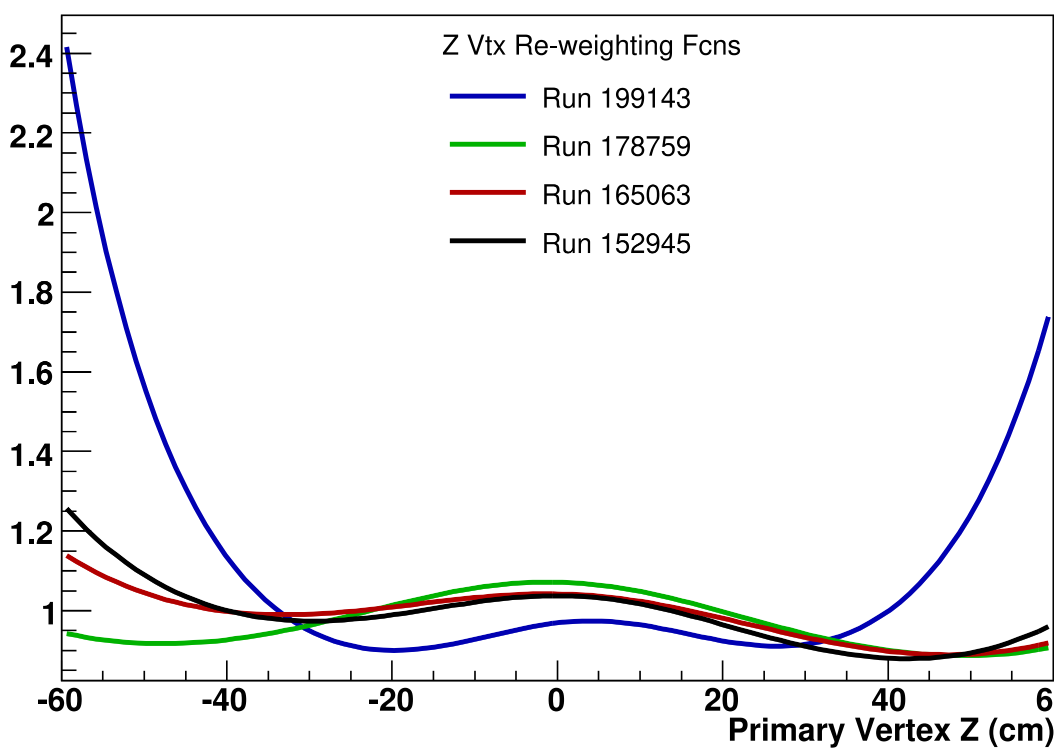

Figure 2: $\frac{dL}{dz}$ re-weighting functions for four runs of different run ranges; the normalization of the β^* and gaussian functions is on [-60.0, 60.0] cm.

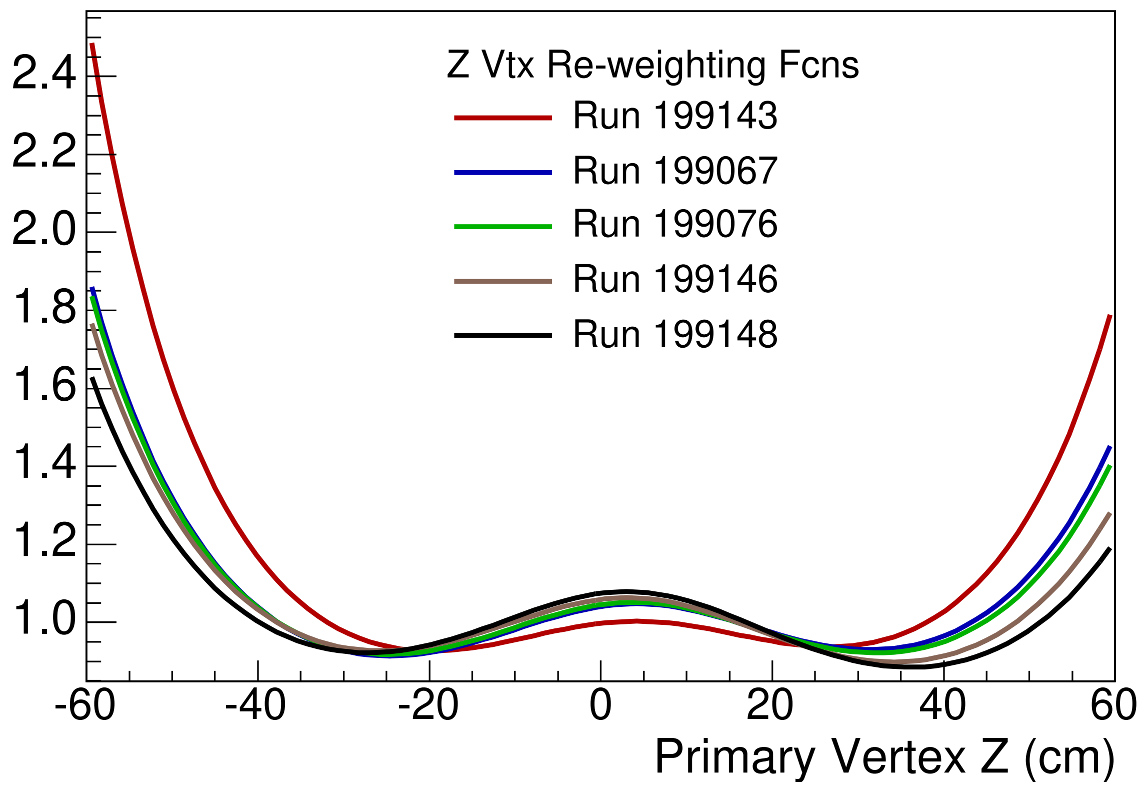

Figure 3: $\frac{dL}{dz}$ re-weighting functions for four runs in the same run range; the normalization is on [-60.0, 60.0] cm. The β^* function parameters are the same for all four runs, but the gaussian parameters are all different.

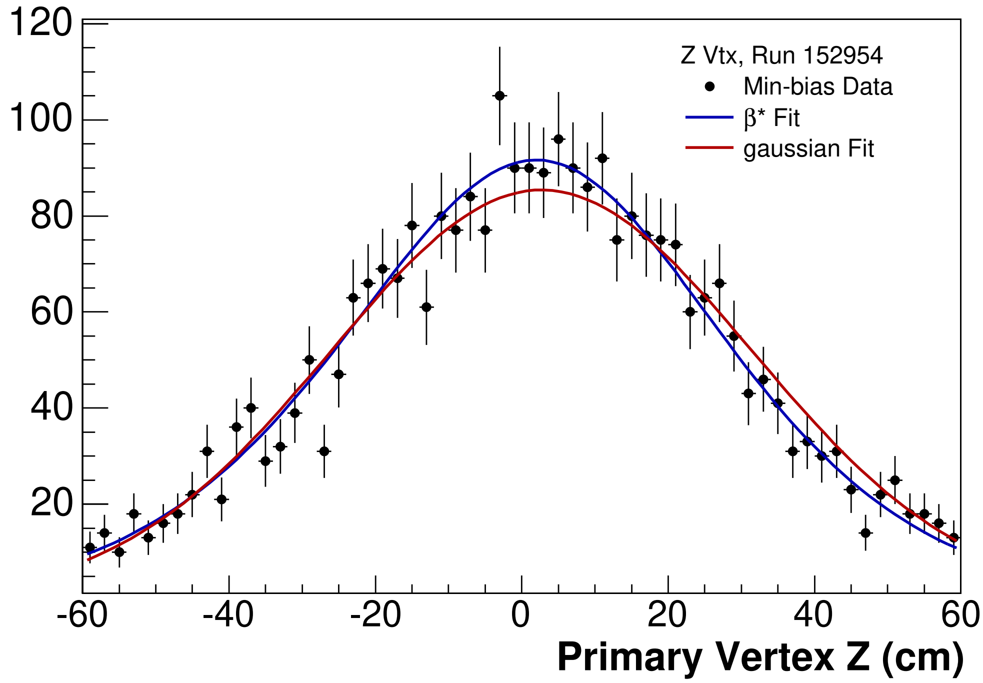

Figure 4: $\frac{dL}{dz}$ fits for run 152954; the β^* function parameters are taken from the run range [152674, 153374).

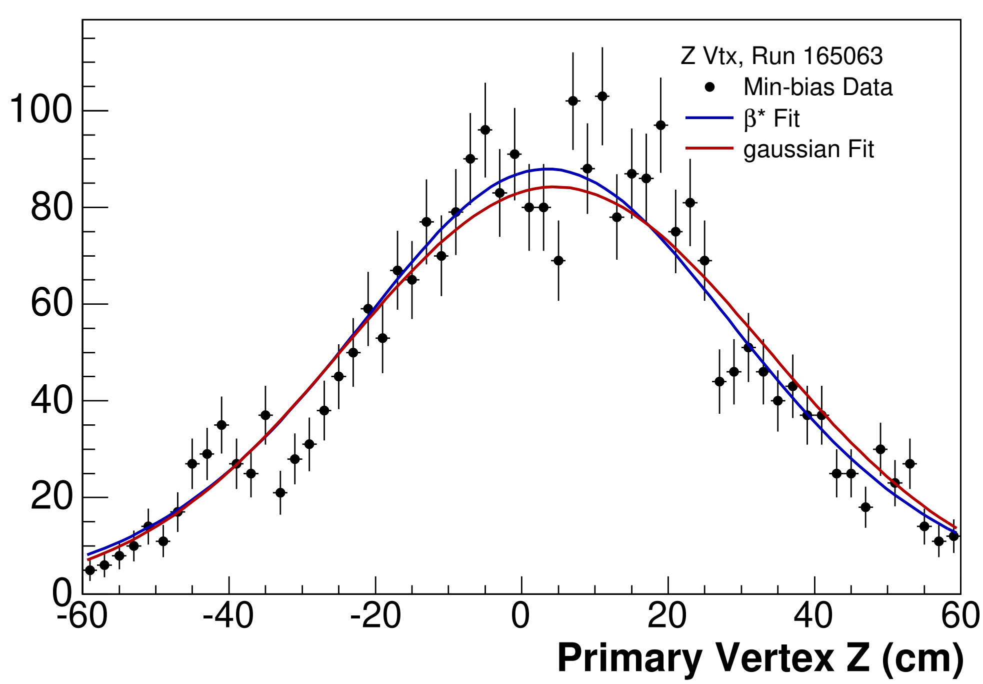

Figure 5: $\frac{dL}{dz}$ fits for run 165063; the β^* function parameters are taken from the run range [164302, 165087).

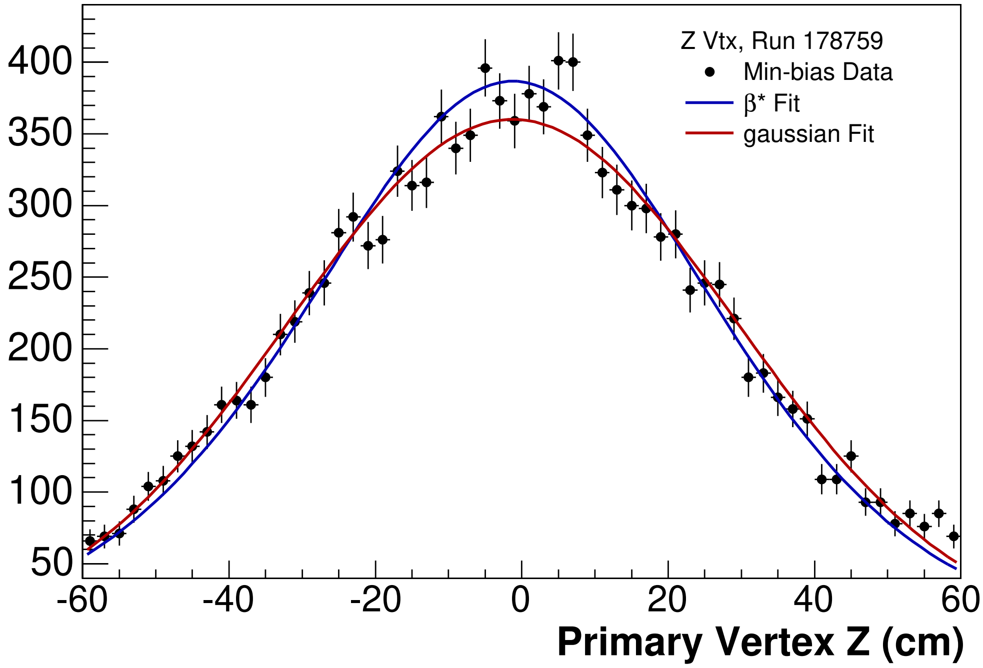

Figure 6: $\frac{dL}{dz}$ fits for run 178759; the β^* function parameters are taken from the run range [178438, 178882).

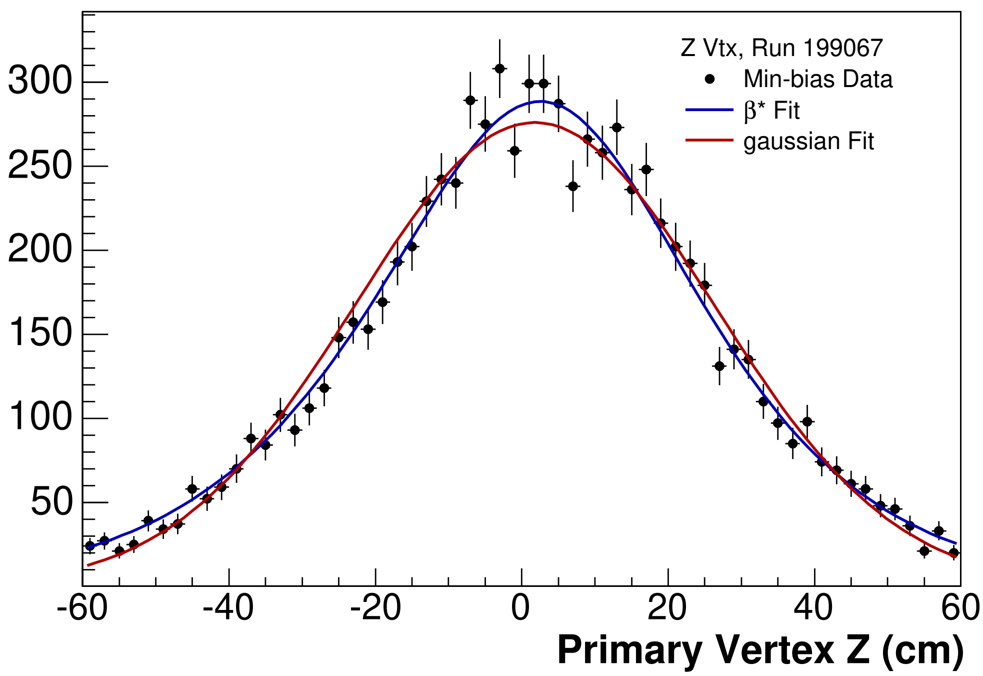

Figure 7: $\frac{dL}{dz}$ fits for run 199067; the β^* function parameters are taken from the run range [198882, 199620).

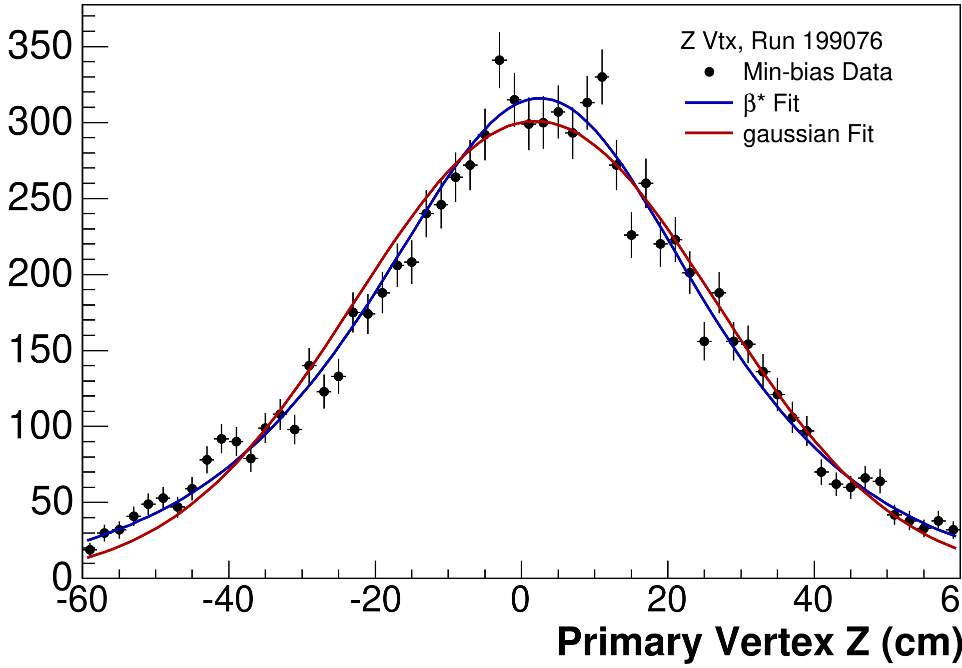

Figure 8: $\frac{dL}{dz}$ fits for run 199076; the β^* function parameters are taken from the run range [198882, 199620).

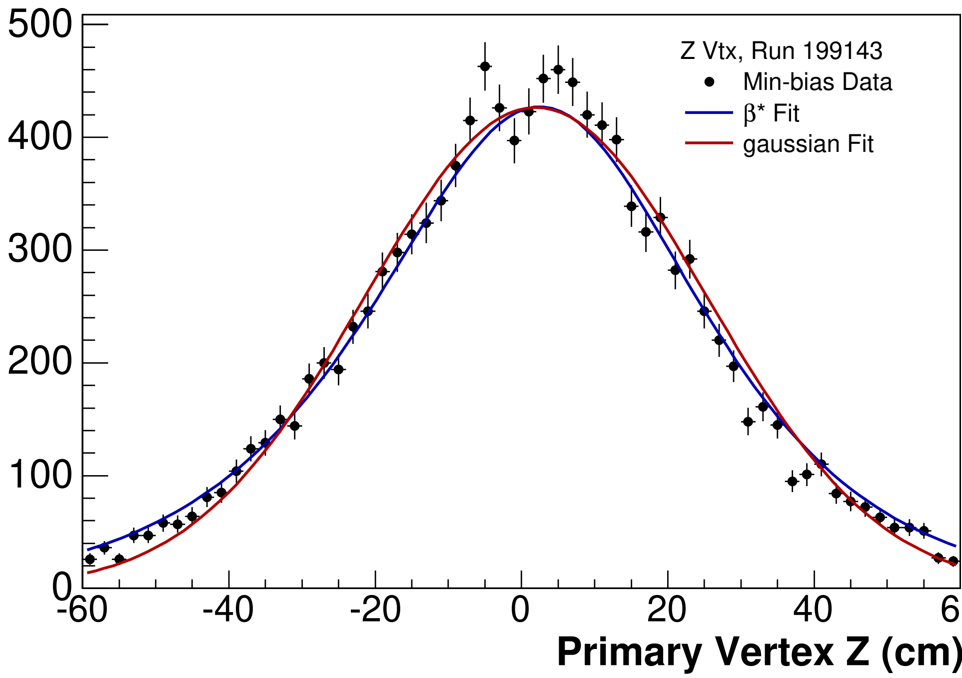

Figure 9: $\frac{dL}{dz}$ fits for run 199143; the β^* function parameters are taken from the run range [198882, 199620).

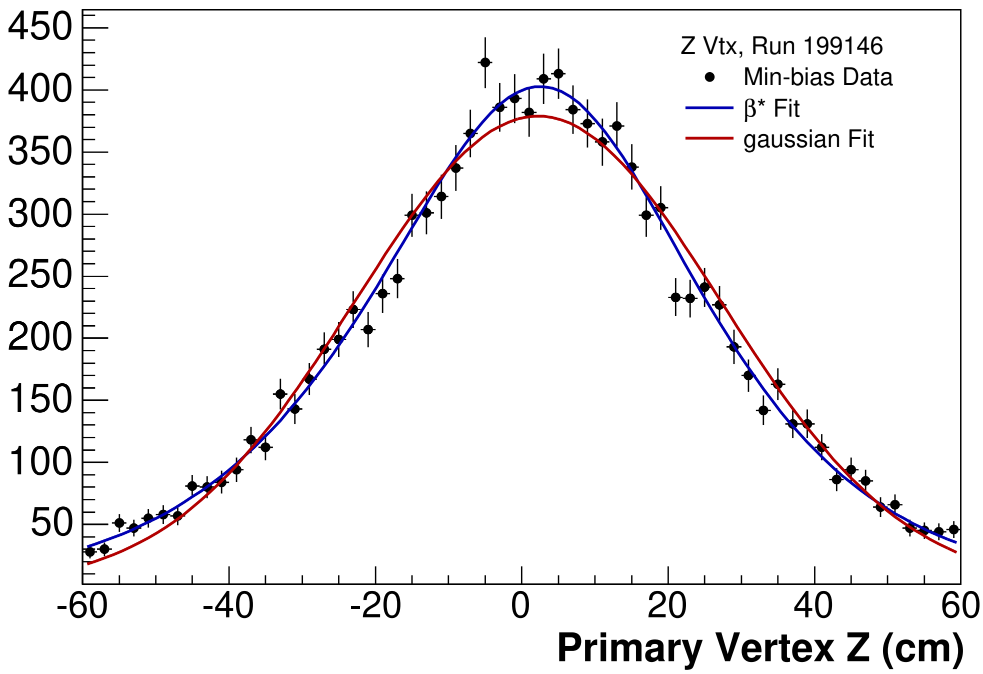

Figure 10: $\frac{dL}{dz}$ fits for run 199146; the β^* function parameters are taken from the run range [198882, 199620).

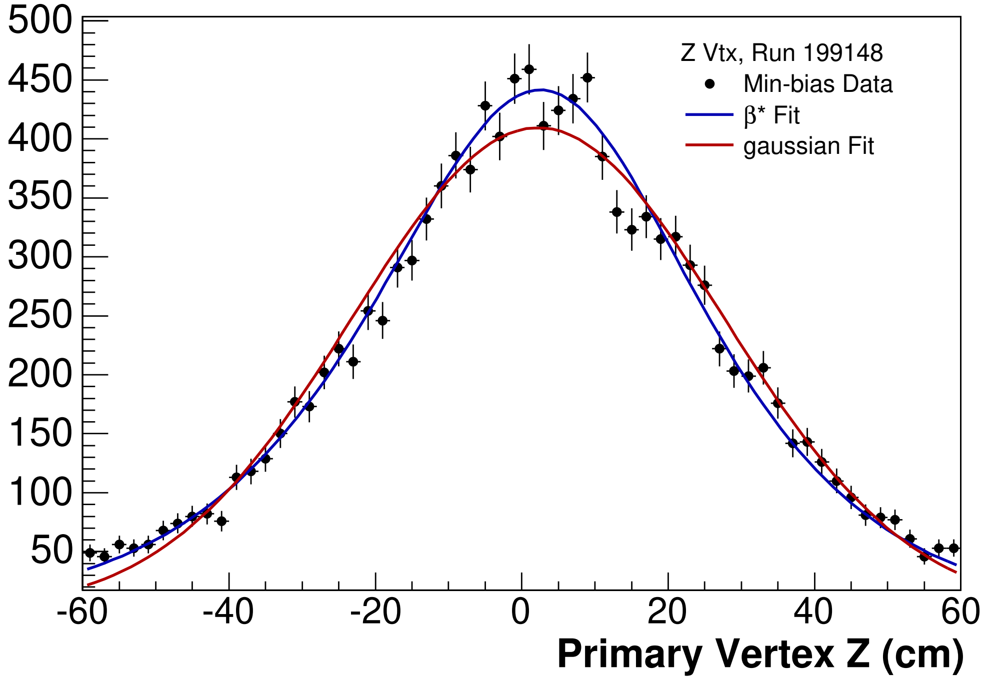

Figure 11: $\frac{dL}{dz}$ fits for run 199148; the β^* function parameters are taken from the run range [198882, 199620).