Guidance on Centrality Definition

Zachary Sweger

Description

This document is meant to be a reference for anyone attempting to define centrality within STAR. The codes mentioned here should be thoroughly inspected for your use-case, and are not intended to be run exactly as provided. Instead, the codes should be viewed as a starting point which may need to be edited before being run. People who are doing centrality calibrations should report updates regularly in the centrality focus group meeting, and the final calibration parameters need to be approved by the group.

Some resources are listed here:

- The STAR Centrality Group page is located at

  `https://drupal.star.bnl.gov/STAR/blog/gnigmat/centrality-definition-star`
- A detailed description of the 2019 19.6 GeV centrality can be used as a guide to other centrality tasks. This is located at

  `https://drupal.star.bnl.gov/STAR/system/files/19p6GeVCentrality_v1.pdf`
- As of the writing of this document the centrality codes are located at

  `https://github.com/star-bnl/star-sw/tree/main/StRoot/PWGTools/CentralityCalibration`

Creating RefMultCorr

The first step to defining centrality is to first define a corrected refMult variable called RefMultCorr. The purpose of this is to reject pileup and account for the effects of the variations in luminosity and V_z on event multiplicity. Before any of these steps can be finalized, you need a final bad-runs list from the QA group. You are encouraged to investigate the centrality procedure prior to final bad-runs being defined, but all steps should be repeated from the beginning once the bad-runs list has been finalized.

The general procedure is performed as follows

- **REJECT PILEUP EVENTS**: The first step is to reject pileup events. For this you may use the pileup code developed for the isobar blind analysis. Instructions on how to obtain and run this code start on page 34 of the following document:

  `https://drupal.star.bnl.gov/STAR/system/files/isobar_analysis_note_v3.pdf`

  You will first need to generate a 2D histogram of refMult vs nBTOFMatched tracks. For this it may help to use the code located at

  `https://github.com/star-bnl/star-sw/tree/main/StRoot/PWGTools/CentralityCalibration/Centrality`

  The `processPicoDst.xml` file submits jobs for the `runCentralityAnalyzer.C` macro. Using this macro you can turn on and off the pileup rejection, Vz corrections, and luminosity corrections. Turn off

Email address: `zwsweger@ucdavis.edu` (Zachary Sweger)

all of these, set other settings to fit your dataset, and generate the 2D histogram for the pileup rejection. For the 2019 19.6 GeV dataset, due to the wide Vz range used, we found that there was a significant Vz-dependence to the pileup cuts. So we defined pileup cuts in 5 Vz bins. This document is not meant to instruct exactly what is right for your dataset, but it may be necessary to do something like that. See slides 4-7 of the 19.6 GeV reference to see what the cuts and parameters look like. Once you have the parameters from the pileup removal code, you may use them in `runCentralityAnalyzer.C` to re-process the picoDsts and evaluate that the cuts worked. Once they work, you may move on to the next step.

- **CORRECT FOR LUMINOSITY DEPENDENCE**: For collider datasets, we use ZDC coincidence rate as a proxy for luminosity. For fixed-target datasets we use the east BBC rate and east ZDC rate. When plotting the average refMult versus luminosity, we expect the average refMult to drop slightly with increasing luminosity. This is because tracking efficiency drops with increasing luminosity due to increasing TPC occupancy. For 19.6 GeV dataset we initially noticed a positive slope when plotting average refMult vs. ZdcCoincidenceRate, but this disappeared when we applied a refMult>20 cut to reject low-multiplicity noise. You should decide whether a cut on refMult is needed, and if so, what the optimal cut value is. Use `runCentralityAnalyzer.C` to plot the average refMult vs luminosity and determine if there is a significant dependence. This is detailed in slides 8-15 of the 19.6 GeV reference. Once you've generated a TProfile with average refMult vs ZdcCoincidenceRate, just fit this with a line in ROOT. If there is a significant slope, then a stretching factor will be applied to refMult values at different luminosities to correct for this. If the slope is small then the Centrality Group may decide not to apply a luminosity correction. If you apply a luminosity correction, verify that it works by re-processing the picoDsts with the luminosity correction turned on in `runCentralityAnalyzer.C`. For fixed-target datasets, it is often not necessary to apply a luminosity correction due to stable run conditions over the short duration of the data-taking. Nevertheless the luminosity-dependence should be checked for each dataset to confirm this.

- **CORRECT FOR Vz-DEPENDENCE**: Now you need to apply two Vz corrections. For fixed-target datasets, this step is not necessary because all events take place at Vz=200cm. The first of the Vz corrections is described in slides 16-25 of the 19.6 GeV reference. This is a stretching correction which is meant to account for drops in tracking efficiency for events away from Vz=0. See slide 16 for how this works. When analyzing a narrow Vz range (like $-35 < Vz < 35$cm), you might find that a sixth-order polynomial gives a sufficient correction. However for the large Vz range analyzed at 19.6 GeV, you can see on slide 25 that a bin-by-bin (pointwise) correction was more appropriate. Reference code to get started with calculating the corrections is available at `Centrality/referenceVzCorr.C`.

  The second Vz correction is a reweight correction which is meant to account for changes in the trigger efficiency for events away from Vz=0. For example, of all the events with a large Vz near the edge of the TPC, low-multiplicity events will be under-counted. This is due to the difficulty of triggering on events with such few tracks near the edge of the TPC. So the peripheral region of the multiplicity distribution will be lower for large-Vz events, causing the total multiplicity distribution to have a different shape overall. To fix this, we give events with a given multiplicity a weighting factor which re-shapes the multiplicity distribution to match the shape at Vz=0. For each Vz bin, you can make a multiplicity distribution, and take the ratio to the multiplicity distribution for your Vz window that's centered around Vz=0cm. Calculate a re-weight factor for each multiplicity bin. An example of what your reweight correction may look like is provided here:

```
Double_t StCentralityAnalyzer::getShapeWeight(Double_t Vz, Double_t RefMult){

  // no shape correction for -9<=Vz<=9
  if(Vz>=-9 && Vz<=9) return 1.;
  //obtain index to load weight
  Double_t VtxZBinDouble = Vz/2. + 17.;
  Int_t VzIndex = 0;
  if(Vz == 25.) VzIndex = 29;
  else if(Vz == -35.) VzIndex = 0;
  else VzIndex = TMath::Nint(VtxZBinDouble);
  //handle VzIndex for Vz>9
  if(VzIndex >= 22) VzIndex = VzIndex - 9;
  //retrive shape weight
  Double_t weight = ShapeWeightArray[mShapeIndex][VzIndex][TMath::Nint(RefMult)];
  //handle bad weight
  if(weight == 0 || TMath::IsNaN(weight)) weight = 1.;
  return weight;

}
```

where the `ShapeWeightArray` holds the correction weights.

- FINISH: Once you've rejected pileup, applied the luminosity correction (or demonstrated it's not necessary), and performed the two Vz corrections, you should generate a refMult distribution with all of these applied. This is your refMultCorr. It is now time to move on to the Glauber procedure.

Glauber Procedure

For a review on what the Glauber model is and what it's use for, please see the following paper: `https://arxiv.org/pdf/nucl-ex/0701025.pdf`. The instructions for running STAR's Glauber model are in the README located at `https://github.com/star-bnl/star-sw/tree/main/StRoot/PWGTools/CentralityCalibration/Glauber/star_glauber/README.md`. These instructions are thorough so I will not repeat them here.

You should double check the centrality cuts output by the Glauber code. Look at slides 31 and 32 of the 19.6 GeV reference as to how you may integrate by hand (in a ROOT terminal) the data and simulated multiplicity distributions to verify that the cumulative integral at each centrality cut is as close to a multiple of 5% as possible. Remember that the centrality procedure returns cuts optimized to, for example, make the integral at the 10% cut close to 10% of the total integral. They are NOT optimized to make cuts such that the 5-10% bin contains 5% of the total integral. This is a minor distinction but will affect exactly where each cut is placed.

Also remember when integrating by hand that for the 0-20% cuts the data is integrated. For the $> 20\%$ cuts the simulated multiplicity distribution is integrated. Review slide 31 of the reference slides to see how this is done. If for any reason there is a mismatch between your by-hand cuts and the Glauber code cuts, you should work to discover the cause. Within the Glauber package, the function to inspect when looking for how centrality is calculated is located at `star_glauber/StRoot/StCentralityMaker/StNbdFitMaker.cxx` in the function `CalculateCentrality()`. You may triple check by using this short macro which, when provided the Glauber fit, will calculate cuts and estimate trigger efficiencies: `Centrality/GetCentrality.C`. Finally, see slide 32 of the 19.6 GeV reference for a brief discussion of the differences between multiplicity, bin number, and the placement of your cut. For example, if you use `data->Integral(301,500)/sim->Integral()` to see that this gives the cut closest to 5%, this really means that the cut belongs at a multiplicity of 300 because 301 is the bin number. Further, multiplicity bins should be centered on the half-integer values in order to make the cut exactly on the integer multiplicity values. This should be handled in the code already, but it is important to be cognizant of this in case any questions arise.