Table of Contents

Generate EDF Curves

Introduction

Introduction from Bob Cousins' email:

```
I have typically used tests based on the empirical distribution function
(EDF), which in in this case is the integrated number of accumulated
events as a function of accumulated lumi, divided by total number of
events. The "model" to be tested is that this is a straight line from 0
to 1, as integrated lumi goes from 0 to total. The most popular test in
HEP using the EDF is the Kolmogorov-Smirnov test, but others
(Anderson-Darling, etc.) are reputed to be better as omnibus tests
(http://www.jstor.org/stable/2286009 ).
```

The general idea is to see if the yield of any variable grows as a constant factor of integrated luminosity as one expects.

Installation

=generateEDF.py= is in `FWCore/PythonUtilities/scripts/generateEDF.py`). The script is self-contained (only depends on Python, Root, and PyRoot), so one could either check out the tag `V01-06-07` of `FWCore/PythonUtilities` or, alternatively, simply download the script:

```
wget "http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/FWCore/PythonUtilities/scripts/generateEDF
```

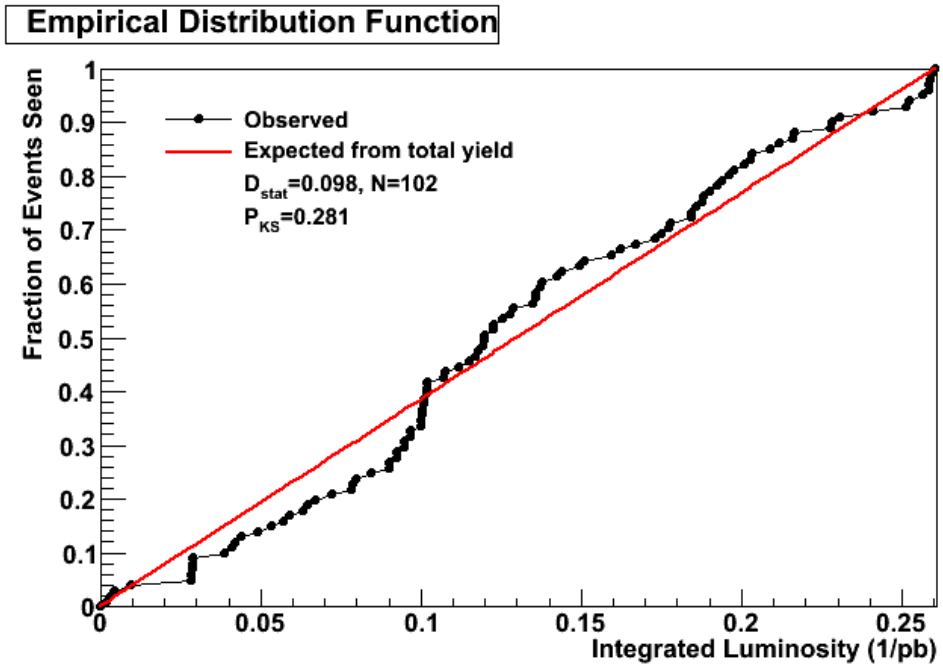
Simple Time-Ordered Example

Time-ordered simply means data taken earlier is listed first, later data is taken last (*i.e.* sorted by run number).

Without worrying about any options, one can run the script with three filenames. This script does come with an extensive `--help` menu.

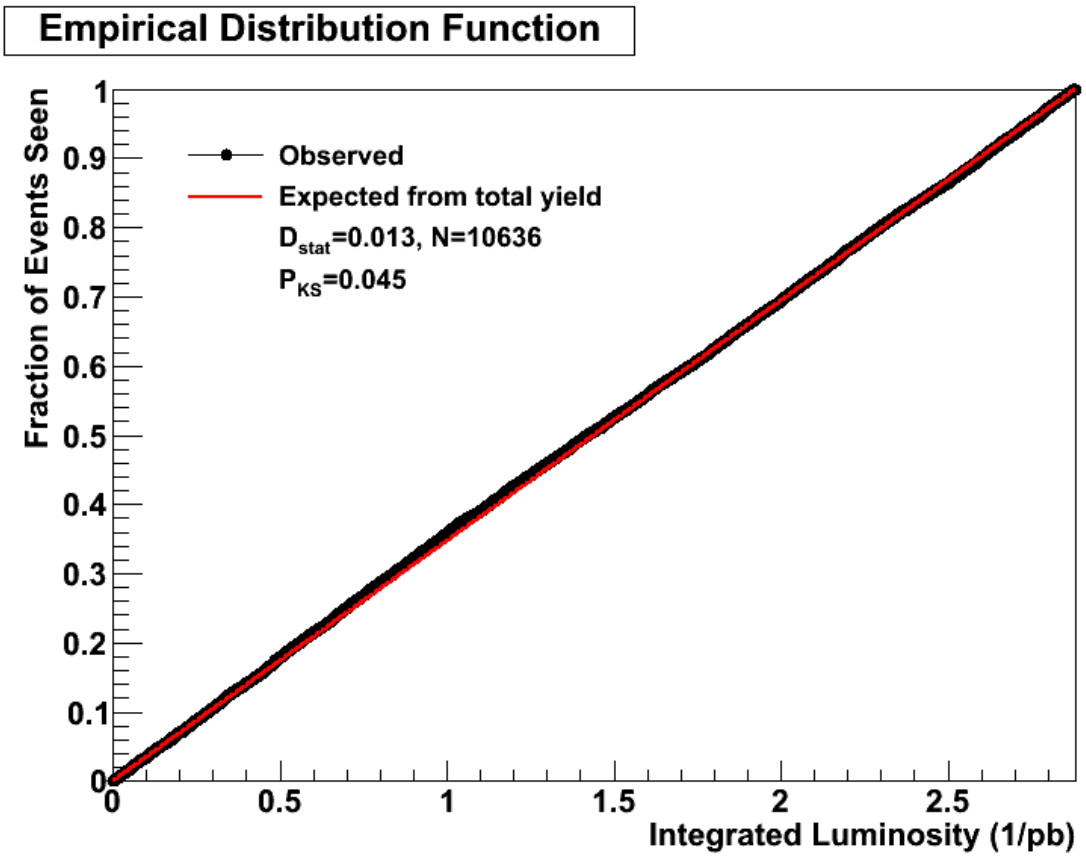
```
cplager@cmslpc17> generateEDF.py 142928-143179.csv ZeeCands_142666_143179.txt Zee.png
loading luminosity information from '142928-143179.csv'.
loading events from 'ZeeCands_142666_143179.txt'
```

- `142928-143179.csv` is luminosity information from `lumiCalc.py` (more below)
- `ZeeCands_142666_143179.txt` is a textfile where each line is *run number*, *luminosity section ID*, and *event number* separated by any combination of commas, spaces, tabs, colons, and semi-colons.
- `Zee.png` is the name of the output. If you want to be able to zoom in, *etc.*, use a =.root. extention (*e.g.,* Zee.root).



D_{stat} is the maximum separation of the two curves. P_{KS} is the resulting probability that the observed data is consistent with the expected curve.

An example with higher statistics looks like this:



Both of these examples do not show any problem.

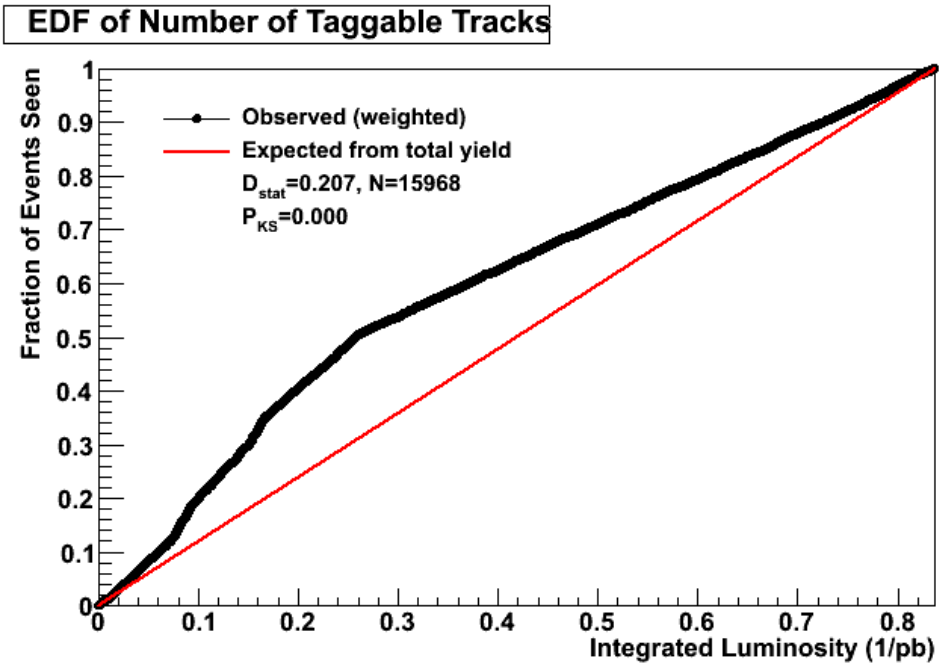
Time-Ordered Example with Weights

the event text file now contains

run, lumi, event, weight

```
generateEDF.py complete_xing.csv run-luminosityBlock-event-n_taggable_fixed_orig.txt taggable_tim
  "--title=EDF of Number of Taggable Tracks" --weights --ignoreNoLumiEvents
```

- `--weights` tells the script to use fourth column
- `--ignoreNoLumiEvents` says to ignore events where there is no luminosity information. In general, this should not happen, but if one doesn't use the quite-right good luminosity JSON file, this can happen.



You'll notice several *kinks* in the curve, suggesting problems. See below to see how one can track down exactly when (*i.e.*, which runs and lumi sections) this has happened.

Instantaneous Luminosity-Ordered Example

Most analyses are concerned with verifying if they are affected by *"pileup"* or not. A very simple way to check is to make an EDF plot, but to order the data from smallest to greatest _Average Crossing Instantaneous Luminosity_ (or AXIL for short) instead of ordering the data from smallest run number to biggest run number.

Why AXIL? AXIL is the quantity that is directly correlated with pileup:

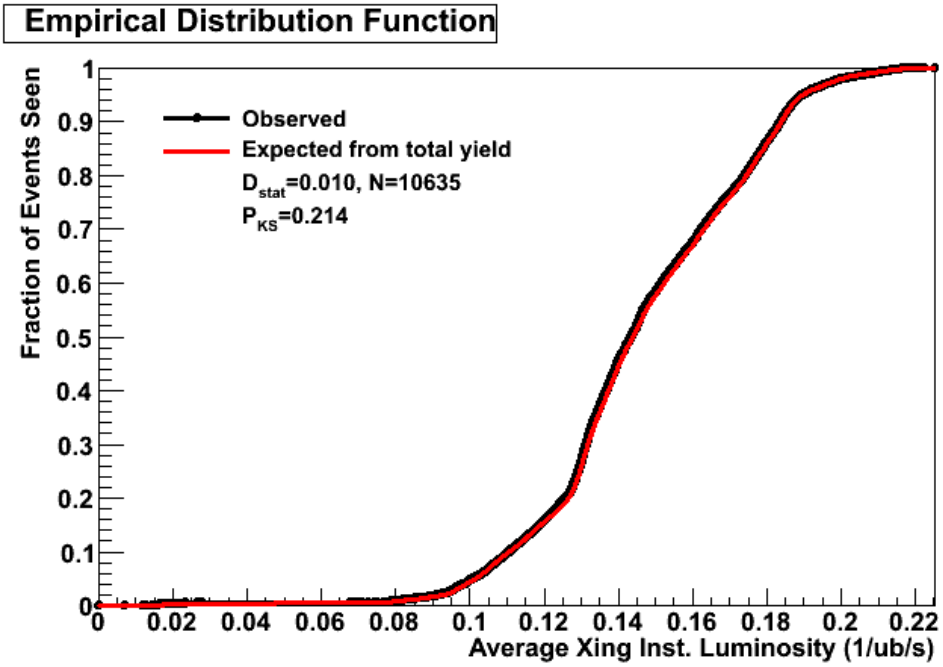
- Assuming a minimum bias cross section 70.3 mb, we expect 1 pileup event for an AXIL of 0.16 $\mu b^{-1}/s$.

To do this, we need a lumi CSV file that has been generated using the `lumibylsXing` option instead of `lumibyls` option. **Note:** Lumi CSV files generated with `lumibylsXing` option will work to generate both time-ordered and AXIL-ordered EDF plots, so there is no need to generate both types of CSV files.

```
generateEDF.py lumiInfo.csv Wenu_Cands_Plager.txt wcand_inst.png \
```

```
--ignoreNoLumiEvents --edfMode=instLum --noWarnings
```

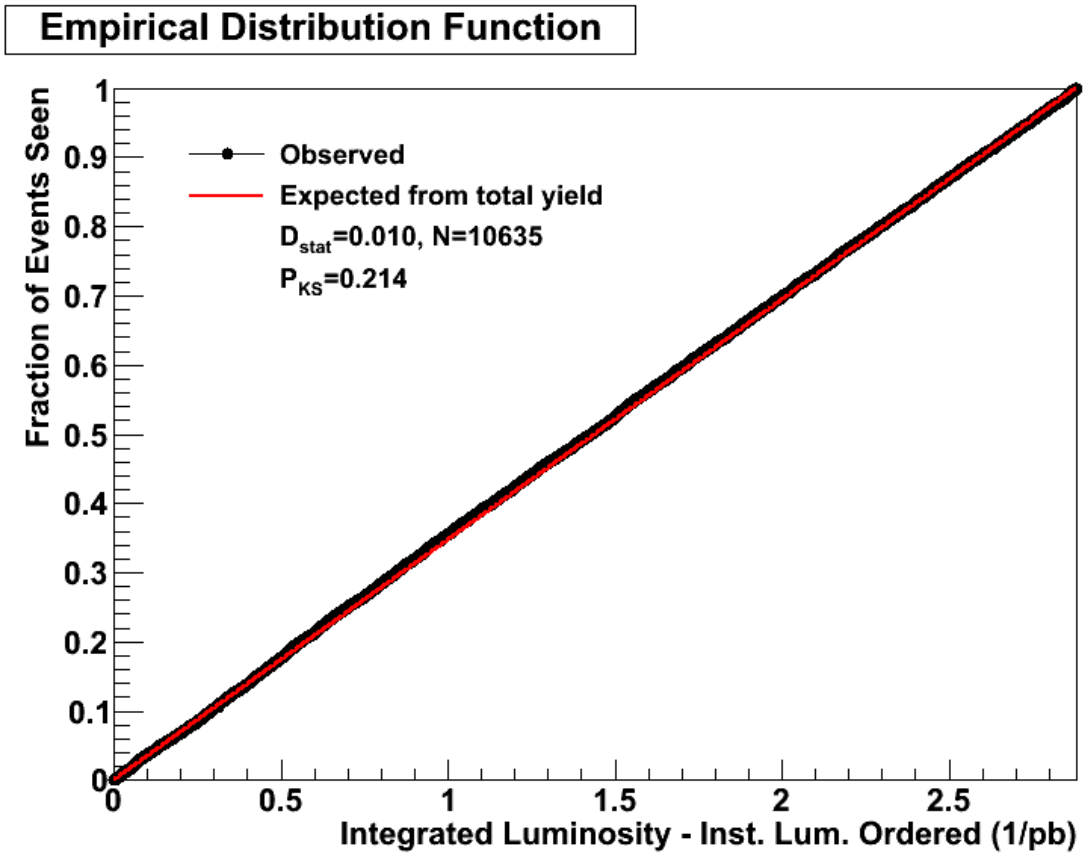
- `--edfMode=instLum` tells script to use AXIL ordering
- `--ignoreNoLumiEvents` and `--noWarnings` tells script to skip events where there isn't luminosity information (a small percentage of lumi sections does not have AXIL information).



In this case, we can see that the red line lies over the black points well (and the KS test has a reasonable value). But for many who are used to looking at the time-ordered EDF plots, these *S* shaped plots can look a little weird.

```
generateEDF.py lumiInfo.csv Wenu_Cands_Plager.txt wcand_inst.png \
   --ignoreNoLumiEvents --edfMode=instIntLum --noWarnings
```

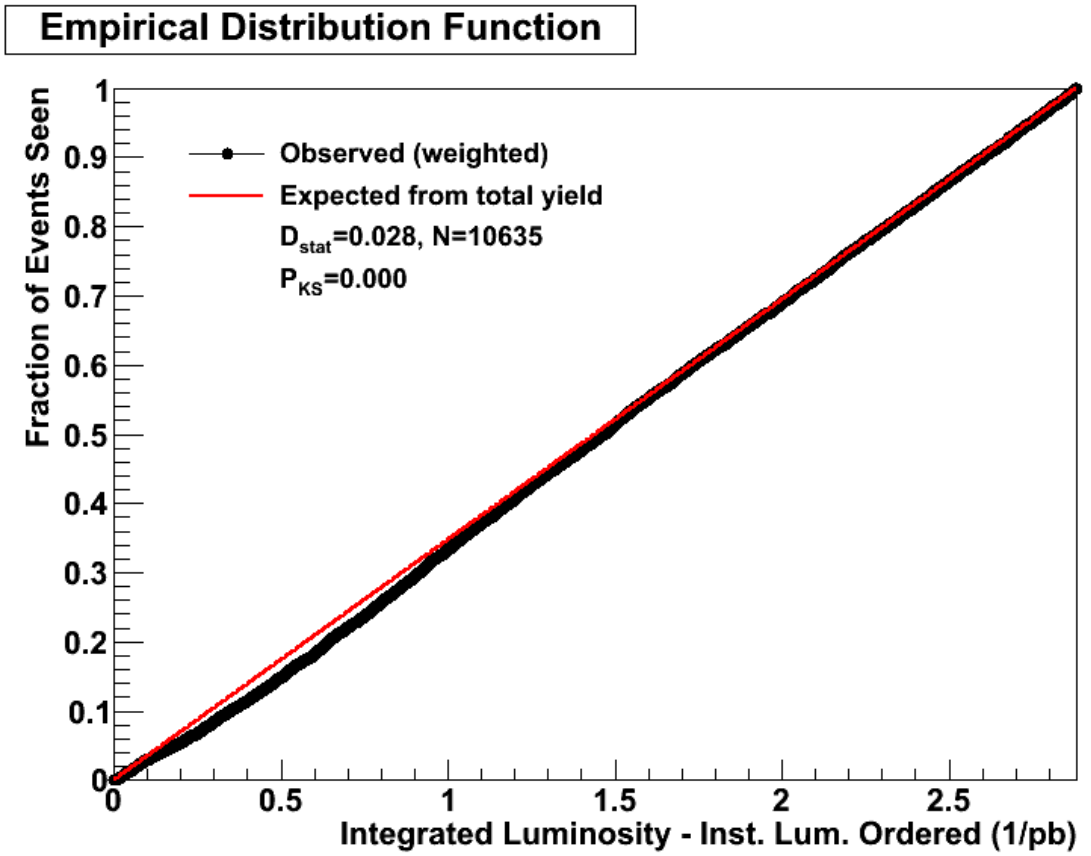
- `--edfMode=instIntLum` tells script to use AXIL ordering, but to use integrated luminosity as the X axis



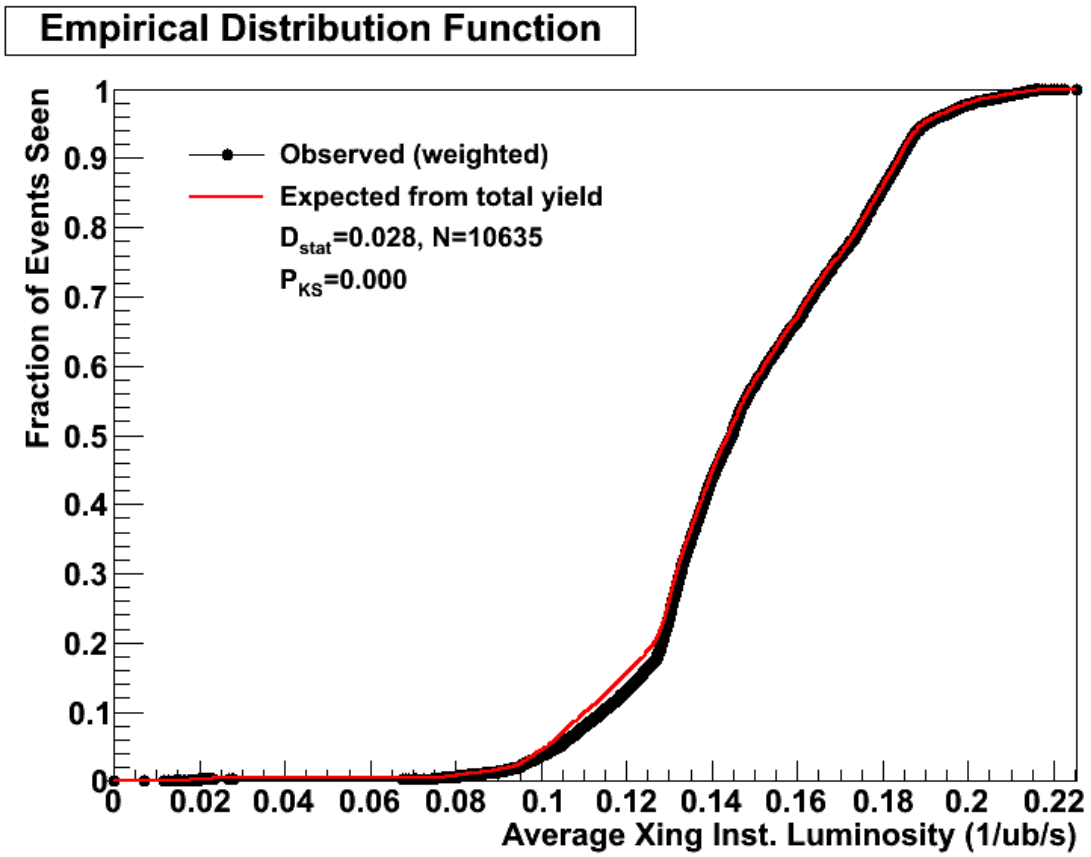
Both of these plots (which have exactly the same KS distance and probability by construction) show that W e does not suffer from pileup effects.

To show an extreme example, we use the number of reconstructed verticies as a weight. This is probably as extreme as an example as we are likely to find since number of reconstructed primary verticies is what many people use to measure pileup.

```
generateEDF.py lumiInfo.csv Wenu_Cands_Plager.txt wcand_inst_S_weights.png \
  --ignore --edfMode=instLum --noWarnings --weights
```




```
generateEDF.py lumiInfo.csv Wenu_Cands_Plager.txt wcand_inst_weights.png \
  --ignore --edfMode=instIntLum --noWarnings --weights
```

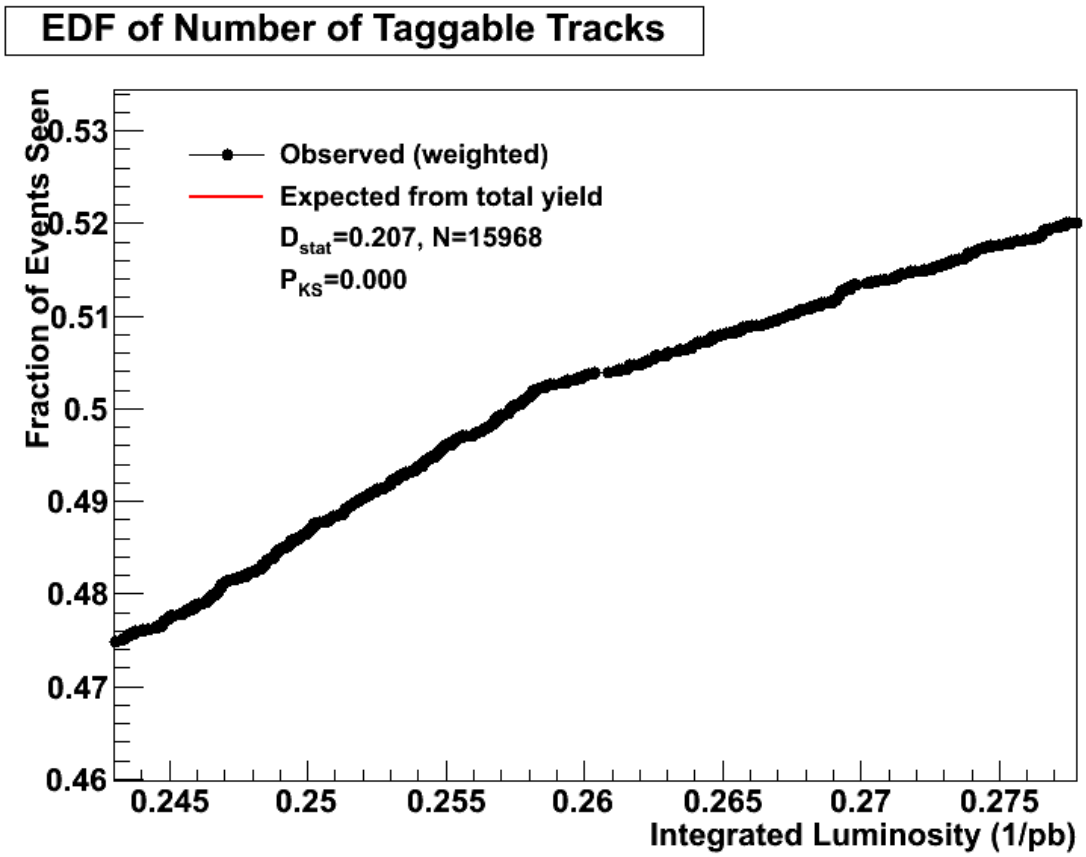


Both these plots show a KS probability of 0. One might have expected a larger discrepency, but at the maximum AXIL we have so far, we only expect, on average, 1.4 pileup events.

More options

Printing Run, Lumi Information Corresponding to Integrated Luminosity

In the example above, we can see an elbow around 0.3/pb. Rerunning the above command, except saving it as `taggable_time.root` instead of `taggable_time.png`, one can open the canvas in Root and zoom in:



O.k. So it looks like it happens around .257/pb or so. What run does that correspond to?

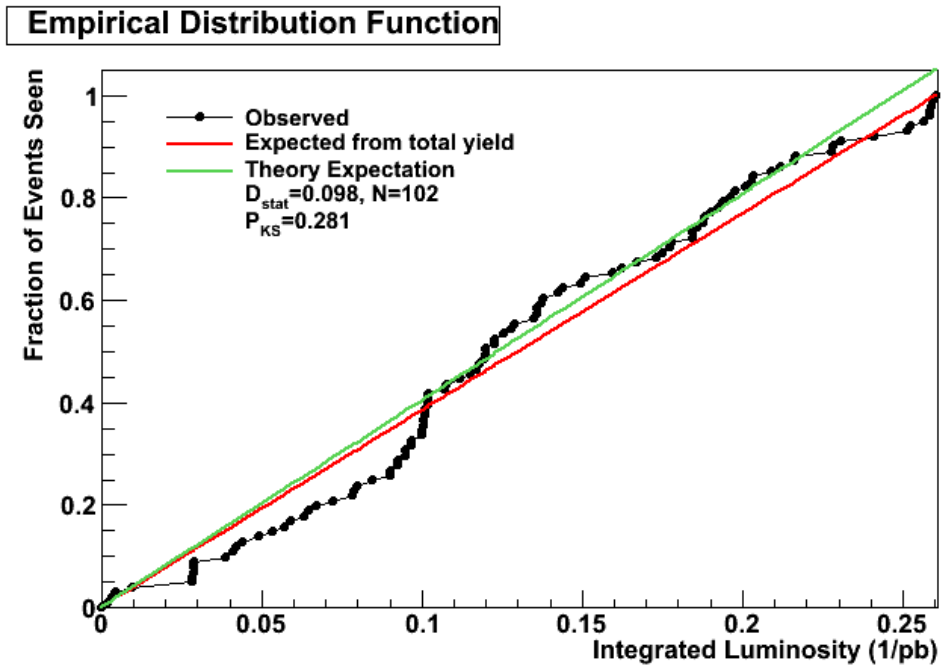
```
cplager@cmslpc17> generateEDF.py complete_xing.csv --runsWithLumis=0.25,0.2525,0.255,0.2575,0.26
(140385, 164) contains total recorded lumi 0.250000
(140387, 65) contains total recorded lumi 0.252500
(140401, 47) contains total recorded lumi 0.255000
(140401, 197) contains total recorded lumi 0.257500
(141956, 225) contains total recorded lumi 0.260000
```

At this point, it looks like something changed around 140401 or 141956. Knowing that several changes were made to the trigger table at 141956 (*e.g.,* changing the primary dataset of the zero bias trigger) suggests that a trigger change could be the culprit.

Overlay Additional Prediction

If you want to overlay another prediction (*e.g.,* say that the theory cross section is 5% higher than the observed number of events suggests),

```
generateEDF.py 142928-143179.csv ZeeCands_142666_143179.txt Zee_other.png \
--predicted=1.05 --predLabel="Theory Expectation"
```



Note that for the KS test, it is always the observed value that is used (*i.e.,* the probabilities are the same on the two above plots).

Now let's use the script to work through an example Ken Bloom and I just went through.

```
generateEDF.py complete_xing.csv run-event-luminosityBlock.txt tagged.png \
  --runEventLumi "--title=EDF of Muon-Tagged Jets"
```

(`--runEventLumi` means read the events as run, event, lumi instead of run, lumi, event).

In this case, what we see is not consistent with the expected curve.

In trying to see is happening, we looked at one of the inputs to tagged jets, the number of taggable tracks. In this case, we don't want to only count number of events, but give each event a weight (in this case, the number of taggable tracks per event). Instead of listing:

run, lumi, event

Generate Lumi CSV Files

For full documentation, please visit Lumi Calculation Twiki.

To generate a CSV file without AXIL information:

```
lumiCalc.py -i Cert_142928-143179.txt -o lumiInfo.csv lumibyls
```

To generate a CSV file **with** AXIL information:

```
lumiCalc.py -i Cert_142928-143179.txt -o lumiInfo_xing.csv lumibylsXing
```

Note that the latter file `lumiInfo_xing.csv` can be used for both time-ordered and AXIL ordered EDF plots.

-- CharlesPlager - 14-Sep-2010

This topic: CMSPublic > SWGuideGenerateEDF
Topic revision: r4 - 2010-09-20 - CharlesPlager