ENHEP Tutorial: Trigger Efficiencies

U. Husemann, M. Schröder, Karlsruhe Institute of Technology

January 28, 2019

In this tutorial, we will study the topic of triggers, which are a crucial tool for data analysis in particle physics. For example, we will learn how trigger efficiencies can be measured.

The tuturial will take place on January 28, 2019, 15:00–16:00, and it will be carried out on computers, running a standalone Python program on a ROOT input file. Therefore, *you are asked to bring a laptop and make sure* ***beforehand*** *that there is a working installation of a recent Python3 version and ROOT6 with PyRoot support* (the exercise has been tested with ROOT version 6.14/08 and Python version 3.6.8). We encourage you to work in small teams of up to three persons, and it is sufficient to have one laptop per group. All further instructions will be provided during the tutorial class.

1 Setting up the environment

You need a working installation of a recent ROOT6 and Python3 version (the exercise has been tested with ROOT version 6.14/06 and Python version 3.6.8). The exercise is performed with the Python program `calculate_eff.py`, which takes the ROOT file `histos.root` as input. Download a gzipped tarball with both files from `https://www.dropbox.com/s/6mu1sirvj2f9p62/trigeff.tar.gz` and unpack it into a local working directory. That's it, you are all set!

2 Measuring the trigger efficiency

In this exercise, we want to measure the efficiency of a high-level trigger path that requires the presence of one jet with transverse momentum p_T above a certain threshold, in our case $p_T > 500\,\text{GeV}$. In CMS, the trigger path is called `HLT_PFJet500`. Since it is this trigger whose efficiency we want to measure, we call it the *probe trigger*.

We measure the efficiency by using a different trigger as the *reference trigger*. In our case, the reference trigger requires the presence of an isolated muon with $p_T > 20\,\text{GeV}$. The corresponding trigger path is called `HLT_IsoMu20`. With this, we can define the efficiency ϵ as

$$\epsilon = \frac{N(\text{reference \&\& probe})}{N(\text{reference})} = \frac{N(\texttt{HLT\_IsoMu20 \&\& HLT\_PFJet500})}{N(\texttt{HLT\_IsoMu20})}\,, \tag{1}$$

where $N(\text{reference})$ denotes the number of events in which the reference trigger fired and $N(\text{reference \&\& probe})$ the number of events in which both the reference trigger and the probe trigger fired.

Motivate the definition (1) by discussing the following questions:

- What is the purpose of the reference trigger?
- Why do we require the muon of the reference trigger to be isolated? What could happen if one considers any muon (also non-isolated)?
- Could we use a trigger that fires at random as the reference trigger?

- Could we, instead of using the single-muon trigger, use a trigger that requires the presence of a jet with a p_T threshold lower than the threshold of `HLT_PFJet500` as the reference trigger, e.g. a trigger requiring a jet with $p_T > 300\,\text{GeV}$?

For reasons that will become clear later, we want to measure the efficiency (1) as a function of the jet p_T: $\epsilon \to \epsilon(p_T)$. The file `histos.root` contains several jet p_T histograms. We first need two histograms: the jet-p_T histograms for all events that fired the reference trigger and for those events that fired both the reference and the probe trigger. Extracting these histograms from `histos.root` is already implemented in the program `calculate_eff.py` (make sure to understand where this happens!). Execute the program with

`python calculate_eff.py`

This will plot the two histograms. Inspect them and discuss the following questions:

- Why does the number of entries per bin decrease towards large p_T?
- What is the reason for the turn-on at low p_T?

Following (1), the efficiency $\epsilon(p_T)$ is then simply the ratio of the two histograms. This is also already implemented (check where!), but the histogram is not drawn. Adjust the script `calculate_eff.py` to also draw the efficiency and execute the program. Inspect the trigger efficiency plot and answer the following questions:

- What is the efficiency of the `HLT_PFJet500` trigger path?
- Why is there a smooth *turn-on* region around 500 GeV where the efficiency gradually increases? Why does the trigger not reach its maximum efficiency instantly at $p_T = 500\,\text{GeV}$?

In the light of the turn-on feature of a trigger efficiency, consider again the question:

- Could we, instead of using the single-muon trigger, use a trigger that requires the presence of a jet with a lower p_T threshold than `HLT_PFJet500`, e.g. a trigger that requires a jet with $p_T > 300\,\text{GeV}$? Which condition must be satisfied when a trigger with lower threshold is used as the reference trigger?

3 Computing the uncertainties of the efficiency

Have a look at the error bars in the efficiency plot produced in the previous exercise:

- Are they reasonable?
- How are they calculated?

The measured efficiency follows a binomial distribution around the true efficiency ϵ^{true}. You can switch to using binomial uncertainties by adding the `"B"` option to the `TH1::Divide` method.

- How do the error bars change?
- Is this reasonable?

The correct way to quote the uncertainties is to use a confidence interval, derived from the measured ϵ (which is an estimator of ϵ^{true}!). There is some discussion in the literature on which confidence intervals to use[1], which also depends on the use case. The PDG recommends to use

[1] See e.g. R.D. Cousins, K.E. Hymes, J. Tucker, *Frequentist Evaluation of Intervals Estimated for a Binomial Parameter and for the Ratio of Poisson Means*, Nuclear Instruments and Methods in Physics Research A 612 (2010) 388–398, doi:10.1016/j.nima.2009.10.156, arXiv:0905.3831

the *Clopper–Pearson* interval. We will follow this recommendation. Computation of Clopper–Pearson (and various other) intervals are implemented in the ROOT `TGraphAsymmErrors::Divide` method[2] (Clopper–Pearson is used when specifying the option `"cp"`, which is also the default). Adjust `calculate_eff.py` to use a `TGraphAsymmErrors` object for the efficiency with Clopper–Pearson intervals as error bars.

- How do the error bars change? Is this reasonable?

4 Determining when a trigger is "fully efficient"

Now we will discuss the question of how to define the trigger efficiency in more detail. Of course, we can use the p_T dependent function and have all information available that is needed in physics analysis. However, often a simplified approach is helpful where we determine the p_T threshold p_T^{thres} above which the trigger has reached its full efficiency.

- Given the turn-on feature, we can assume the trigger efficiency to remain constant for large p_T far above the turn-on region. Why?

Thus, a typical choice is to define p_T^{thres} as that p_T value above which the trigger reaches an efficiency of 99% of its maximum efficiency.

Before we determine p_T^{thres}, consider the following: We have measured the trigger efficiency in bins of jet p_T, and thus, there is the danger that statistical fluctuations in one bin could affect our choice of p_T^{thres}. In order to correct for binning effects, we first want to fit the efficiency with a continuous function. A suitable function is

$$f(p_T; a_0, a_1, a_2) = \tfrac{1}{2} \cdot a_2 \cdot \left[\text{erf}(\tfrac{1}{\sqrt{2}a_0}(p_T - a_1)) + 1\right] \tag{2}$$

with the free fit parameters a_i ($i = 0, 1, 2$) and erf denoting the *error function*. Before applying (2), inspect the proposed function:

- How is the error function defined and why is it suitable in this case? (Remember again what the reason for the turn-on feature was!)
- What is the interpretation of the parameters a_i?

The fit function (2) is already implemented in `calculate_eff.py` as a `TF1` object (where?). Use it to fit the efficiency (using the `TGraphAsymmErrors` object with the correct uncertainties, of course!). You can now read off the trigger threshold p_T^{thres} from the fitted function. Which threshold do you find? What is the efficiency of the trigger above the threshold?

5 Measuring the efficiency of a different trigger

We want to use the tools developed above to measure the efficiency of a single-jet trigger with a different threshold. Adjust your `calculate_eff.py` to measure the efficiency of the `HLT_PFJet60` trigger (the necessary histogram is also provided in the ROOT file `histos.root`). What do you observe?

[2] See `https://root.cern.ch/doc/master/classTGraphAsymmErrors.html`