Final Project Computing Methods in High Energy Physics

Richard Friedrichs, Jan Loder, Shannon Moraes

May 30, 2025

Contents

1 Introduction

In this project we simulate the Drell-Yan Dimuon production at the LHC Run 3 with a centre-of-mass energy of $\sqrt{s} = 13.6\,\text{TeV}$. The main background contribution of this process is the $\mathrm{t\bar{t}}$ production.

In the first step we use the event generator PYTHIA8 to simulate the data and in the second step we analyse the data using the data analysis framework ROOT. In the final step we reflect on our analysis and give ideas on how to improve it.

2 Data generation

The signal process we are looking at in this project is the Drell-Yan dimuon process

$$q\bar{q} \rightarrow Z \rightarrow \mu^+ + \mu^-$$

The decay of the Z boson to the muon antimuon pair is only one of the possible decay channels, but it is the only one we are considering here. The corresponding PYTHIA configuration to simulate this process is

```
1       PythiaParallel pythia;
2
3       pythia.readString("WeakSingleBoson:ffbar2gmZ=on");
4       pythia.readString("23:onMode = off");
5       pythia.readString("23:onIfMatch = 13 -13");
6       pythia.readString("PhaseSpace:mHatMin = 60.");
7       pythia.readString("PhaseSpace:mHatMax = 120.");
8       pythia.readString("Beams:eCM = 13600");
9
10      pythia.init();
```

This is only an excerpt of our data generation; all of our code can be found in our git repository. Already from this we can see what physical settings were used to generate the signal. First we are using a parallelised version of PYTHIA to speed up the simulation. Furthermore, we are forcing the decay of the Z boson to the muon antimuon pair and have imposed $60\,\text{GeV}$ and $120\,\text{GeV}$ as a lower and upper phase space limits respectively. This can be seen from lines 4 and 5 of the above code excerpt. This means that our simulation is not realistic in the sense that we do not allow all possible standard model decays to happen but rather only allow decays that are interesting to us. The reason for doing this is that we want to generate a reasonable number of interesting events without having to simulate too many events. The branching ratio of the dimuon decay is $\Gamma_{\mu^+\mu^-}/\Gamma_{\text{tot}} = 3.3662\,\%$.[1] That means that in a simulation without forcing

[1]Value for this and following branching ratios taken from `https://pdg.lbl.gov/index.html`

only about 3 % of events would be interesting to us which means that we discard a very big chunk of the generated data. In order to avoid wasting that much computing power we are forcing the decay into the muon antimuon pair.

The background process we are simulating is $\mathrm{t\bar{t}}$ production. The processes for this are

$$\begin{aligned} gg &\to t\bar{t} \\ q\bar{q} &\to t\bar{t} \\ t &\to W^+ b \\ \bar{t} &\to W^- \bar{b} \\ W^+ &\to \mu^+ \nu_\mu \\ W^- &\to \mu^- \bar{\nu}_\mu \end{aligned}$$

Again these are not all of the possible decays. The top quark almost exclusively decays into a W boson and a bottom quark. The branching ratio of this decay is about 95 %. Therefore there is no need to force any decays here. The decay of the W boson on the other hand is a little bit more involved. Our desired decay into a (anti)muon and a (anti)muon neutrino has a branching ratio of only $\Gamma_i/\Gamma = 10.63\,\%$. Therefore, for the same reasons as above, we choose to force this decay in our simulation. The PYTHIA settings for simulating this background are

```
    PythiaParallel pythia;

    pythia.readString("Top:gg2ttbar = on");
    pythia.readString("Top:qqbar2ttbar = on");
    pythia.readString("24:onMode = off");
    pythia.readString("24:onIfMatch = -13 14");
    pythia.readString("-24:onMode = off");
    pythia.readString("-24:onIfMatch = 13 -14");
    pythia.readString("PhaseSpace:mHatMin = 60.");
    pythia.readString("Beams:eCM = 13600");

    pythia.init();
```

Again we choose to run in parallel mode to speed up the simulation. This time we do not impose an upper phase space limit since the top quark is the heaviest standard model particle.

For both the background and signal simulation we save the data in separate ROOT files containing a ROOT Tree with branches for: transverse momentum p_T, polar angle θ, particle ID, azimuthal angle ϕ and the trigger for the conditions $p_\mathrm{T} > 20\,\mathrm{GeV}$ and $|\eta| < 2.1$. The trigger branch contains a boolean per event, detailing whether the event passes the trigger or not. Filtering with this trigger will be done in the following section.

To see how many events we need to simulate, we started out with a lower number of events and then ran the filtering scripts on the background. Our target of background events that survive the filters was 150.000×10^3 to ensure that we have enough events for statistics. By trying a different number of events we found that 1 million events surpassed that filter. Therefore we decided to simulate 1 million events, which is also in agreement with what other groups have presented in the progress report.

3 Data analysis

Before the data can be analysed, it needs to be filtered. This is done to both reduce file sizes and increase the proportionality of interesting events.

3.1 Trigger

At a real detector, only events, which pass selection triggers, get registered in the first place. This is done to reduce data size and screen for interesting results (as is the case for all filters and will be omitted moving forwards). Whether an event passes a trigger is stored in the HLT_DoubleIsoMu20_eta2p1 branch. If the PYTHIA generator simulates the events happening inside the detector, then the trigger_filter.py simulates the action of the detector capturing events, which pass the trigger.

Below you can see the difference between the simulated events and the events passing the trigger.

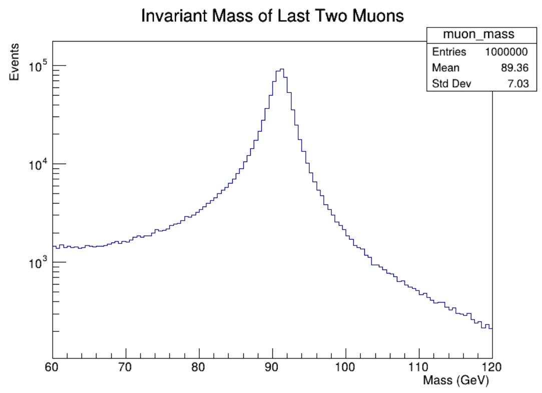

Figure 1: Invariant mass of the last two muons in each event.

One can see the total amount of generated events in the amount of entries to the histogram: 1000000.

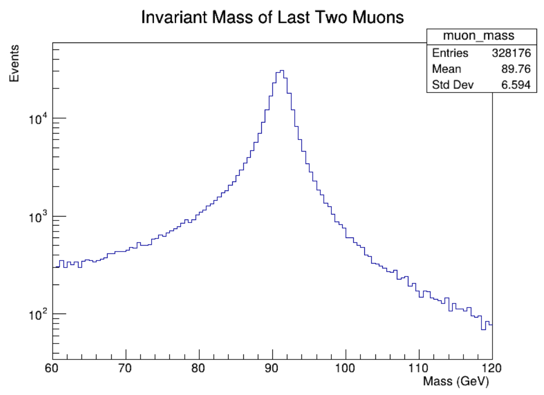

Figure 2: Invariant mass of the last two muons in each event passing the filter.

One can see that the amount of events passing the filter is smaller: 328176. These two allow one to calculate the trigger efficiency: about 32.82%.

Furthermore, as a consequence of uninteresting events being filtered out, the shape of the distribution changes very slightly.

3.2 Detector Smear

Real detectors aren't perfectly accurate: they can make mistakes when measuring the parameters of particles passing through them. Although detector simulation programs like GEANT exist, a simpler alternative will be adapted. By assuming the hypothetical detector to have measuring uncertainties of 1% in p_T and 2 mrad in ϕ, θ, detector smear is simulated by gaussian_smear.py. The script takes the parameters of muons and scrambles them according to Gaussian distributions centred at their previous values with aforementioned uncertainties. This approach is minimal, but sufficient for our goals.

3.3 Additional Filters

At real colliders, the amount of data coming in is orders of magnitudes higher than the amount, which can be recorded. To combat this, a second layer of filters (considering the trigger being the first) screens events for recording. Whereas the filter can be considered hardware related (physical) these triggers are slower and software related (digital). Two additional filters are simulated: a stricter filter for muon momenta and the isolation filter.

3.3.1 Stricter Filter

This filter (applied by stricter_filter.py) screens for events with exactly one dimuon (muon pair), where the p_T of both muons is higher than 30 GeV/c.

3.3.2 Isolation Filter

This filter (applied by isolation_filter.py) screens for events, where both of the muons of interest have less than 1.5 GeV/c worth of charged pions in their vicinities ($\triangle R < 0.3$) combined. This excludes muons in jets, for example.

3.4 Aftermath

A quick look at the signal before and after filtering reveals how the peak stands out clearly at the location we'd expect to find the Z mass resonance at.

Our simulation yields 206621 signal events passing all the filters (out of 1 million).

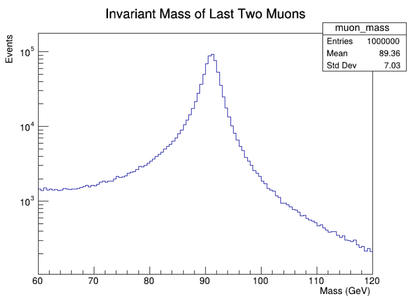

Figure 3: Signal invariant mass profile before filtering.

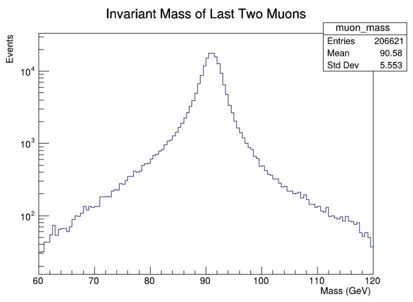

Figure 4: Signal invariant mass profile after filtering.

Compared to the signal, the background appears less clear: no distinct peaks.

Our simulation yields 182540 background events passing all the filters (out of 1 million).

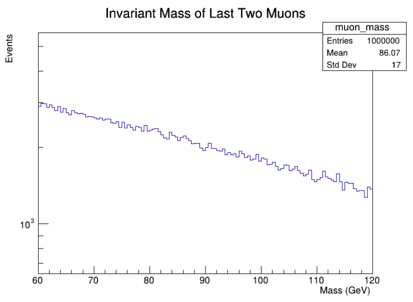

Figure 5: Background invariant mass profile before filtering.

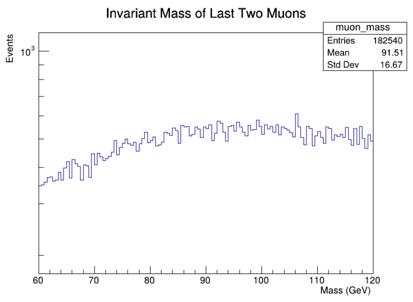

Figure 6: Background invariant mass profile after filtering.

We see, that in both cases, only about 20% of events survive the filtering process.

3.5 Combination of Datasets

To scale the histogram such that each entry corresponds to the correct amount of cross section in fb we need to include scale factors for the histograms. The scale factors are given by

$$S_i = \frac{\sigma_i}{N_i} \ ,$$

where i labels whether we have signal or background. Then the combined histogram is given by

$$\text{combined histo} = S_{\text{sg}} \times \text{signal histo} \ + \ S_{\text{bg}} \times \text{background histo} \ .$$

For the cross sections we choose to use the hypothetical values instead of the real PYTHIA cross section because this makes the background fit easier and the combined histogram look more like a typical background + signal histogram. These values are $\sigma_{\text{sg}} = 62\,\text{pb}$ and $\sigma_{\text{bg}} = 924\,\text{pb}$. Combining signal and background with their respective weights gives us a new combined histogram.

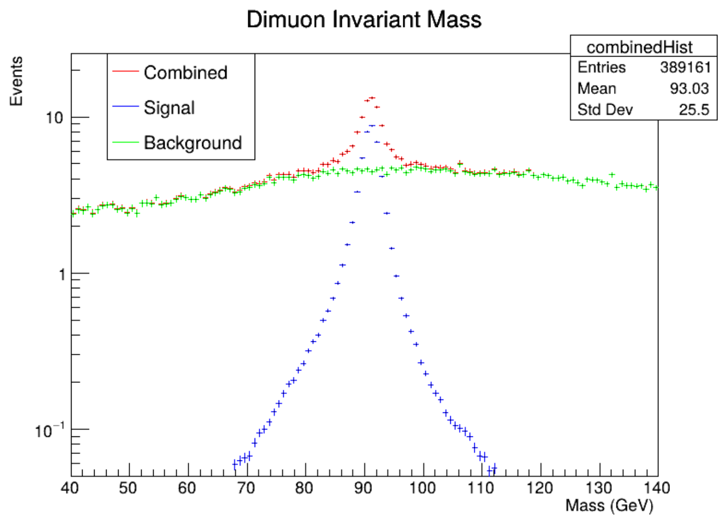

Figure 7: Combined histogram

3.6 Fitting

The function used to model the data is a sum of a Landau distribution and a Gaussian distribution. The Landau function describes the background and the Gaussian function describes the signal. Taking them together we get the fit function for the combined histogram

$$\begin{aligned} f(x) &= A_L \cdot \text{Landau}(x; \mu_L, \sigma_L) + A_G \cdot \text{Gauss}(x; \mu_G, \sigma_G) \\ &= A_L \cdot \frac{1}{\sigma_L} \cdot L\left(\frac{x-\mu_L}{\sigma_L}\right) + A_G \cdot \frac{1}{\sqrt{2\pi}\sigma_G} \exp\left(-\frac{(x-\mu_G)^2}{2\sigma_G^2}\right) \end{aligned} \tag{1}$$

Where $L(z)$ is the Landau probability density function. It does not have a simple analytical form, but is defined via a convolution of an exponential and a Gaussian. It is usually evaluated numerically and is included as function in ROOT. The fit function f has 6 parameters:

- A_L: Amplitude of the Landau distribution
- μ_L: Most Probable Value of the Landau distribution
- σ_L: Width of the Landau distribution
- A_G: Amplitude of the Gaussian peak
- μ_G: Mean of the Gaussian distribution
- σ_G: Standard deviation of the Gaussian.

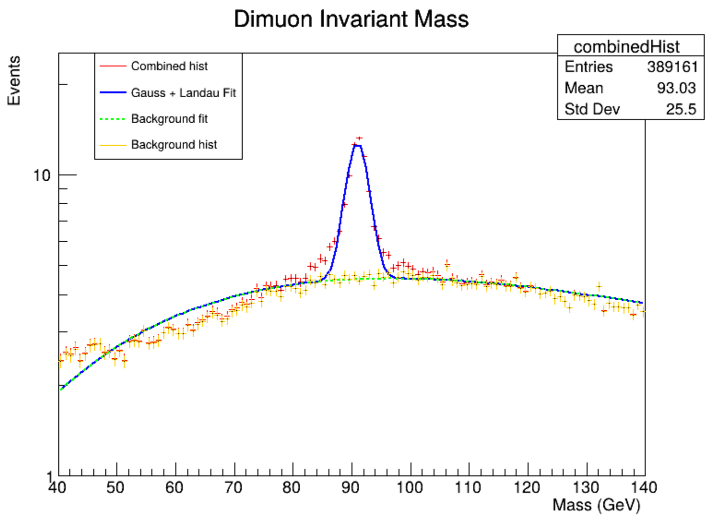

Figure 8: Histogram with fit function

Figure 8 shows the result of the fit and the fit parameters including their uncertainty are listed in Table 1.

Table 1: Fit parameters for Landau + Gaussian model

Parameter	**Value ± Uncertainty**	
A_L	952.695	± 12.151
μ_L	105.204	± 0.658
σ_L	38.148	± 0.572
A_G	38.050	± 0.543
μ_G	91.002	± 0.025
σ_G	1.847	± 0.032

We can see that there is a visible peak at mass $M = (91.002 \pm 1.847)\,\text{GeV}$. This peak can be associated with the Z boson which has a known mass of $M_Z = (91.188 \pm 0.002)\,\text{GeV}$.[2]

In the next step we evaluate the significance s of the signal by using the formula

$$s = \frac{N_S}{\sqrt{N_B}} \; , \tag{2}$$

where N_S is the number of signal events and N_B the number of background events in a mass window of our choosing. By looking at Fig. 8 and Tab. 1 we choose the mass window $[81.002\,\text{GeV}, 101.002\,\text{GeV}]$, which is a $\pm 10\,\text{GeV}$ interval around the Gaussian fit's peak position.

We get the number of events by integrating the fit function and the background function over that mass range. Doing so we get a significance of

$$s = \frac{45.660}{\sqrt{107.099}} \approx 4.412 \; . \tag{3}$$

This shows us that we are still short of the desired 5σ we need for a proper discovery. We know that this significance would get better if we simulate more events/let the experiment run for longer. This can easily be seen by considering the following. Let us assume we run the experiment for twice the time which means that we will get twice as many events for signal and background. This implies that we need to multiply N_S and N_B by a factor of 2 which leads to an overall factor of $\sqrt{2}$. In general this means we can include a weight factor

$$w = \frac{N_{\text{target}}}{N_{\text{simulated}}} \tag{4}$$

to get

$$s = \frac{N_S}{\sqrt{N_B}} \sqrt{w} \; . \tag{5}$$

To find out how many events need to be simulated or in a real experiment how many data needs to be taken for a 5σ discovery we reverse engineer this factor

[2]Value taken from `https://pdg.lbl.gov/index.html`

w by requiring $s = 5$. This leads to

$$w = \left(\frac{5}{s}\right)^2 \approx 1.284 \tag{6}$$

which means that we should have simulated about 1.3 million events instead of just a million.

With this we can now estimate how much integrated luminosity we need for this discovery by considering the equation

$$N_{\text{tot}} = \sigma_{\text{sg}} \cdot \mathcal{L} \ , \tag{7}$$

where $\mathcal{L}$ is the total integrated lumonosity and σ is the cross section of the signal. Using $N_{\text{tot}} = N_{\text{simulated}} \cdot w$ and $\sigma_{\text{sg}} = 62.000 \times 10^3\,\text{fb}$ we find

$$\mathcal{L} = 20.710\,\text{fb}^{-1} \ . \tag{8}$$

Now taking a look at the total integrated luminosity of CMS run 3 in Fig. 9 we find that this luminosity was reached in about middle of May, after about one and a half months of data taking.[3]

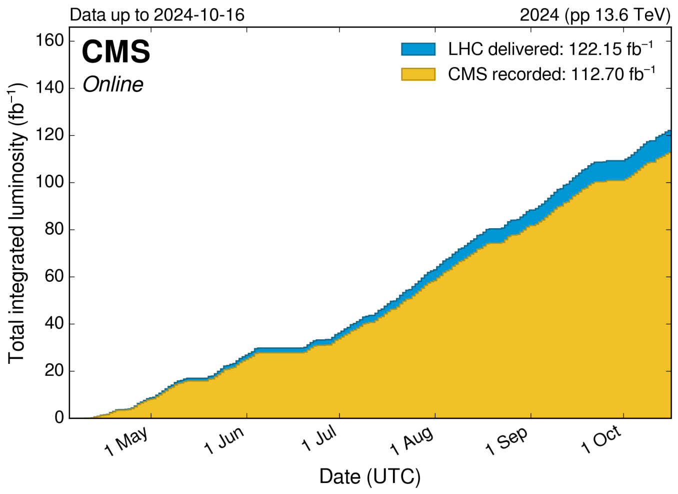

Figure 9: CMS run 3 total integrated luminosity

4 Conclusion

If this analysis were based on real data, it would need to meet the standards upheld by the global high-energy physics community. However, several aspects

[3]Figure taken from `https://cern.ch/cmslumi/publicplots/2023/int_lumi_per_day_cumulative_pp_2023_Normtag.png`

of the current study already form a credible and convincing foundation. The methodology follows physically motivated procedures with realistic selection criteria that align well with the capabilities of the CMS detector. These include the use of a symmetric double-muon trigger, a transverse momentum cut of 30 GeV, and a track isolation condition, all of which reflect typical event selection strategies seen in published analyses.

To approximate detector effects, we applied a Gaussian smearing to the muon momentum and angular measurements. While simple, this step helps bridge the gap between idealized generator-level simulations and more realistic detector-level responses, bringing the kinematics closer to what one would expect in practice. The analysis also includes a clear separation of signal and background $t\bar{t}$ events, followed by a fit to the invariant mass distribution in order to extract the relevant yields. To estimate the significance of the Z boson peak, we use the common metric $\frac{N_S}{\sqrt{N_B}}$. Admittedly, this is a simplified approach, but it serves well as a first estimate of the signal's visibility.

Of course, there are some quite significant limitations that need to be acknowledged before such an analysis could be considered realistic by the standards of experimental collaborations. The study currently does not include a full detector simulation, no pileup modeling is performed, and there is no treatment of systematic uncertainties, all of which are integral to modern-day analyses. Furthermore, the statistical significance is calculated without accounting for fit uncertainties or selection efficiencies. Another important point is that the analysis is not blinded, which introduces the risk of observer bias—something real-world collaborations take very seriously and are careful to avoid.

There is certainly room for improvement. Incorporating a fast detector simulation framework would offer a more accurate and computationally efficient way to include detector effects such as acceptance and resolution. It would also be important to quantify and include systematic uncertainties, those related to trigger and reconstruction efficiencies, as well as luminosity, for example, in any final significance estimate. Pileup conditions, especially those expected during LHC Run 3, should be modeled realistically since they can impact track isolation and event selection. Additionally, validating the background fit using alternative techniques, such as checks in control regions, would help strengthen confidence in the result and guard against potential biases in the signal extraction.

In summary, while this analysis does not yet reach the level required for a publishable CMS result, it demonstrates a well-structured and physically sound approach to signal extraction and background estimation. With the suggested improvements, the analysis could gain both statistical and experimental robustness. As it stands, it serves as a strong prototype that convincingly demonstrates the feasibility of identifying the Z boson in the dimuon channel using LHC Run 3 data.

5 Manual of Use

This section describes how to recreate our results. We will focus on the FewVariables version, which only saves the Trigger, Pt, Phi, Theta, Id as branches, since this version runs faster and has smaller file sizes. In the FewVariables directory you can find a readme that gives a brief instruction on how to run the code. If you are seeking for a more in-depth explanation of how to run the code we refer to this chapter.

5.1 Data generation

You're going to want to start by accessing the Makefile: make sure the PYTHIA8_DIR points to your PYTHIA directory. Then everything is set.

To generate the signal and the background independently, use "make compile", "make makesignal", "make makebackground" and then run "./makebackground >>background.output" or "./makesignal >>signal.output". This will write all output from PYTHIA to output files which you can access to get the cross sections. The simulation of 1 million events with 8 threads should take about one hour each.

5.2 Filtering

We will thoroughly describe the filtering process for Signal.root. The same process should be followed for Background.root separately.

What follows are a list of commands to activate various python scripts for filtering events and simulating detector defects. When using them, consider using them in the same order (as it best simulates how filtering is done at a particle collider) and mind replacing "python" with "python3" or "py" or other prescripts your system has for activating python scripts.

- python trigger_filter.py Signal.root
- python gaussian_smear.py filtered_Signal.root
- python stricter_filter.py Smeared_filtered_Signal.root
- python isolation_filter.py Stricterfiltered_Smeared_filtered_Signal.root

Here's what each step does:

- Rewrites into filtered_Signal.root only the events which passed the trigger. This is meant to simulate which events even get registered by the detector.
- Rewrites into Smeared_filtered_Signal.root each event from filtered_Signal.root after applying a gaussian smear of 1% to the Pt and 2mrad to Phi and Theta. This is meant to simulate the inaccuracies in the detector measurements.

- Rewrites into Stricterfiltered_Smeared_filtered_Signal.root only the events, which have exactly two muons with Pt above 30 GeV. This is meant to simulate a second, stricter filter, which is applied digitally to reduce the amount of data stored and increase the percentage of interesting events.
- Rewrites into Isolated_Stricterfiltered_Smeared_filtered_Signal.root only the events, for which the two electrons have little charged pion activity near them. This is also simulating a stricter filtering criterion, which is applied digitally to reduce data size and increase the proportion of interesting events.

Since these filters (+smear) don't discriminate between which dataset the events are a part of, the outcome should be the same whether the filtering is done prior to or after combining the background and signal datasets. In our case, it could be easier to do them separately, but in real detectors the datasets are combined from the start.

5.3 Graphing

At any point between the filtering steps you can graph out the invariant mass of the muon pair (in case the event has yet to be thrown out by filtering for having more than 2 electrons of appropriate energy, only the last 2 electrons are taken) by using "python plot_invariant_muon_mass.py currentdataset.root". The output will be a matching currentdataset.png.

5.4 Analysis

To run the analysis you type "make compile", "make analysis" and "./analysis >>fit.output". It is important that the files "Isolated_Stricterfiltered_Smeared_filtered_Signal.root" and "Isolated_Stricterfiltered_Smeared_filtered_Background.root" are located in the same directory as the "analysis.cpp" file.

If these requirements are met running the commands above will create two images in your current folder titled "dimuon_mass.png" and "dimuon_mass_fit.png" and a textfile called "fit.output". The former image shows three histograms in the same plot, which are the correctly scaled background, the correctly scaled signal and scaled background + scaled signal. The latter image will show the combined histogram of signal and background, a fit of a Landau + Gauss function as defined in Eq. (1) to the combined histogram and a fit for the background only based on the fit parameters of the combined histogram fit. The textfile "fit.output" will contain all fit parameters and other output produced by analysis.cpp.