Université Catholique de Louvain

LPHY2131 : Particle Physics I

CMS OpenData lab: measuring Z and W bosons

Vincent Lemaitre
Christophe Delaere

Contents

1 Introduction

1.1 Welcome to the LPHY2131 wiki!

This project assembles material for the LPHY2131 lab at UCLouvain. This wiki also documents the three sessions of the laboratory and provides some additional information.

The results that are obtained in this lab can be compared to the published cross-section measurement for the Z and W at the LHC, at 7TeV, by the CMS collaboration: Measurement of the Inclusive W and Z Production Cross Sections in pp Collisions at sqrt(s) = 7 TeV

To install, follow the instructions below:

1. Download the virtual machine Image for 2011 CMS open data from `http://opendata.cern.ch/VM/CMS#how`
2. After booting, change the keyboard layout (if needed)

- In the menu on the top left, select Settings/keyboard
- In the Layout tab, select what you need (in UCLouvain we have Belgian keyboards)

1. In a terminal (black icon at the bottom of the screen), run the following two commands. Note that the second command starts with a dot followed by a space.

- git clone `https://github.com/delaere/LPHY2131.git`
- . LPHY2131/install.sh

This prepares the environment for the lab:

- downloads samples files and example scripts
- installs CMSSW 5.3.32 (used for 2011 data processing)
- installs CMSSW 7.1.1 (used to setup the analysis environment)
- installs MadGraph 5
- installs Delphes 3.3.2

1.1.1 Credits:

Thanks to Liliya Milenska, Gilles Parez and Martin Michel for their help in preparing this activity during their internship in summer 2015.
Thanks to Victor Massart and Julien Touchèque for their help in porting the lab to 2011 data in summer 2016.
Thanks to Jérôme de Favereau, Pavel Demin and Michele Selvaggi for their support.

2 Day 1 - Understanding the data and simulation workflows

2.1 Looking at the 2011 CMS data

Data comes in different formats, from bulky raw data to compact processed and skimmed data.
In CMS, it goes like this (for that period):

```
RAW -> RECO -> AOD -> PAT -> Ntuples
```

- RAW data is a binary format. It is compact but proprietary and requires a lot of knowledge about the detector electronics to be decoded.
- RECO data contains higher level objects (tracks, calorimeter clusters, electrons, ...). It typically takes a lot of disk space (because low-level data is kept) but can be the starting point of data reprocessing.
- AOD (for Analysis Object Data) is a thinner data format where only the high-level calibrated objects are retained. It is centrally produced. These are the AOD files that CMS made public.
- PAT (for Physics Analysis Toolkit) is an even more simplified dataformat, typically produced by each analysis team. They typically contain only the subset of objects needed for a given study.
- Ntuples are simple data tables generally containing simple float quantities. Very compact, these are the files that we usually use in the interactive analysis.

In this lab, we will inspect AOD files with the event display tool of CMS, then produce a small ntuple. In the second lab, we will analyze larger ntuples produced by our IT team in advance of the lab.

2.1.1 Event visualization

In a fresh terminal, run the following:

```
cd CMSSW_5_3_32/src/
cmsenv
cmsShow root://eospublic.cern.ch//eos/opendata/cms/Run2011A/DoubleElectron/AOD/12Oct2013-v1/20000/0014CE62-9C3E-E311-8FCC-00261894389F.root
cmsShow root://eospublic.cern.ch//eos/opendata/cms/Run2011A/DoubleMu/AOD/12Oct2013-v1/20000/045CCED6-033F-E311-9E93-003048678F74.root
```

1. What kind of events do you see?
2. Do you see electrons or muons?
3. You may try to filter the collection, for example asking for one or more muons... note that it takes time.

2.1.2 Processing data to create a ntuple

The data directory contains a sample of type "PAT". It contains some
preprocessed events, similar to the AOD just visualized, but filtered and slimmed down (only the high-level info). We will process it and create a ntuple with the information relevant for this lab.

Take some time to look at the code in Labo/WeakBosonsAnalyzer/src/WeakBosonsAnalyzer.cc. Most of the job is done in the method
WeakBosonsAnalyzer::analyze.
This analysis code accesses the information in the PAT (using the CMS library) and create a simple file that can be read by the ROOT standalone software.

To run the code on few events:

```
cd Labo/WeakBosonsAnalyzer/test
cmsRun weakBosonsAnalyzer_cfg.py
```

By defaults, this runs one one file containing 21000 CMS event passing the double muon trigger. You can change that by editing the configuration file, weakBosonsAnalyzer_cfg.py to either point to another file or to run on a subset of events.

When this is done, you can run ROOT to look at the result:

```
root -l LPHY2131tree.root
root [0] new TBrowser
```

Take some time to get familiar with the various quantities in the ntuple.

2.2 Simulating Monte Carlo (MC) events to compare to

To understand what the data contains, a standard approach consist in simulating event using Monte Carlo techniques to reflect the best of our understanding of the theory. By emulating the detector response and the analysis workflow, we obtain files that can be compared to data.

In this case, we will use MadGraph to simulate events of interest (pp->Z->l+l- and pp->W->lnu), pass them through a simplified detector simulation (Delphes) and produce a ntuple with the same format as for data.

2.2.1 Running MadGraph

Start MadGraph in a new terminal:

```
cd CMSSW_7_1_1/src/
cmsenv
cd ../..
cd MG5_aMC_v2_4_3/
./bin/mg5_aMC
```

Generate Drell-Yann events:

```
MG5_aMC>generate p p > l+ l-
MG5_aMC>output ppNeutralCurrents
MG5_aMC>open index.html
MG5_aMC>launch ppNeutralCurrents
```

On the first prompt, enable “Pythia”.

PYTHIA is a computer simulation program for particle collisions at very high energies.
Some of the features PYTHIA is capable of simulating are Hard and soft interactions, Parton distributions, Initial/final-state parton showers, Multiple interactions, Fragmentation and decay, etc.
PYTHIA is used here to perform the hadronization of final state quarks, and to simulate both initial state and final state radiations.

Various parameters can be modified in the run card. This goes from the number of events (the default 10k is perfect in our case), some cuts (for example on the minimum Pt of some particles), or the beam energy.

Then, repeat the operation to generate W bosons leptonic decays:

```
MG5_aMC>generate p p > l- vl~
MG5_aMC>add process p p > l+ vl
MG5_aMC>output ppChargedCurrents
MG5_aMC>open index.html
MG5_aMC>launch ppChargedCurrents
```

Again, on the first prompt, enable "Pythia".

By default, and as just described, the simulation is performed at leading order (LO). To perform the simulation at next-to-leading order (NLO) it is enough to add "[QCD]" at the end of the process string. In that case, Pythia is not used, as the hadronization is performed internally by another method adapted to NLO.

Then, we quit:

```
MG5_aMC>quit
```

Finally, we will unzip the results:

```
gunzip ppChargedCurrents/Events/run_01/tag_1_pythia_events.hep.gz
gunzip ppNeutralCurrents/Events/run_01/tag_1_pythia_events.hep.gz
```

2.2.2 Running Delphes

You just generated MC events, with sets of particles as they would emerge from the collisions. The next step is to emulate the detector response. We use for that Delphes, a standalone public tool that emulates a simplified version of the CMS detector.

```
cd ../Delphes-3.3.2
cp ~/LPHY2131/analysis/delphes_card_CMS_mod.tcl cards/
./DelphesSTDHEP cards/delphes_card_CMS_mod.tcl ppChargedCurrents.root ../MG5_aMC_v2_4_3/ppChargedCurrents/Events/run_01/tag_1_pythia_events.hep
./DelphesSTDHEP cards/delphes_card_CMS_mod.tcl ppNeutralCurrents.root ../MG5_aMC_v2_4_3/ppNeutralCurrents/Events/run_01/tag_1_pythia_events.hep
```

2.2.3 Visualizing the events

We can now look as the events as we did for data:

```
root -l examples/EventDisplay.C'("cards/delphes_card_CMS.tcl","ppChargedCurrents.root")'
root [0] .q
```

What are the similarities and differences between the two types of events?

2.2.4 Processing the events to generate a ROOT ntuple

Finally, we will produce a ntuple similar to the one we have for data, using the `CreateTreeFromDelphes` script. You can briefly look at it and then run the conversion:

```
cp ~/LPHY2131/analysis/CreateTreeFromDelphes.C .
root -l
root [0] gSystem->Load("libDelphes.so")
root [1] .L CreateTreeFromDelphes.C++
root [2] CreateTreeFromDelphes("ppChargedCurrents.root","delpheAnalysisW.root")
root [3] CreateTreeFromDelphes("ppNeutralCurrents.root","delpheAnalysisZ.root")
```

You can look at the resulting files... how does it compare to what we have for data?

3 Day 2 - Measuring the Z boson and determining the luminosity

3.1 Goals

In this lab, we want to

- see the Z boson
- measure the mass and width; understand the limitations of the approach
- determine the purity (estimate the background)
- determine the luminosity, from the number of Z events observed
- study the jet multiplicity if these events, and compare to MC predictions

Half of the class will work with electrons, the other half with muons. We will then compare.

3.2 Preparation for the lab

We have to set up the environment and to copy some files to the work area (more precisely, we will create symbolic links to the files available).

```
cd CMSSW_7_1_1/src/
cmsenv
cd
cd LPHY2131/analysis
./prepare.sh
cd labo2
ls
root -l
```

So, we use the same procedure than on day 1 to setup the environment, move back to home, move to the analysis directory, run the prepare.sh script (that creates links to root files in the labo2 and labo3 subdirectories), move to the labo2 subdirectory and finally start ROOT.

3.3 Selection of the Z events

We have ntuples for data and Monte Carlo events (Drell-Yann events produced with MadGraph).
Using these files, one can determine the selection criterias needed to get a pure sample, keeping the efficiency high. Monte Carlo files generated by CMS using a detailed simulation of the detector are provided in addition to the sample that was produced in the first lab. That CMS sample has a cross-section of 2475/pb, and contains originally 40'000'000 events.

We are looking for Z bosons decaying into either electrons or muons. How would you select such events based on the content on the ntuples produced on day1?

In ROOT, start a TBRowser and open the ROOT file for data (electron or muon sample) and simulations (the delphesAnalysisZ.root that you created on day 1).

```
root [0] new TBrowser
```

In the browser, open the file and the ntuple in the file (double click). By clicking on variable names, you get plots of that quantity for all events in the file. In the console, it is easy to draw the variables for a subset of events:

```
root [1] WeakBosonsAnalysis->Draw("ElectronsPt[0]", "nElectrons==2")
```

In that example, you draw the Pt of the first electrons (electrons are sorted by Pt) for events with two electrons. By default, the previous histogram is replaced when you issue that command. To superimpose two plots, add a third argument with the `"same"` string:

```
root [2] WeakBosonsAnalysis->Draw("ElectronsPt[1]", "nElectrons==2", "same")
```

In that example, you superimpose the Pt distribution of the second electrons. Remember that the electrons are sorted in Pt.

You can also do 2D plots of one variable against the other:

```
root [2] WeakBosonsAnalysis->Draw("ElectronsPt[1]:ElectronsPt[0]", "nElectrons==2")
```

Many more possibilities are available. Check the ROOT documentation.

A natural selection will consist in asking two leptons of the same flavor. A cut on the isolation and on the Pt of these leptons might also be a good idea. Where should we cut? You can make up your mind by comparing data to Monte Carlo, interactively. Would you suggest additional cuts? What about jets and missing transverse momentum?

When preparing the cuts, keep in mind that the Monte Carlo simulation is imperfect. Don't blindly cut on quantities there without making sure that this makes also sense on data.

3.4 Determination of the purity

Even with a good selection, there may be remaining background events. To estimate this, we will do a fit of the data with two templates:

- invariant mass distribution of Z events from MadGraph
- invariant mass distribution for the background, with an ad hoc analytical distribution.
  This will gives us an estimate of the purity.

The fit is easily done with ROOT, but requires some familiarity with the tool. A script is provided to do the fit. It returns the main results in the console. After editing the script, simply run it by executing:

```
root -l Zyield.C
```

Things that should be adapted in the script are:

- the proper input data file should be uncommented
- the binning (min/max/number of bins)
- the cuts (that you just decided at the previous step)
- the analytical form for the background contribution, and the value of the parameters.

3.5 Determination of the luminosity

Knowing the number of selected events, the purity, the Z cross-section (from MadGraph) and the selection efficiency, it is easy to get the luminosity:

N_Z = sigma * L * efficiency = N_data * Purity

For this, we have of course to assume that the efficiency in data and simulations are the same.

- the cross-section is given by MadGraph and can be found in the MadGraph folder where you produced the MC sample: `file:///home/cms-opendata/MG5_aMC_v2_4_3/ppNeutralCurrents/crossx.html`
- in case you use the official CMS sample, remember that it has a cross-section of 2475/pb, and contains originally 40'000'000 events.
- the efficiency is given by the number of MC events passing your cuts divided by the number of generated events (10000)
- N_data, Purity, and N_Z (the product of the two) are given by the`Zyield.C` script.

Good to know: we run on a subset of the 2011 data. The total luminosity recorded in 2011 was 5.55/fb, but the sample available for this lab (2011A, corresponding to data up to 21/08/2011, represents 2.676/fb).

3.6 Other studies

1. From a fit of the Z mass distribution (use the `makePlot.C` script below), determine the Z boson mass and its width. Does it match your expectations?
2. Look at events with one jet. What is the angle between the lepton pair and the jet? How does the Pt of the jet and the Pt of the lepton pair compare. Is that expected? What is the origin of that jet?
3. Look at the jet multiplicity. Does the MC reproduce the data? What if you generate events at NLO? Try to fit with a power law or an exponential. Why does it work?
4. Compare electrons and muons samples. Are the results the same?
5. Look at events with one electron and one muon. Do we expect such events from a Z boson? Ask for two additional jets. What are these events in data? N.b.: obviously, this has to be done using the single lepton samples, not the dilepton samples linked by default in the labo2 directory.

3.7 ROOT tricks and scripts

ROOT is a powerful but complex tool to do data analysis. Basics operations are simple and will be exemplified below, but much more advanced things can be done.
Sample scripts are also provided for some of the more advanced actions, but will have to be sometimes adapted to the precise use in this lab.
Complementary documentation can be found in a dedicated page and on the ROOT webpage.

3.7.1 Plotting and filtering

Simple plotting can be performed by using the Draw method of the ROOT TTree object. It takes three arguments:

- the variable (or combination of) to draw
- the cuts used to filter the events
- the drawing options (graphical)

In the expression for the variable, one can also redirect the result in an existing histogram by using the `variable>>histoName` synthax. This is an easy way to (1) define the binning (range + number of bins) (2) have access to the resulting histogram later on in the console or in a script.

We propose that you run root in the analysis directory. Sample scripts are located in the examples directory.

Typical session:

```
gROOT->LoadMacro("functions.C");
gStyle->SetOptFit(111111);
TFile* data_file = TFile::Open("LPHY2131tree.root");
data_file->cd("LPHY2131analysis");
WeakBosonsAnalysis->Draw("nJets","nElectrons==2 && ElectronsPt[0]>20")
```

3.7.2 Fitting

We are also interested to do a fit of the Z peak with the proper analytical function. This can be done interactively using the fit tool (right click on the histogram you want to fit) but is limited to basic functions.

Some more functions are defined in `function.C`, like the Voigt function, well suited in our case. To use it:

```
TF1 *func = new TF1("myVoigt", myVoigt ,low, high,4);
func->SetParameter(0,10.0);   func->SetParName(0,"const");
func->SetParameter(1,90.0);   func->SetParName(1,"mean");
func->SetParameter(2,3.0);    func->SetParName(2,"sigma");
func->SetParameter(3,3.0);    func->SetParName(3,"gamma");
htemp->Fit("myVoigt","LLR","SAME");
```

3.7.3 Provided scripts

Two scripts are provided to automatize some of the analysis steps. They can be used both as tools and as examples for writing your own scripts or for interactive analysis.

Extracting the MC scale factor from a fit to data: `Zyield.C` That script performs the following actions:

- Fills histogram for MC and data for the chosen variable with chosen cuts.
- Fits MC to data, taking into account a background contribution described by an analytical function.
- Returns the fraction of signal in data (from the fit) and the scale factor to apply to MC to get the proper normalization.

It is configured by editing the constants at the beginning of the script. You should already know about the selection cuts before using that script.

Combined display: `makePlot.C` That script performs the following actions:

- Fills histogram for MC and data for the chosen variable with chosen cuts.
- Does a fit of the data with a Voigt function
- Displays Data, MC and the fit in a relatively nice way.

Note that the script does not display the background contribution (if any).

4 Day 3 - Measuring the W boson and determining its cross section

4.1 Goals

In this lab, we want to

- see the W boson
- estimate the background from data using a control region
- determine the W boson cross-section from the event yield and the luminosity determined previously

Half of the class will work with electrons, the other half with muons. We will then compare.

4.2 Preparation for the lab

We have to set up the environment.

```
cd CMSSW_7_1_1/src/
cmsenv
cd
cd LPHY2131/analysis/labo3
ls
root -l
```

The content of the directory was already prepared by running the prepare.sh script on day 2.

4.3 Selection of the W events

We have ntuples for data and Monte Carlo events (W+ and W- events produced with MadGraph). We will use a full-simulation sample from CMS instead of the Delphes sample from day 1 to make sure that the efficiency is under control. We can then compare with the Delphes sample. Using these files, one can determine the selection criteria needed to get a pure sample, keeping the efficiency high.

In ROOT, start a TBRowser and open the ROOT file for data (electron or muon sample) and simulations (the delphesAnalysisW.root that you created on day 1).

```
root [0] new TBrowser
```

While we were looking at the invariant mass for the Z, this is not possible here, as the neutrino escapes undetected. Still, by momentum conservation we can estimate the transverse component of its momentum. Using the lepton 4-vector and the missing transverse momentum, one defines the transverse mass. This variable will be used here.

```
root [1] WeakBosonsAnalysis->Draw("transvMass", "nElectrons>=1")
```

You will notice that it is much more difficult to get a pure sample. It is also more difficult to simulate that background, mostly driven by QCD events with leptons.
We will therefore measure the background shape in a control region (to be defined) and then do a template fit using the background shape from data on one side and the signal shape from MC on the other side.

We are looking for W bosons decaying into either one electron or one muon, plus one neutrino. How would you select such events based on the content on the ntuples produced on day1?

A natural selection will consist in asking one lepton and some missing transverse momentum. A cut on the isolation and on the Pt of the lepton might again be a good idea. Where should we cut? You can make up your mind by comparing data to Monte Carlo, interactively. Would you suggest additional cuts? What are the expected specificities of the kinematic of a W event?

4.4 Choice of the control region where to measure the background

As already discussed, there is an irreducible background that comes from QCD events, with leptons produced in jets or wrongly identified (fakes).

Since the background cannot easily be estimated from simulations, we propose to define a control region where it can be measured.
That control region should be

- as similar as possible to the signal region
- as much as possible free of any signal event
  Especially, as we will use the control region to derive the shape of the background for the transverse mass, the kinematics should be the same in both regions.

Given all of the above, how would you define the control region?

4.5 Determination of the cross-section

The fit is easily done with ROOT, but requires some familiarity with the tool. A script is provided to do the fit. It returns the main results in the console. After editing the script, simply run it by executing:

```
root -l Wyield.C
```

Things that should be adapted in the script are:

- the binning (min/max/number of bins)
- the cuts for the signal and control regions (that you just decided)

That script performs the following actions:

- Fills histogram for MC and data for the chosen variable with chosen cuts.
- Fills an histogram with an estimate of the background, obtained from data in a control region.

- Fits MC to data.
- Returns the fraction of signal in data (from the fit) and the scale factor to apply to MC to get the proper normalization.

Knowing the number of selected events, after background substraction, the luminosity measured in lab2, and the selection efficiency (from MC), it is easy to get the cross-section:

```
N_W = sigma * L * efficiency = N_data * Purity

sigma = N_data * Purity / (L * efficiency)
```

For this, we have of course to assume that the efficiency in data and simulations are the same.

- N_data, Purity, and efficiency are given by the`Wyield.C` script.
- the luminosity is the one obtained on day 2.
- the MC sample used contains 78'347'691 events, for a cross-section of 2.584E4 pb

How does the cross-section obtained that way compare to the value obtained from MadGraph?
That one can be found on that local link: `file:///home/cms-opendata/MG5_aMC_v2_4_3/ppChargedCurrents/crossx.html`

4.6 Other studies

1. What is the charge ratio between W+ and W-? Do the results depend on the charge?
2. How could we determine the W boson mass?
3. Does the jet multiplicity behave as for the Z boson studied last time? Do the results depend on the jet multiplicity?

5 Linux Primer

The virtual machine that we run for this lab runs Scientific Linux 5.
More specifically, it runs a slim version of it. The file distributed by CERN does not exceed 15MB. It is build on a network drive and will need the network at all time.

Interested in going further with linux? Give it a try with one recent distribution (Fedora, Ubuntu, Mint, or other). Keep in mind that slc5 is more than 5 years old!

Linux is not much different from any operating system, but we will use a lot the console to navigate through the directories and issue commands. We then run in what is called a shell.

5.1 Shell commands

Some basic commands that will be useful:

```
pwd (print current directory);
cd DIRECTORY (move to directory DIRECTORY)
ls [-ltrh] DIRECTORY (list files in directory DIRECTORY)
cp [-r] FILE DIRECTORY (copy file FILE to directory DIRECTORY)
mv FILE1 FILE2 (rename FILE1 into FILE2)
rm [-ir] FILE (remove FILE)
mkdir DIRECTORY (create new directory)
```

When navigating accross directories, remember that "." represents the current directory. ".." represents the parent directory. Compared to the OSes of the MS family, the directory separator is "/" instead of "\".

5.2 Editing files

To edit files, you have several options:

- **vi or vim**
  vi runs fully in the terminal. It is available on most of the UNIX-like systems, which makes it the default choice on many systems. MadGraph uses vi to edit the config files. A dedicated page on this wiki is available to help you start with vi.
- **emacs**
  emacs is another very well known editor in the UNIX universe. It is more user-friendly than vi.
- **leafpad**
  leafpad looks like the notepad in Windows. It is definitely the simplest and easiest editor in the list.

All the three can be launched directly from the terminal or from the "start menu" at the bottom left of the screen.

5.3 About the copy/paste

If there is one thing easy in Linux, this is the copy paste. Just forget about ctrl-c/ctrl-v, that's not needed anymore. To copy, just highlight the text. It is "in the mouse". To paste, click on the central button (on the wheel).

6 ROOT Primer

ROOT is a modular scientific software framework. It provides all the functionalities needed to deal with big data processing, statistical analysis, visualization and storage. It is mainly written in C++ but well integrated with other languages such as Python and R. (from the ROOT website)

ROOT are be the main tool used in parts two and three of this lab, and you are encouraged to get familiar with it. Its use is ubiquitous in particle physics, as it is at the core of almost all the experimental analysis frameworks. Here are some basic keys to start using it.

ROOT is simply started by issuing the ROOT command from the directory where you wand to work. The environment has to be set beforehand, in our case by using the `cmsenv` script in the proper release directory:

```
cd CMSSW_7_1_1/src
cmsenv
cd ~/analysis
root
```

You can also directly open a root file or execute a script by passing it to the command line. Passing the `-l` option removes the splash popup:

```
root -l  #don't show the splash
root myfile.root  #opens the file directly
root myscript.C  #executes myscript.C directly
```

Once started, ROOT presents you an interactive shell, where you can issue commands.

We will use ROOT to produce histograms from ntuples (called trees in ROOT), and to do various kinds of fits. A ntuple is a data structure very similar to a table. For each event (row) it contains a set of valyes (columns). One column can also contain a vector of values, and the type of each column (called branch in ROOT) can be different (int, float, char, even custom types).

Here is a typical session:

```
[cms-opendata@localhost analysis]$ root -l delpheAnalysisZ.root
root [0]
Attaching file delpheAnalysisZ.root as _file0...
root [1] new TBrowser
(class TBrowser*)0xc9b270
root [2] .ls
TFile**         delpheAnalysisZ.root
 TFile*         delpheAnalysisZ.root
  OBJ: TTree    WeakBosonsAnalysis      WeakBosonsAnalysis : 0 at: 0x1812050
  OBJ: TTree    WeakBosonsAnalysis      WeakBosonsAnalysis : 0 at: 0x1900650
  KEY: TTree    WeakBosonsAnalysis;1    WeakBosonsAnalysis
root [3] WeakBosonsAnalysis->Scan()
************************************************************************************************************************
*    Row   * Instance *    nMuons *   MuonsPt *  MuonsEta *  MuonsPhi * nElectron * Electrons * Electrons * Electrons *
************************************************************************************************************************
*        0 *        0 *        0 *           *           *           *         1 * 15.572341 * 0.3791305 * -2.803331 *
*        1 *        0 *        0 *           *           *           *         2 * 32.556041 * 2.3843421 * 1.4344033 *
*        1 *        1 *        0 *           *           *           *         2 * 29.342258 * 0.4682755 * -2.499644 *
*        2 *        0 *        0 *           *           *           *         2 * 46.182983 * -0.064993 * -0.630331 *
*        2 *        1 *        0 *           *           *           *         2 * 43.471630 * -0.475959 * 2.2772684 *
*        3 *        0 *        0 *           *           *           *         1 * 13.715698 * -0.248964 * -1.724608 *
*        4 *        0 *        0 *           *           *           *         1 * 35.757526 * -0.108560 * -2.099552 *
*        5 *        0 *        0 *           *           *           *         1 * 11.523519 * -1.868670 * 2.0663385 *
*        6 *        0 *        0 *           *           *           *         2 * 44.405941 * 0.5547180 * -1.455895 *
*        6 *        1 *        0 *           *           *           *         2 * 43.797855 * -0.366821 * 1.8280531 *
*        7 *        0 *        0 *           *           *           *         0 *           *           *           *
*        8 *        0 *        0 *           *           *           *         2 * 22.132816 * -0.105786 * 2.0497729 *
*        8 *        1 *        0 *           *           *           *         2 * 21.133201 * 1.4819486 * -1.283928 *
*        9 *        0 *        0 *           *           *           *         1 * 59.511413 * 1.2132505 * 1.7350055 *
*       10 *        0 *        0 *           *           *           *         2 * 29.598985 * 0.9521472 * -3.116238 *
(...)
Type <CR> to continue or q to quit ==> q
************************************************************************************************************************
(Long64_t)25
root [4] new TCanvas
root [5] WeakBosonsAnalysis->Draw("ElectronsPt[0]","nElectrons>=2 && nMuons==0")
root [6] .q
```

In that example, ROOT is started and one file is opened. A TBrowser is then created, where one can visualize the content of the file and do basic plotting. One then go back to the console and list the content of the file with the `.ls` command. One sees that the file contains a TTree object, on which we can call the `Scan` method. This shows the first branches for the first events. We then create a new canvas (a window where to draw), and call the Draw method on the tree, to see the transverse momentum of the first electron, for events where there are at least two electrons but no muon. Finally, we quit ROOT by issuing the `.q` command.

As already mentioned, commands can either be issued interactively in the console, or assembled in a script (written in C++). Examples of scripts are found in the analysis/examples directory.
Many more or less advanced examples can be found on the ROOT website.

In this lab, we do mostly use the following ROOT classes:

Class	Use
TFile	Manipulate ROOT files (open, create, save, close, ...)
TCanvas	Create a window where to draw/plot
TH1F	Represents histograms
TF1	Represents functions
TLegend	Represents the legend object (optional)

Class	Use
THStack	Allows to stack histograms to represent various contributions

Please refer to the ROOT documentation for details on the use of these classes.

7 Vim Primer

When editing the run card in MadGraph, the editor used is Vim.

Vim (a contraction of Vi IMproved) is designed for use both from a command-line interface and as a standalone application in a graphical user interface. Vim is free and open source software and is released under a license that includes some charityware clauses, encouraging users who enjoy the software to consider donating to children in Uganda.

Although Vim was originally released for the Amiga, Vim has since been developed to be cross-platform, supporting many other platforms. In 2006, it was voted the most popular editor amongst Linux Journal readers.

Like vi, Vim's interface is not based on menus or icons but on commands given in a text user interface. Here are few basic things to remember that you will need for this lab:

- by default, vim is in command mode and does not allow editing. Press `i` to enter *insert mode*. When you are done editing, press `ESC`.
- once you are done editing, press `:wq` to *write and quit*.
- while in command mode, you can initiate a search by pressing `/`, followed by the string to search. You navigate through the search results by pressing `n` (forward) or `N` (backward).

(partly adapted from Wikipedia)

8 The analysis tree

List and meaning of the branches stored in the analysis TTree:

Branch name	Content
nMuons	Number of muons
MuonsPt	Transverse momentum of muons
MuonsEta	Pseudorapidity of muons
MuonsPhi	Azimuthal angle of muons
MuonIsolation	Relative isolation of muons
nElectrons	Number of electrons
ElectronsPt	Transverse momentum of electrons
ElectronsEta	Pseudorapidity of electrons
ElectronsPhi	Azimuthal angle of electrons
ElectronIsolation	Relative isolation of electrons
nJets	Number of jets

Branch name	Content
JetsPt	Transverse momentum of jets
JetsEta	Pseudorapidity of jets
JetsPhi	Azimutal angle of jets
MET_pt	Missing transverse momentum
MET_phi	Azimuthal angle of the missing transverse momentum
MET_eta	Pseudorapidity of the missing momentum (MC only, should not be used)
invMass	Invariant mass of lepton pairs
transvMass	Transverse mass built of the leading lepton + MET

Branch name	Content
dileptonPt	Transverse momentum of the dilepton pair
dileptonEta	pseudorapidity of the dilepton pair
dileptonPhi	azimuthal angle of the dilepton pair
dileptondeltaR	distance in eta-phi between the two leading leptons
dileptondeltaPhi	distance in phi between the two leading leptons
Wcharge	charge of the leading lepton