Basic data analysis with ATLAS

Eirik Gramstad

Research Based Particle Physics(FYS5555)
University of Oslo

February 5, 2020

What are we doing today?

1. half: "Lecture". Will try to answer these questions:

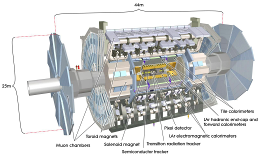
- How is the road from a *pp* collision in ATLAS to the data files you will analyze? (Very briefly!)
- What types of data do we have?
- What is ATLAS Open Data and how can you access it?
- How is the data organized?
- How to get started with Jupyter notebook analysis?

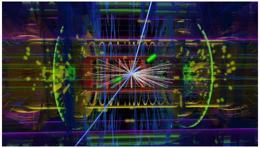
Please interrupt and ask questions along the way!

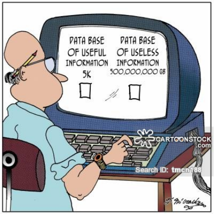
2. half: Practical work. Start playing with Jupyter notebook and ATLAS Open Data.

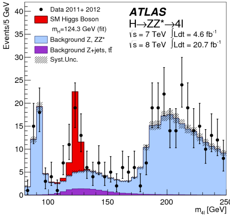
How do we go from a *pp* collision in ATLAS to the datasets we analyze at our PC's?

pp collisions in ATLAS ⇒ Magic happens ⇒ We can analyze the data









What is really happening in the “magic” part?

Triggers

- When the LHC is running at full speed:
  Collision rate of 40 MHz = one collision per 25 ns.
- Storage capacity of ∼1 kHz ⇒ we can only keep about one in 40,000 events. How do we decide which events to keep?
- Answer: triggers. Very complicated and extremely important!
- Main purpose of the triggers: tell (almost) instantly if an event is worth keeping or not.
- Triggers implemented both in hardware and software.
- Main focus of ATLAS: looking for new heavy particles ⇒ triggers mainly focused on high-p_T objects.

Reconstruction

- Aim: figure out what was going on in the detector when the event was triggered.
- Reconstruct objects found in the event: leptons, jets, photons, following specific requirement to how these objects should look like in the detector.
- Determine particle charges, momentum, energy, etc.
- Output: Large ($\sim$PB sized) datasets. Quite incomprehensible to deal with for an analyzer $\Rightarrow$ Needs further "slimming".

Derivations and nTuples

- A lot of ATLAS analysis groups with different needs in terms of information and variables in the dataset.
- $\Rightarrow$ *derivations* are made for analysis groups, designed to suit their needs as good as possible.
- Output: $\sim$100 different formats of $\sim$TB size. Still a bit incomprehensible to deal with on your computer.
- Each group (or individual analyzer) usually make $\sim$MB-GB sized *nTuples* from the derivations, which are possible to work with locally.
- The nTuple format is the one **YOU** will work with!

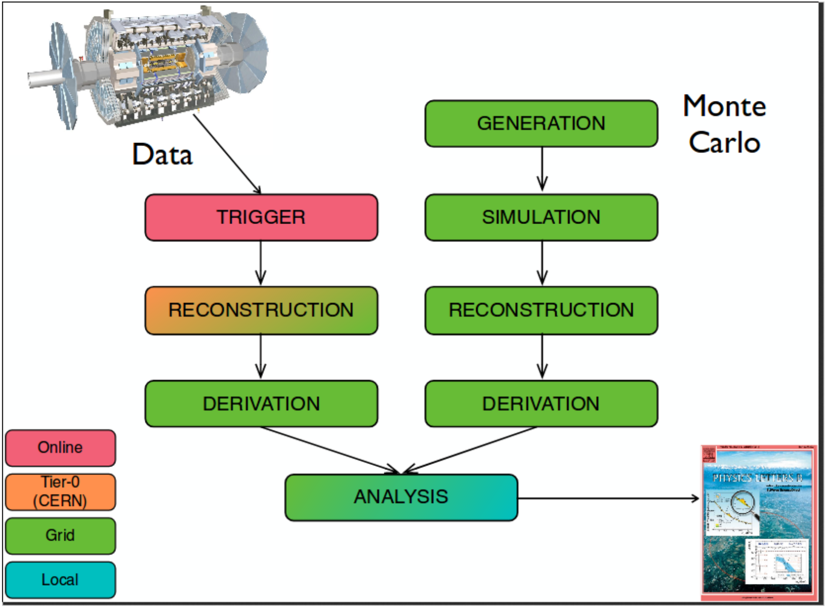
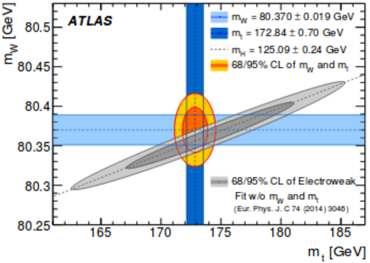


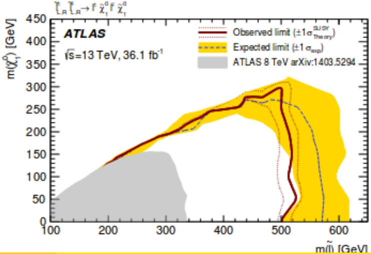
Figure from James Catmore (HEP seminar 30/08-2018).

So.. our final data format has been produced. But what now?

Analysis approaches

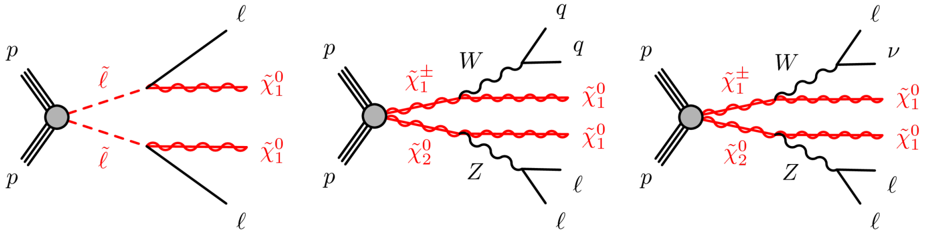
- Precision measurements
  - Measuring particle properties more precisely.
    - Masses, coupling constants, mixing angles, cross sections etc.
  - Deviation from SM ⇒ new physics?
- Searches for new physics
  - Searching directly for some specific beyond SM physics.
    - Supersymmetry, dark matter, new gauge bosons, gravitons, extra dimensions, etc.
  - Study kinematic variable distributions searching for deviations from SM predictions.
  - If no deviations: put limits on the new physics scenario you study.





Final states

- **Final state:** The detector signature of a certain process; i.e. what you observe in the detector.
- Final state objects: leptons ($e^{\pm}$, $\mu^{\pm}$), photons, jets (and missing E_T).
- Many different processes have the same final states $\Rightarrow$ background.
- When searching for new physics: look at events with the final state characteristic of the new physics signal, and try to eliminate the background.

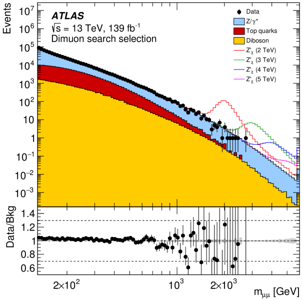


$2\ell + \not{E}_T$ $\quad$ $2\ell + 2j + \not{E}_T$ $\quad$ $3\ell + \not{E}_T$

Real and simulated data

Real data

- Real data from *pp* collision in ATLAS.
- Run II of LHC:
  - Collisions at $\sqrt{s} = 13$ TeV.
  - Collected 140 fb^{-1} of data ($\sim 10^{16}$ *pp* collisions)



Simulated data

- Monte Carlo simulations
  - Standard Model
  - New physics scenarios
- Generating events and simulating detector response.
- Used for comparisons with real data.

The big question: Does the real data match the Standard Model predictions?

ATLAS Open Data

2016:

- Release of 1 fb^{-1} of ATLAS data at 8 TeV (Run I).
- Both data and simulated background, and some signal samples.
- To be used for educational purposes and outreach.
- Mainly suited for lepton (and jet) analyses.
- The dataset can be found at the ATLAS Open Data Portal.

2020:

- Release of 10fb^{-1} Open Data at 13 TeV on February 11. Will use this data for the final project!

Structure of Open Data dataset

Note: This is presenting the old release, but new release is very similar - just bigger and better!

Real data:

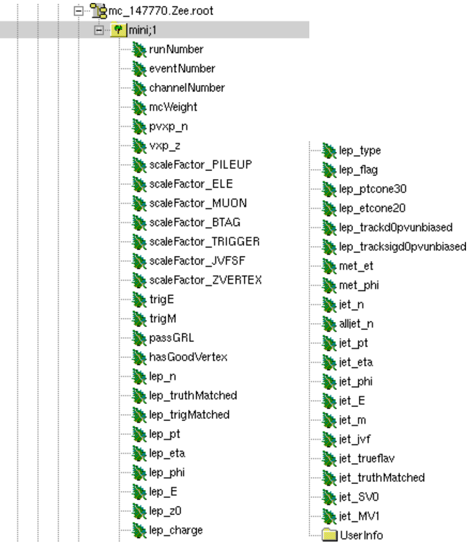
- $\sim$ 14 million events.
- One file with muon events and one with electron events.

Monte Carlo:

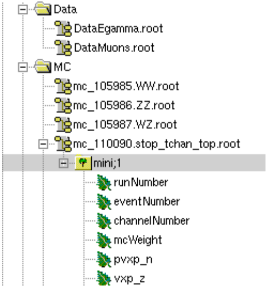
- $\sim$ 44 million events.
- 31 Standard Model samples: Diboson, single top, $t\bar{t}$, Z+jets, W+jets, Higgs, Drell-Yan.
- 11 beyond SM samples: $Z' \rightarrow t\bar{t}$ (others are available, but not through the data portal).

Structure of Open Data dataset II

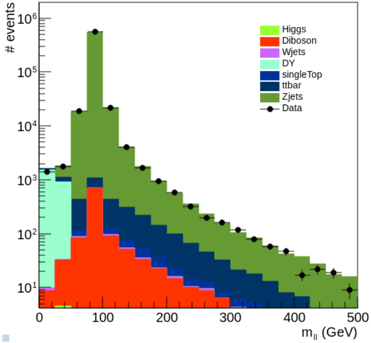
- Data stored in *nTuples*, or *trees*.
- One *event* = one *entry* in the tree.
- One variable = one *branch* in the tree.
- Branches can be vectors, integers, floats, booleans, etc.
- All events have the same branches.



How can we go from nTuples to colorful plots in a notebook?



??
$\Rightarrow$



Tools to be used for analysis in your projects

- ATLAS Open Data
  → see previous slides.
- ROOT
  → data analysis software developed at CERN.
- C++ (or Python if you strongly prefer that).
- Jupyter notebook
  → interactive environment where you can write pieces of code and text.

Jupyter notebook (with ROOT and C++) is available in VirtualBox running CERN CentOS 7. Installation instructions can be found here. Once inside the virtual machine you can start Jupyter notebook by typing *jupyter notebook* in the terminal.

Possible step-by-step approach

1. Read your data files.
2. Define variables and relate your variable to the branches in your tree.
3. Define the histograms you want to make.
4. Loop over all the events, do an event selection and fill the histograms you want to make.
5. Scale the backgrounds to cross section and luminosity.
6. Make a *stack* of the MC background histograms.
7. Plot data and MC together.

A simple example is outlined in the following slides. However, this example only includes real data, and **not** MC. (Treating the MC is a bit more complicated than real data $\Rightarrow$ see other examples here.)

Step 1: Read the data files

- Using a **TChain** you can link together data from several nTuples.
- Typically you want to make one chain for data and one for MC.

In [1]:
```
TChain *data = new TChain("mini"); // make a TChain
```

In [2]:
```
data->Add("http://opendata.atlas.cern/release/samples/Data/DataEgamma.root"); // Add data samples to the TChain
data->Add("http://opendata.atlas.cern/release/samples/Data/DataMuons.root");
```

Step 2: Define variables

- Define the variables you need.
- Link the variables to branches in the TTree.

In [3]:
```
Int_t lep_n, lep_charge[2], lep_type[2];
Float_t lep_pt[2], lep_E[2], lep_phi[2], lep_eta[2];
Bool_t passGRL, hasGoodVertex;
Float_t lep_etcone20[2], lep_ptcone30[2];
Int_t lep_flag[2];
```

In [4]:
```
data->SetBranchAddress("lep_n",      &lep_n);
data->SetBranchAddress("lep_charge", &lep_charge);
data->SetBranchAddress("lep_type",   &lep_type);
data->SetBranchAddress("lep_pt",     &lep_pt);
data->SetBranchAddress("lep_eta",    &lep_eta);
data->SetBranchAddress("lep_phi",    &lep_phi);
data->SetBranchAddress("lep_E",      &lep_E);
data->SetBranchAddress("passGRL", &passGRL);
data->SetBranchAddress("hasGoodVertex", &hasGoodVertex);
data->SetBranchAddress("lep_flag", &lep_flag);
data->SetBranchAddress("lep_ptcone30", &lep_ptcone30);
data->SetBranchAddress("lep_etcone20", &lep_etcone20);
```

Step 3: Make histograms (and other stuff you need...)

- Make the histograms you want to look at. E.g. $m_{\ell\ell}$, p_T, $\not{E}_T$ etc.
- Also define other thing you want to use. E.g. **TLorentzVector**'s, which are very practical for handling kinematics.

```
In [7]: TLorentzVector dilepton;
        TLorentzVector l1, l2;
```

```
In [6]: TH1F *hist_m = new TH1F("hist_m", "Invariant mass", 20, 0, 500);
```

Step 4: Loop over all events

- Make a loop that loop through all events.
- Do some **data quality cuts** to ensure high quality data.
- Do your **event selection** and **fill histograms**.
- Most time consuming part of the analysis.
- See code on next slide...

In [18]:

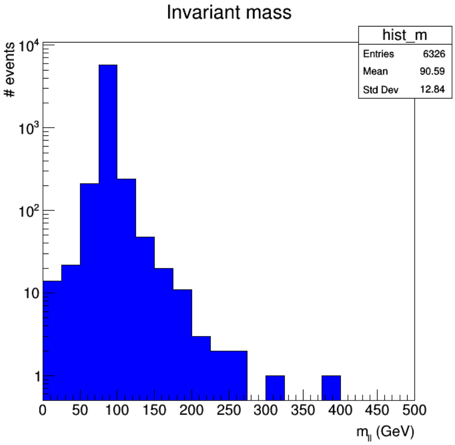
```
cout << "Looping over " << data->GetEntries() << " events...." << endl;
for(int i = 0; i < data->GetEntries(); i++){

    if( i%1000000 == 0 && i>0){ cout << i/1000000 << " million events processed" << endl;}
    if(!(i%100 == 0)){ continue; }  // Only keep 1 in 1000 events (for testing purposes)

    data->GetEntry(i);

    // Data quality cuts:

    if(passGRL == 0){ continue; }
    if(hasGoodVertex == 0){ continue; }
    //if(trigM == 0 && trigE == 0){ continue; }

    // Require "good leptons":

    if( lep_pt[0]/1000.0 < 25 ){ continue; }
    if( lep_etcone20[0]/lep_pt[0] > 0.15 ){ continue; }
    if( lep_ptcone30[0]/lep_pt[0] > 0.15 ){ continue; }
    if( !(lep_flag[0] & 512) ){ continue; }

    if( lep_pt[1]/1000.0 < 25 ){ continue; }
    if( lep_etcone20[1]/lep_pt[1] > 0.15 ){ continue; }
    if( lep_ptcone30[1]/lep_pt[1] > 0.15 ){ continue; }
    if( !(lep_flag[1] & 512) ){ continue; }

    // Event selection:

    // Cut #1: Require (exactly) 2 leptons
    if(lep_n != 2){ continue; }
    // Cut #2: Require opposite charge
    if(lep_charge[0] == lep_charge[1]){ continue; }
    // Cut #3: Require same flavour (2 electrons or 2 muons)
    if(lep_type[0] != lep_type[1]){ continue; }

    // Set Lorentz vectors:
    l1.SetPtEtaPhiE(lep_pt[0]/1000., lep_eta[0], lep_phi[0], lep_E[0]/1000.);
    l2.SetPtEtaPhiE(lep_pt[1]/1000., lep_eta[1], lep_phi[1], lep_E[1]/1000.);
    // Variables are stored in the TTree with unit MeV, so we need to divide by 1000
    // to get GeV, which is a more practical and commonly used unit.

    dilepton = l1 + l2;

    hist_m->Fill(dilepton.M());
```

Step 5: Make lovely plots

In [33]:
```
hist_m->SetFillColor(kBlue);
hist_m->GetYaxis()->SetTitle("# events");
hist_m->GetYaxis()->SetTitleOffset(1.3);
hist_m->GetXaxis()->SetTitle("m_{ll} (GeV)");
hist_m->GetXaxis()->SetTitleOffset(1.3);
```

In [ ]:
```
TCanvas *c = new TCanvas("c", "c", 10, 10, 700, 700);
```

In [34]:
```
hist_m->Draw();
c->Draw();
```



Taking it a bit further...

- Include MC background samples. See this notebook.
  → Many samples ⇒ more "bookkeeping".
  → Need to be scaled to cross section and luminosity, and weighted correctly.
  → You need to do this on Project 1.
- Include MC signal samples.
- Statistical analysis of results. (will have a session about that later...)

Relevant material

- Virtual box installation
- Notebook examples.
- This presentation
- Get all the material from GitHub by doing
  `git clone https://github.com/Etienne357/FYS5555.git`
- ATLAS Open Data Portal
- ATLAS note about Open Data (all the details you need about the dataset).