Plotting Histograms Using `plot_all.py`

Ibrahim H.I. Abushawish

May 8, 2025

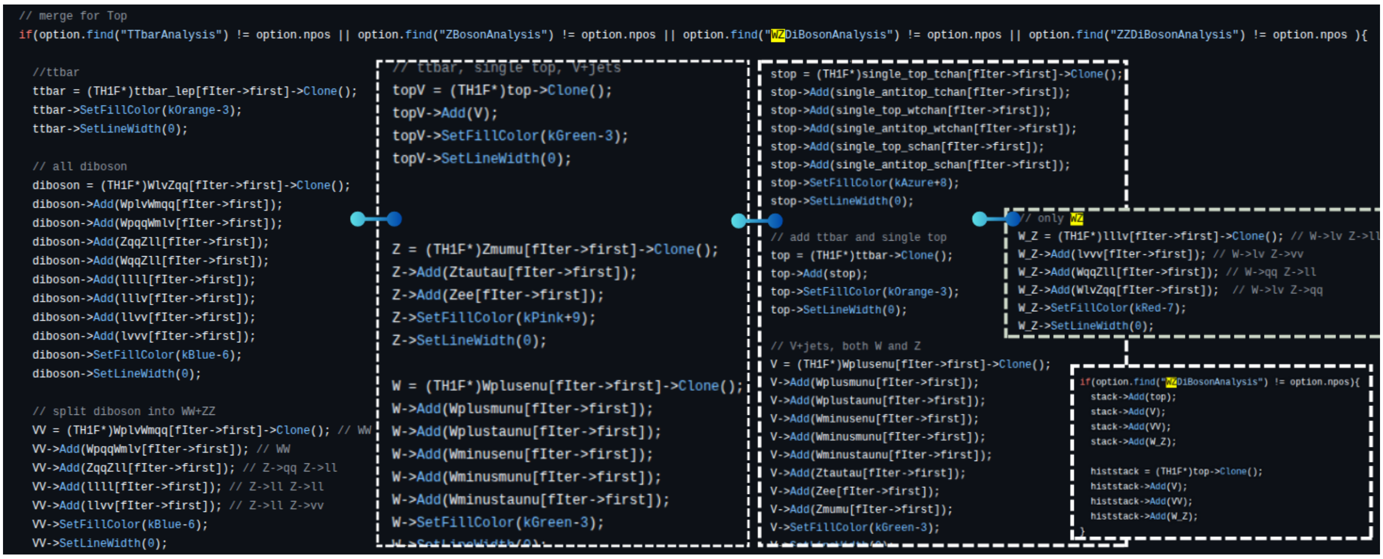
Plotting Histograms Using `plot_all.py`

The process of plotting histograms over each other using the provided `plot_all.py` script involves the following steps:

The files to combine are determined based on the reference tutorial for the analysis, as outlined in the following links:

- `https://github.com/atlas-outreach-data-tools/atlas-outreach-cpp-framework-13tev/blob/688d73fb73a742a4e43912189411dcae4dec85e2/Plotting/Plotting.cxx`
- `https://github.com/atlas-outreach-data-tools/atlas-outreach-cpp-framework-13tev/blob/master/Plotting/Plotting.cxx`

These files should be combined with proper naming conventions to match the expected input of the `plot_all.py` script. Ensure that the combined files are named appropriately (e.g., `data.root`, `V.root`, `VV.root`, etc.) to align with the script's requirements.



```
// merge for Top
if(option.find("TTbarAnalysis") != option.npos || option.find("ZBosonAnalysis") != option.npos || option.find("WZDiBosonAnalysis") != option.npos || option.find("ZZDiBosonAnalysis") != option.npos ){

  //ttbar
  ttbar = (TH1F*)ttbar_lep[fIter->first]->Clone();
  ttbar->SetFillColor(kOrange-3);
  ttbar->SetLineWidth(0);

  // all diboson
  diboson = (TH1F*)WlvZqq[fIter->first]->Clone();
  diboson->Add(WplvWmqq[fIter->first]);
  diboson->Add(WpqqWmlv[fIter->first]);
  diboson->Add(ZqqZll[fIter->first]);
  diboson->Add(WqqZll[fIter->first]);
  diboson->Add(llll[fIter->first]);
  diboson->Add(lllv[fIter->first]);
  diboson->Add(llvv[fIter->first]);
  diboson->Add(lvvv[fIter->first]);
  diboson->SetFillColor(kBlue-6);
  diboson->SetLineWidth(0);

  // split diboson into WW+ZZ
  VV = (TH1F*)WplvWmqq[fIter->first]->Clone(); // WW
  VV->Add(WpqqWmlv[fIter->first]); // WW
  VV->Add(ZqqZll[fIter->first]); // Z->qq Z->ll
  VV->Add(llll[fIter->first]); // Z->ll Z->ll
  VV->Add(llvv[fIter->first]); // Z->ll Z->vv
  VV->SetFillColor(kBlue-6);
  VV->SetLineWidth(0);
```

```
// ttbar, single top, V+jets
topV = (TH1F*)top->Clone();
topV->Add(V);
topV->SetFillColor(kGreen-3);
topV->SetLineWidth(0);

Z = (TH1F*)Zmumu[fIter->first]->Clone();
Z->Add(Ztautau[fIter->first]);
Z->Add(Zee[fIter->first]);
Z->SetFillColor(kPink+9);
Z->SetLineWidth(0);

W = (TH1F*)Wplusenu[fIter->first]->Clone();
W->Add(Wplusmunu[fIter->first]);
W->Add(Wplustaunu[fIter->first]);
W->Add(Wminusenu[fIter->first]);
W->Add(Wminusmunu[fIter->first]);
W->Add(Wminustaunu[fIter->first]);
W->SetFillColor(kGreen-3);
```

```
stop = (TH1F*)single_top_tchan[fIter->first]->Clone();
stop->Add(single_antitop_tchan[fIter->first]);
stop->Add(single_top_wtchan[fIter->first]);
stop->Add(single_antitop_wtchan[fIter->first]);
stop->Add(single_top_schan[fIter->first]);
stop->Add(single_antitop_schan[fIter->first]);
stop->SetFillColor(kAzure+8);
stop->SetLineWidth(0);

// add ttbar and single top
top = (TH1F*)ttbar->Clone();
top->Add(stop);
top->SetFillColor(kOrange-3);
top->SetLineWidth(0);

// V+jets, both W and Z
V = (TH1F*)Wplusenu[fIter->first]->Clone();
V->Add(Wplusmunu[fIter->first]);
V->Add(Wplustaunu[fIter->first]);
V->Add(Wminusenu[fIter->first]);
V->Add(Wminusmunu[fIter->first]);
V->Add(Wminustaunu[fIter->first]);
V->Add(Ztautau[fIter->first]);
V->Add(Zee[fIter->first]);
V->Add(Zmumu[fIter->first]);
V->SetFillColor(kGreen-3);
```

```
// only WZ
W_Z = (TH1F*)lllv[fIter->first]->Clone(); // W->lv Z->ll
W_Z->Add(lvvv[fIter->first]); // W->lv Z->vv
W_Z->Add(WqqZll[fIter->first]); // W->qq Z->ll
W_Z->Add(WlvZqq[fIter->first]);  // W->lv Z->qq
W_Z->SetFillColor(kRed-7);
W_Z->SetLineWidth(0);
```

```
if(option.find("WZDiBosonAnalysis") != option.npos){
  stack->Add(top);
  stack->Add(V);
  stack->Add(VV);
  stack->Add(W_Z);

  histstack = (TH1F*)top->Clone();
  histstack->Add(V);
  histstack->Add(VV);
  histstack->Add(W_Z);
}
```

Figure 1: Example code screen shots of the `Plotting.cxx` script. Those lines should give you hints about the data and simulations you need to combine. This is for the WZ analysis, but similarly, you can find the same lines for other analysis.

1. Combining `.root` Files

- Use the `hadd` command to combine `.root` files generated by the ADL (Analysis Description Language) analysis. These files should be named appropriately (e.g., `data.root`, `V.root`, `VV.root`, etc.) to match the expected input for the `plot_all.py` script.
- Example commands for combining files:

```
hadd data.root file1.root file2.root file3.root
```

```
hadd data.root file*.root
```

As a reference, you can find example commands in the `hadd.sh` script under the `https://github.com/ibeuler/CutLang-exercise-analysis` repository.

2. Setting Up the `plot_all.py` Script

- The script is designed to handle different analyses (e.g., `WZ`, `Zboson`, `HZZ`, etc.) based on the `analysis_name` parameter passed as a command-line argument.
- The script reads the combined `.root` files and extracts histograms for specific regions and variables.

3. Defining Variables to Plot

- At the bottom of the script, specific variables to be plotted are defined for each analysis. For example, for the `WZ` analysis, the following variables are defined:

```
#plotting("WZDiBoson","hist_etmiss",350)
#plotting("WZDiBoson","hist_threelepteta",600)
#plotting("WZDiBoson","hist_threeleptphi",400)
#plotting("WZDiBoson","hist_threeleptpt",1900)
#plotting("WZDiBoson","hist_threeleptE",1100)
#plotting("WZDiBoson","hist_mtw",400)
#plotting("WZDiBoson","hist_mLL",400)
plotting("WZDiBoson","hist_ptLL",400)
```

- Uncomment the desired line(s) to plot specific variables.

4. Running the Script

- Execute the script with the appropriate analysis name using the `-a` or `--analysis` option. For example:

```
python plot_all.py -a WZ
```

5. Plotting Histograms

- The script reads the histograms from the `.root` files, applies any necessary scaling, and stacks them for visualization.
- It uses ROOT's `THStack` to overlay histograms for different components (e.g., data, signal, background) and displays them in a single plot.

6. Customizing the Output

- The script includes options for setting axis labels, titles, and ranges. It also calculates ratios (e.g., Data/MC) and adds legends for clarity.
- The final plots are displayed and can be saved as images or PDFs.

7. Ensuring Proper Histogram and Region Naming in the ADL File

- The `plot_all.py` script expects histograms and regions to follow specific naming conventions. Ensure that both histograms and regions are named correctly in the ADL file to match the script's requirements.
- For example, histograms for the `WZ` analysis should be named as follows:

```
histogram hist_etmiss { ... }
histogram hist_threelepteta { ... }
histogram hist_threeleptphi { ... }
histogram hist_threeleptpt { ... }
histogram hist_threeleptE { ... }
histogram hist_mtw { ... }
histogram hist_mLL { ... }
histogram hist_ptLL { ... }
```

- Similarly, regions should also follow specific naming conventions.
- Verify that the histogram and region names in the ADL file match the names used in the `plotting()` function calls and other parts of the `plot_all.py` script.
- Consistent naming ensures that the script can correctly locate and process the histograms and regions for plotting.

By following this process, you can visualize the results of your analysis for the desired variables and compare different components in a single plot.