News about FCC physics analyses

Clement Helsens, Valentin Volkl

CERN-EP

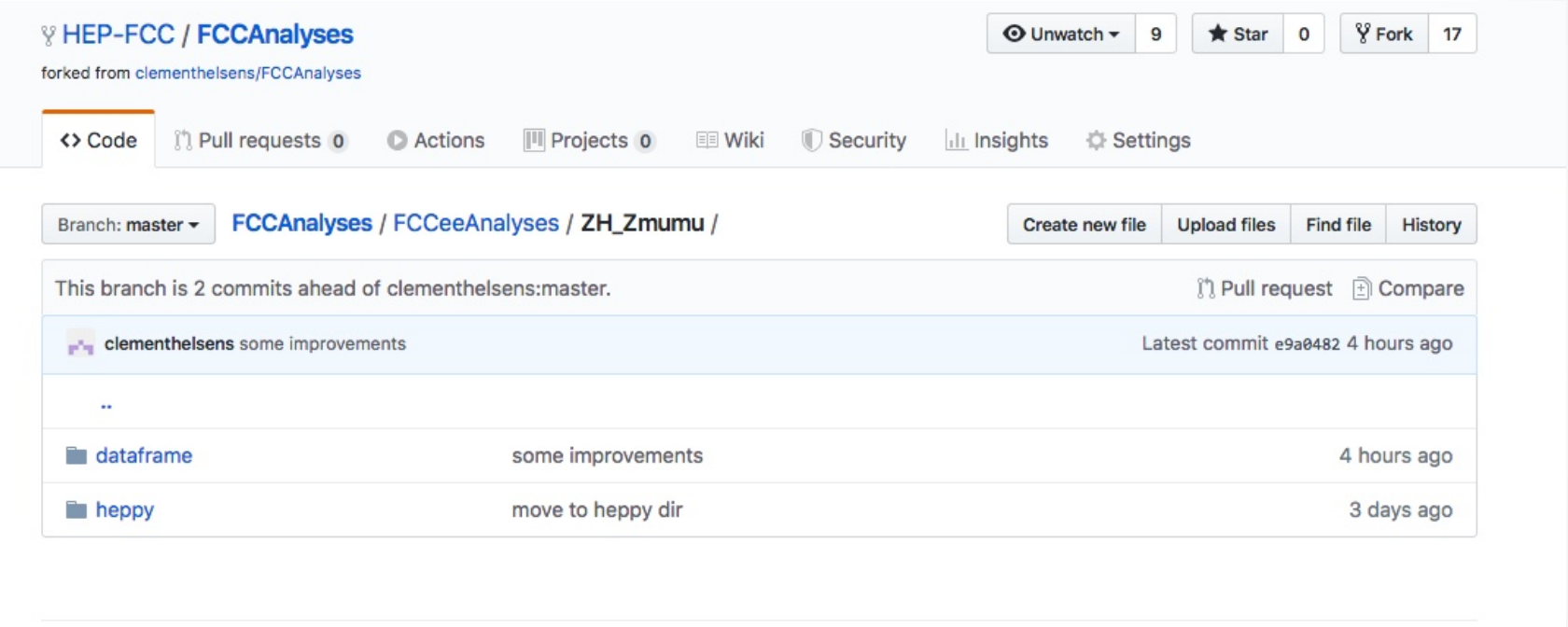
Introduction

- As we are not in a hurry right now, decided to change the way analyses are done
- Get away from heppy and use Root DataFrame instead
- Merge the FCCAnalyses with the FlatTreeAnalyser

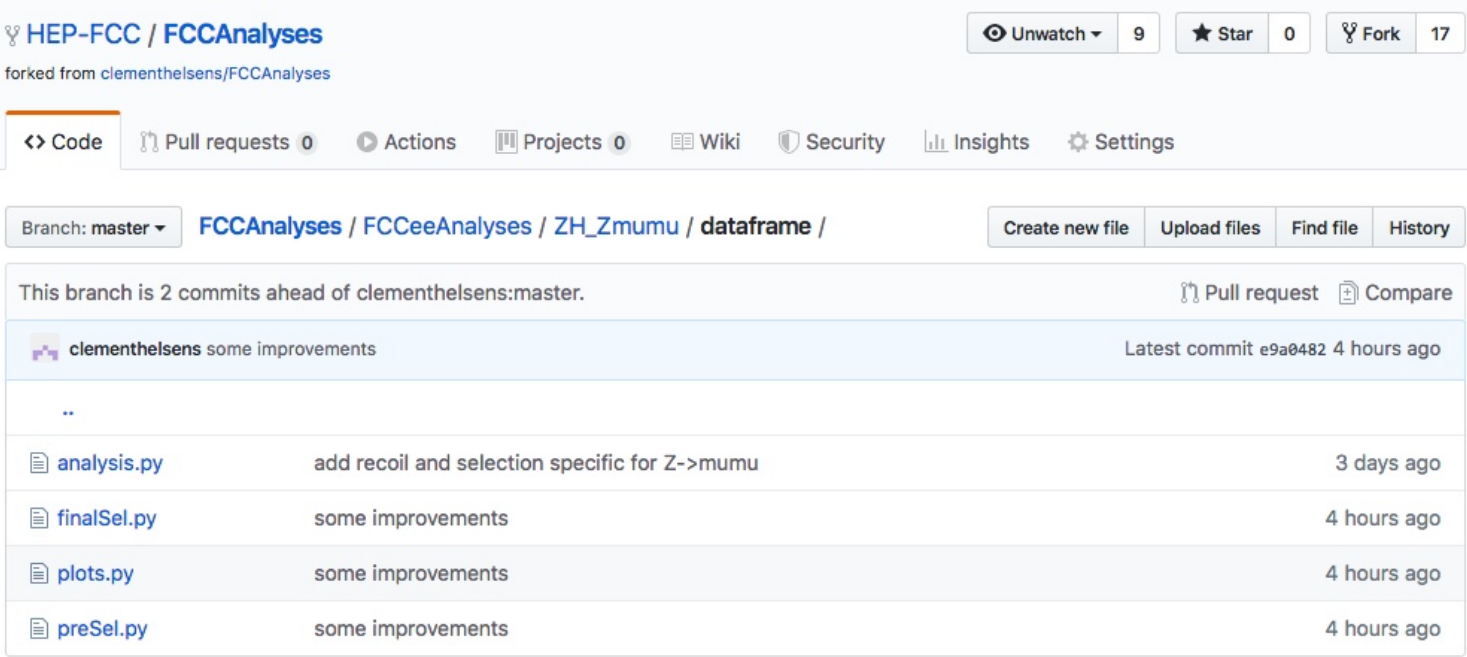
How it works

- There is a “FCCAnalysis” C++ library that is used by the python code
- This is where what we called heppy modules are defined
- There is also a possibility to define user specific functions in the analysis.py using the glinterpreter

How it look like now



How it look like now



Generalities

Each analysis is hosted in a single directory, for example `FCCeeAnalyses/ZH_Zmumu/dataframe/` and contains the same kind of files, please use the same naming convention for all analysis.

1. `analysis.py` : This class that is used to define the list of analysers and filters to run on as well as the output variables.
2. `preSel.py` : This configuration file is used to define how to run the `analysis.py` . It contains the list of samples, the number of CPUs, the fraction of the original sample to process and the base directory for the yaml files (that contains the informations about the samples). This will run the `analysis.py` with a common code `bin/runDataFrame.py` (this last file is common to all analyses and should not be touched).
3. `finalSel.py` : This configuration file contains the final selections and it runs over the locally produced flat ntuples from the `preSel.py` . It contains a link to the `procDict.json` for getting cross section etc...(this might be removed later to include everything in the yaml, closer to the sample), the list of processes, the number of CPU, the cut list, and the variables (that will be both written in a `TTree` and in the form of `TH1` properly normalised to an integrated luminosity of 1pb-1.
4. `plots.py` : This configuration files is used to select the final selections from running `finalSel.py` to plot. Informations about how to merge processes, write some extra text, normalise to a given integrated luminosity etc... For the moment it is possible to only plot one signal at the time, but several backgrounds.

Delphes FCCee Physic events v0.1

Search for names..

NO	NAME	NEVENTS	NWEIGHTS	NFILES	NBAD	NEOS	SIZE (GB)	OUTPUT PATH	MAIN PROCESS	FINAL STATES	CROSS SECTION (PB)	K-FACTOR	MATCHING EFF
1	p8_ee_ZH_ecm240	29,850,000	0	2985	0	2985	125.57	/eos/experiment /fcc/ee/generation /DelphesEvents /fcc_v01 /p8_ee_ZH_ecm240/	ZH ecm=240GeV	inclusive decays	0.201037	1.0	1.0
2	p8_ee_ZZ_ecm240	29,880,000	0	2988	0	2988	96.24	/eos/experiment /fcc/ee/generation /DelphesEvents /fcc_v01 /p8_ee_ZZ_ecm240/	ZZ ecm=240GeV	inclusive decays	1.35899	1.0	1.0
3	p8_ee_WW_ecm240	29,630,000	0	2963	0	2963	91.49	/eos/experiment /fcc/ee/generation /DelphesEvents /fcc_v01 /p8_ee_WW_ecm240/	WW ecm=240GeV	inclusive decays	16.4385	1.0	1.0
4	p8_ee_Zqq_ecm240	19,970,000	0	1997	0	1997	51.75	/eos/experiment /fcc/ee/generation /DelphesEvents /fcc_v01 /p8_ee_Zqq_ecm240/	Z/Gamma* ecm=240GeV	hadronic decays	53.188	1.0	1.0
5	mgp8_ee_tt_ecm350	100,000	0	10	0	10	0.54	/eos/experiment /fcc/ee/generation /DelphesEvents /fcc_v01 /mgp8_ee_tt_ecm350/	tt	tt inclusive	1.0	1.0	1.0
6	total	109,430,000	0.0	10,943	0.0	10,943.0	365.59						

Pre-selection

The pre-selection runs over already existing and properly registered FCCSW EDM events. The dataset names with the corresponding statistics can be found here. One important parameter is the fraction of the total dataset to run. It can be found in the `preSel.py` file by setting a value between]0,1]. For example `fraction=0.1` will run over 10% of the statistics. Reading the files on `eos`, and with 15 CPUs we observe processing speeds between 3000 and 10000 events per seconds depending on the number of files. Only run full statistics after having done all the proper testing and analysis design as it can take some time (that of course depends on the sample total statistics). To run the pre-selection of the `ZH_Zmumu` analysis, just run:

```
python FCCeeAnalyses/ZH_Zmumu/dataframe/preSel.py
```

```
3   basedir="/afs/cern.ch/work/h/helsens/public/FCCDicts/yaml/FCCee/fcc_v01/"
4   outdir="FCCee/ZH_Zmumu/"
5   NUM_CPUS = 15
6   process_list=['p8_ee_ZZ_ecm240','p8_ee_WW_ecm240','p8_ee_ZH_ecm240']
7   fraction=0.01
8
9   import bin.runDataFrame as rdf
10  myana=rdf.runDataFrame(basedir,process_list)
11  myana.run(ncpu=NUM_CPUS,fraction=fraction,outDir=outdir)
```

This automatically runs the analysis.py

analysis.py

```python
import sys
import ROOT

print "Load cxx analyzers ... ",
ROOT.gSystem.Load("libdatamodel")
ROOT.gSystem.Load("libFCCAnalyses")
ROOT.gErrorIgnoreLevel = ROOT.kFatal

_p = ROOT.fcc.ParticleData()
_s = ROOT.selectParticlesPtIso

class analysis():

    #__________________________________________
    def __init__(self, inputlist, outname, ncpu):
        self.outname = outname
        if ".root" not in outname:
            self.outname+=".root"

        ROOT.ROOT.EnableImplicitMT(ncpu)

        self.df = ROOT.RDataFrame("events", inputlist)
        print " done"
```

```python
    #__________________________________________
    def run(self):
        df2 = self.df.Define("selected_muons", "selectParticlesPtIso(10, 0.4)(muons, muonITags)") \
                     .Define("selected_muons_pt", "get_pt(selected_muons)") \
                     .Define("selected_muons_y", "get_y(selected_muons)") \
                     .Define("selected_muons_p", "get_p(selected_muons)") \
                     .Define("selected_muons_e", "get_e(selected_muons)") \
                     .Define("jets_10_bs", "selectJets(10, true)(efjets, efbTags)") \
                     .Define("jets_10_lights", "selectJets(10, false)(efjets, efbTags)") \
                     .Define("selected_bs", "noMatchJets(0.2)(jets_10_bs, selected_muons)") \
                     .Define("selected_lights", "noMatchJets(0.2)(jets_10_lights, selected_muons)") \
                     .Define("nbjets", "get_njets(selected_bs)") \
                     .Define("njets", "get_njets2(selected_bs, selected_lights)") \
                     .Define("weight"," id_float(mcEventWeights)") \
                     .Define("zed_leptonic","ResonanceBuilder(23, 91)(selected_muons)") \
                     .Define("zed_leptonic_m", "get_mass(zed_leptonic)") \
                     .Define("zed_leptonic_pt","get_pt(zed_leptonic)") \
                     .Define("zed_hadronic_light","JetResonanceBuilder(23, 91)(jets_10_lights)") \
                     .Define("zed_hadronic_light_m", "get_mass(zed_hadronic_light)") \
                     .Define("zed_hadronic_light_pt","get_pt(zed_hadronic_light)") \
                     .Define("zed_hadronic_b","JetResonanceBuilder(23, 91)(jets_10_bs)") \
                     .Define("zed_hadronic_b_m", "get_mass(zed_hadronic_b)") \
                     .Define("zed_hadronic_b_pt","get_pt(zed_hadronic_b)") \
                     .Define("zed_leptonic_recoil","recoil(240)(zed_leptonic)") \
                     .Define("zed_leptonic_recoil_m","get_mass(zed_leptonic_recoil)") \

```

Call the functions to define the collections

How functions looks like?

```
141  recoil::recoil(float arg_sqrts) : m_sqrts(arg_sqrts) {};
142  ROOT::VecOps::RVec<fcc::ParticleData>  recoil::operator() (ROOT::VecOps::RVec<fcc::ParticleData> in) {
143    ROOT::VecOps::RVec<fcc::ParticleData> result;
144    auto recoil_p4 = TLorentzVector(0, 0, 0, m_sqrts);
145    for (auto & v1: in) {
146      TLorentzVector tv1;
147      tv1.SetXYZM(v1.core.p4.px, v1.core.p4.py, v1.core.p4.pz, v1.core.p4.mass);
148      recoil_p4 -= tv1;
149    }
150    auto recoil_fcc = fcc::ParticleData();
151    recoil_fcc.core.p4.px = recoil_p4.Px();
152    recoil_fcc.core.p4.py = recoil_p4.Py();
153    recoil_fcc.core.p4.pz = recoil_p4.Pz();
154    recoil_fcc.core.p4.mass = recoil_p4.M();
155    result.push_back(recoil_fcc);
156    return result;
157  };
```

In the FCCAnalysis.cc

In the FCCAnalysis.h

```
54  struct recoil {
55    recoil(float arg_sqrts);
56    float m_sqrts = 240.0;
57    ROOT::VecOps::RVec<fcc::ParticleData> operator() (ROOT::VecOps::RVec<fcc::ParticleData> in) ;
58  };
59
```

analysis.py

```
1  import sys
2  import ROOT
3
4  print "Load cxx analyzers ... ",
5  ROOT.gSystem.Load("libdatamodel")
6  ROOT.gSystem.Load("libFCCAnalyses")
7  ROOT.gErrorIgnoreLevel = ROOT.kFatal
8
9  _p = ROOT.fcc.ParticleData()
10 _s = ROOT.selectParticlesPtIso
11
12 class analysis():
13
14     #__________________________________________________
15     def __init__(self, inputlist, outname, ncpu):
16         self.outname = outname
17         if ".root" not in outname:
18             self.outname+=".root"
19
20         ROOT.ROOT.EnableImplicitMT(ncpu)
21
22         self.df = ROOT.RDataFrame("events", inputlist)
23         print " done"
```

What will be stored in the outputs

```
52
53         branchList = ROOT.vector('string')()
54         for branchName in [
55                 "selected_muons_pt",
56                 "selected_muons_y",
57                 "selected_muons_p",
58                 "selected_muons_e",
59                 "zed_leptonic_pt",
60                 "zed_leptonic_m",
61                 "zed_hadronic_light_pt",
62                 "zed_hadronic_light_m",
63                 "zed_hadronic_b_pt",
64                 "zed_hadronic_b_m",
65                 "zed_leptonic_recoil_m",
66                 "nbjets",
67                 "njets",
68                 "weight",
69         ]:
70             branchList.push_back(branchName)
71             df2.Snapshot("events", self.outname, branchList)
```

Pre-selection

The pre-selection runs over already existing and properly registered FCCSW EDM events. The dataset names with the corresponding statistics can be found here. One important parameter is the fraction of the total dataset to run. It can be found in the `preSel.py` file by setting a value between]0,1]. For example `fraction=0.1` will run over 10% of the statistics. Reading the files on `eos`, and with 15 CPUs we observe processing speeds between 3000 and 10000 events per seconds depending on the number of files. Only run full statistics after having done all the proper testing and analysis design as it can take some time (that of course depends on the sample total statistics). To run the pre-selection of the `ZH_Zmumu` analysis, just run:

```
python FCCeeAnalyses/ZH_Zmumu/dataframe/preSel.py
```

This will output 3 files in `Outputs/FCCee/ZH_Zmumu/` following the parameter `outdir` in the `preSel.py` configuration file.

```
248M Mar  9 12:45 p8_ee_ZZ_ecm240.root
229M Mar  9 13:00 p8_ee_WW_ecm240.root
253M Mar  9 13:11 p8_ee_ZH_ecm240.root
```

Final selection

The final selection runs on the pre-selection files that we produced in the Pre-selection step. In the configuration file `finalSel.py` we define the various cuts to run on and the final variables to be stored in both a `TTree` and histograms. This is why the variables needs extra fields like `title`, number of bins and range for the histogram creation. In this example it should run like:

```
python FCCeeAnalyses/ZH_Zmumu/dataframe/finalSel.py
```

```
import ROOT

###Input directory where the files produced at the pre-selection level are
baseDir  = "/afs/cern.ch/user/h/helsens/FCCsoft/FCCAnalyses/Outputs/FCCee/ZH_Zmumu/"

###Link to the dictonary that contains all the cross section informations etc...
procDict = "/afs/cern.ch/work/h/helsens/public/FCCDicts/FCCee_procDict_fcc_v01.json"

process_list=['p8_ee_ZZ_ecm240','p8_ee_WW_ecm240','p8_ee_ZH_ecm240']

###Dictionnay of the list of cuts. The key is the name of the selection that will be added to the output file
cut_list = {"sel0":"zed_leptonic_m.size() == 1",
            "sel1":"zed_leptonic_m.size() == 1 && zed_leptonic_m[0] > 80 &&  zed_leptonic_m[0] < 100",
            "sel2":"zed_leptonic_m.size() == 1 && zed_leptonic_m[0] > 80 &&  zed_leptonic_m[0] < 100 && nbjets==2"
            }


###Dictionary for the ouput variable/hitograms. The key is the name of the variable in the output files. "name" is the name of the variab
variables = {
    "mz":{"name":"zed_leptonic_m","title":"m_{Z} [GeV]","bin":125,"xmin":0,"xmax":250},
    "mz_zoom":{"name":"zed_leptonic_m","title":"m_{Z} [GeV]","bin":40,"xmin":80,"xmax":100},
    "nbjets":{"name":"nbjets","title":"number of bjets","bin":10,"xmin":0,"xmax":10},
    "leptonic_recoil_m":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","bin":100,"xmin":0,"xmax":200},
}

###Number of CPUs to use
NUM_CPUS = 10

###This part is standard to all analyses
import bin.runDataFrameFinal as rdf
myana=rdf.runDataFrameFinal(baseDir,procDict,process_list,cut_list,variables)
myana.run(ncpu=NUM_CPUS)
```

Final selection

The final selection runs on the pre-selection files that we produced in the Pre-selection step. In the configuration file `finalSel.py` we define the various cuts to run on and the final variables to be stored in both a `TTree` and histograms. This is why the variables needs extra fields like `title`, number of bins and range for the histogram creation. In this example it should run like:

```
python FCCeeAnalyses/ZH_Zmumu/dataframe/finalSel.py
```

This will create 2 files per selection `SAMPLENAME_SELECTIONNAME.root` for the `TTree` and `SAMPLENAME_SELECTIONNAME_histo.root` for the histograms. `SAMPLENAME` and `SELECTIONNAME` corresponds to the name of the sample and selection respectively in the configuration file.

```
 33M Mar 12 14:11 p8_ee_ZZ_ecm240_sel0.root
 14K Mar 12 14:11 p8_ee_ZZ_ecm240_sel0_histo.root
6.1M Mar 12 14:12 p8_ee_WW_ecm240_sel0.root
 12K Mar 12 14:12 p8_ee_WW_ecm240_sel0_histo.root
 17M Mar 12 14:12 p8_ee_ZH_ecm240_sel0.root
 13K Mar 12 14:12 p8_ee_ZH_ecm240_sel0_histo.root
```

Plotting

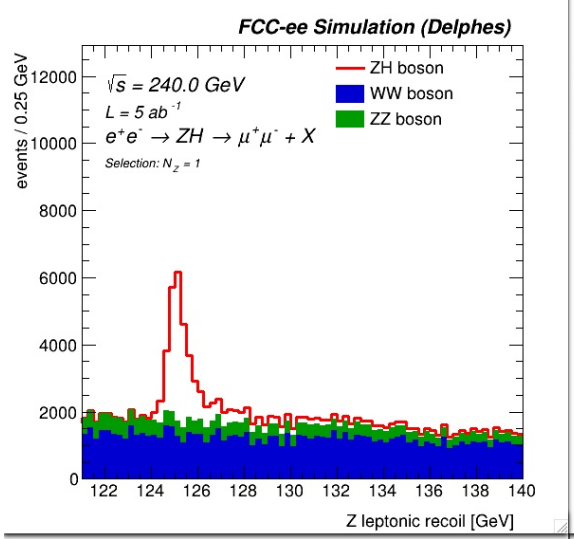
The plotting configuration file `plots.py` contains informations about plotting details for plots rendering but also ways of combining samples for plotting. In this example just run like:

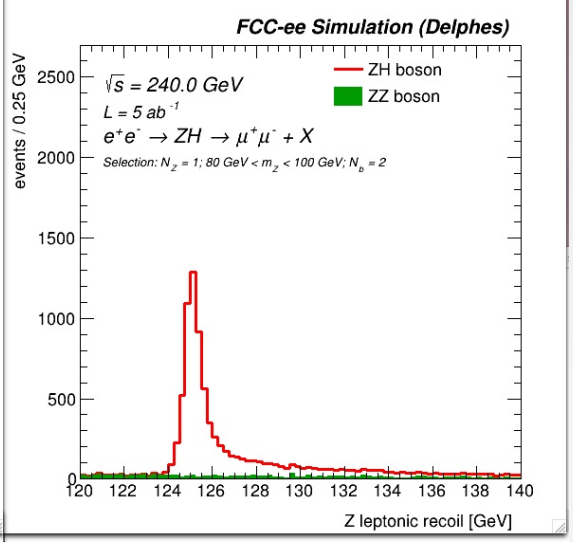
```
python bin/doPlots.py FCCeeAnalyses/ZH_Zmumu/dataframe/plots.py
```

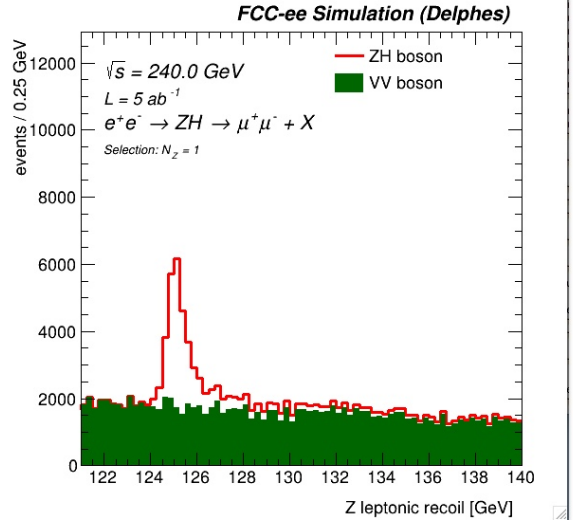
This will produce the plots in the `outdir` defined in the configuration file.

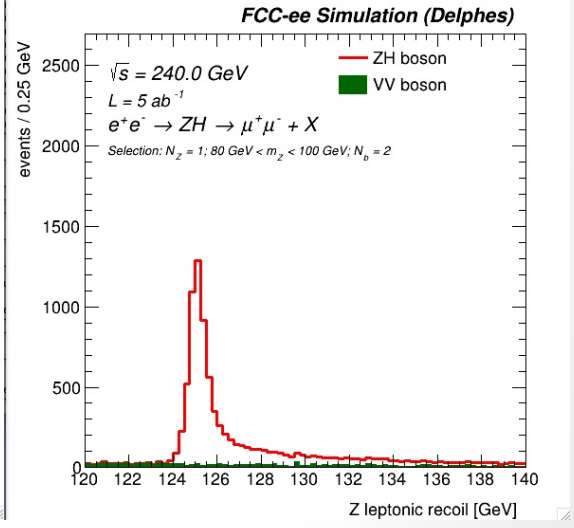
```
import ROOT

# global parameters
intLumi        = 5.0e+06 #in pb-1
ana_tex        = "e^{+}e^{-} #rightarrow ZH #rightarrow #mu^{+}#mu^{-} + X"
delphesVersion = "3.4.2"
energy         = 240.0
collider       = "FCC-ee"
inputDir       = "Outputs/FCCee/ZH_Zmumu/"
formats        = ['png','pdf']
yaxis          = ['lin','log']
stacksig       = ['stack','nostack']
outdir         = 'Outputs/FCCee/ZH_Zmumu/plots/'

variables = ['mz','mz_zoom','nbjets','leptonic_recoil_m']

###Dictonnary with the analysis name as a key, and the list of selections to be pl
selections = {}
selections['ZH']   = ["sel0","sel1","sel2"]
selections['ZH_2'] = ["sel0","sel2"]

extralabel = {}
extralabel['sel0'] = "Selection: N_{Z} = 1"
extralabel['sel1'] = "Selection: N_{Z} = 1; 80 GeV < m_{Z} < 100 GeV"
extralabel['sel2'] = "Selection: N_{Z} = 1; 80 GeV < m_{Z} < 100 GeV; N_{b} = 2"

colors = {}
colors['ZH'] = ROOT.kRed
colors['WW'] = ROOT.kBlue+1
colors['ZZ'] = ROOT.kGreen+2
colors['VV'] = ROOT.kGreen+3

plots = {}
plots['ZH'] = {'signal':{'ZH':['p8_ee_ZH_ecm240']},
               'backgrounds':{'WW':['p8_ee_WW_ecm240'],
                              'ZZ':['p8_ee_ZZ_ecm240']}
           }


plots['ZH_2'] = {'signal':{'ZH':['p8_ee_ZH_ecm240']},
                 'backgrounds':{'VV':['p8_ee_WW_ecm240','p8_ee_ZZ_ecm240']}
             }

legend = {}
legend['ZH'] = 'ZH boson'
legend['WW'] = 'WW boson'
legend['ZZ'] = 'ZZ boson'
legend['VV'] = 'VV boson'
```

```
27
28  colors = {}
29  colors['ZH'] = ROOT.kRed
30  colors['WW'] = ROOT.kBlue+1
31  colors['ZZ'] = ROOT.kGreen+2
32  colors['VV'] = ROOT.kGreen+3
33
34  plots = {}
35  plots['ZH'] = {'signal':{'ZH':['p8_ee_ZH_ecm240']},
36                 'backgrounds':{'WW':['p8_ee_WW_ecm240'],
37                                'ZZ':['p8_ee_ZZ_ecm240']}
38              }
39
40
41  plots['ZH_2'] = {'signal':{'ZH':['p8_ee_ZH_ecm240']},
42                   'backgrounds':{'VV':['p8_ee_WW_ecm240','p8_ee_ZZ_ecm240']}
43                }
44
45  legend = {}
46  legend['ZH'] = 'ZH boson'
47  legend['WW'] = 'WW boson'
48  legend['ZZ'] = 'ZZ boson'
49  legend['VV'] = 'VV boson'
```


FCC-ee Simulation (Delphes)
ZH boson
WW boson
ZZ boson
√s = 240.0 GeV
L = 5 ab-1
e+e- → ZH → μ+μ- + X
Selection: NZ = 1
events / 0.25 GeV
Z leptonic recoil [GeV]


FCC-ee Simulation (Delphes)
ZH boson
ZZ boson
√s = 240.0 GeV
L = 5 ab-1
e+e- → ZH → μ+μ- + X
Selection: NZ = 1; 80 GeV < mZ < 100 GeV; Nb = 2
events / 0.25 GeV
Z leptonic recoil [GeV]


FCC-ee Simulation (Delphes)
ZH boson
VV boson
√s = 240.0 GeV
L = 5 ab-1
e+e- → ZH → μ+μ- + X
Selection: NZ = 1
events / 0.25 GeV
Z leptonic recoil [GeV]


FCC-ee Simulation (Delphes)
ZH boson
VV boson
√s = 240.0 GeV
L = 5 ab-1
e+e- → ZH → μ+μ- + X
Selection: NZ = 1; 80 GeV < mZ < 100 GeV; Nb = 2
events / 0.25 GeV
Z leptonic recoil [GeV]