FCCee Analysis Examples

Using the example `higgs/mH-recoil/mumu` from FCCAnalyses

In [1]:
```
using Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
 Activating project at `~/Development/EDM4hep.jl/examples/FCC`
```

In [2]:
```
using EDM4hep
using EDM4hep.RootIO
using EDM4hep.SystemOfUnits
using EDM4hep.Histograms
using EDM4hep.Analysis
```

Definition of some analysis functions

These are couple of examples of high-level functions that makes use of `ReconstructedParticle` objects to build resonances and recoils. They make use of standard Julia functions to generate combinations, to sort a vector, and to work with LorentzVectors.

In [3]:
```
# re-using convenient existing packages
using LorentzVectorHEP
using Combinatorics

"""
    resonanceBuilder(rmass::AbstractFloat, legs::AbstractVector{Reconstru

Returns a container with the best resonance of 2 by 2 combinatorics of th
sorted by closest to the input `rmass` in absolute value.
"""
function resonanceBuilder(rmass::AbstractFloat, legs::AbstractVector{Reco
    result = ReconstructedParticle[]
    length(legs) < 2 && return result
    for (a,b) in combinations(legs, 2)
        lv = LorentzVector(a.energy, a.momentum...) + LorentzVector(b.ene
        rcharge = a.charge + b.charge
        push!(result, ReconstructedParticle(mass=mass(lv), momentum=(lv.x
    end
    sort!(result, lt =  (a,b) -> abs(rmass-a.mass) < abs(rmass-b.mass))
    return result[1:1]  # take the best one
end;

"""
    recoilBuilder(comenergy::AbstractFloat, legs::AbstractVector{Reconstr
```

```
    build the recoil from an arbitrary list of input `ReconstructedPartic
"""
function recoilBuilder(comenergy::AbstractFloat, in::AbstractVector{Recon
    result = ReconstructedParticle[]
    isempty(in) && return result
    recoil_lv = LorentzVector(comenergy, 0, 0, 0)
    for p in in
        recoil_lv -= LorentzVector(p.mass, p.momentum...)
    end
    push!(result, ReconstructedParticle(mass=mass(recoil_lv), momentum=(r
    return result
end;
```

Defining the resulting analysis data

We create a custom structure with all summary information of each event.

In [4]:
```
using DataFrames

mutable struct AnalysisData <: AbstractAnalysisData
    df::DataFrame
    pevts::Int64
    sevts::Int64
    AnalysisData() = new(DataFrame(Zcand_m = Float32[], Zcand_recoil_m =
end
```

Open the data file to get the events

- It is using a file in EOS with the `root:` protocol
- The obtained `events` is a `LazyTree` created by the UnROOT.jl package. As the name indicates no event is actually read yet.

In [5]:
```
f = "root://eospublic.cern.ch//eos/experiment/fcc/ee/generation/DelphesEv
#f = "/Users/mato/cernbox/Data/events_000189367.root"
reader = RootIO.Reader(f);
events = RootIO.get(reader, "events");

reader
```

Out[5]:

```
┌─────────────────┬──────────────────────────────────────────────────────────
─────────
│ Atribute        │ Value
…
├─────────────────┼──────────────────────────────────────────────────────────
─────────
│ File Name(s)    │ root://eospublic.cern.ch//eos/experiment/fcc/ee/genera
tion/D …
│ # of events     │ 100000
…
│ IO Format       │ TTree
…
│ PODIO version   │ 0.16.2
…
│ ROOT version    │ 6.26.6
…
└─────────────────┴──────────────────────────────────────────────────────────
─────────
                                                                    1 column
omitted
```

BranchName	Type	CollectionID
CalorimeterHits	CalorimeterHit	0x00000007
EFlowNeutralHadron	Cluster	0x0000000d
EFlowNeutralHadron#0	ObjectID	0x00000000
EFlowNeutralHadron#1	ObjectID	0x00000000
EFlowNeutralHadron#2	ObjectID	0x00000000
EFlowPhoton	Cluster	0x0000000c
EFlowPhoton#0	ObjectID	0x00000000
EFlowPhoton#1	ObjectID	0x00000000
EFlowPhoton#2	ObjectID	0x00000000
EFlowTrack	Track	0x00000006
EFlowTrack#0	ObjectID	0x00000000
EFlowTrack#1	ObjectID	0x00000000
Electron#0	ObjectID	0x00000000
Jet	ReconstructedParticle	0x0000000e
Jet#0	ObjectID	0x00000000
Jet#1	ObjectID	0x00000000
Jet#2	ObjectID	0x00000000
Jet#3	ObjectID	0x00000000
Jet#4	ObjectID	0x00000000
Jet#5	ObjectID	0x00000000
MCRecoAssociations	MCRecoParticleAssociation	0x00000002
MCRecoAssociations#0	ObjectID	0x00000000
⋮	⋮	⋮

23 rows omitted

Loop over events and fill the DataFrame

In [6]:
```julia
function myanalysis!(data::AnalysisData, reader, events)
```

```
    for evt in events
        data.pevts += 1
        #---get the collection of Muons and ReconstructedParticles
        muids = RootIO.get(reader, evt, "Muon#0")
        length(muids) < 2 && continue
        recps = RootIO.get(reader, evt, "ReconstructedParticles")
        muons = recps[muids]        # use the objectids to collect the ref
        sel_muons = filter(x -> pₜ(x) > 10GeV, muons)
        zed_leptonic = resonanceBuilder(91GeV, sel_muons)
        zed_leptonic_recoil = recoilBuilder(240GeV, zed_leptonic)
        if length(zed_leptonic) == 1    #  Filter to have exactly one Z c
            Zcand_m        = zed_leptonic[1].mass
            Zcand_recoil_m = zed_leptonic_recoil[1].mass
            Zcand_recoil_θ = zed_leptonic_recoil[1].momentum |> EDM4hep.θ
            Zcand_q        = zed_leptonic[1].charge

            if 80GeV <= Zcand_m <= 100GeV
                push!(data.df, (Zcand_m, Zcand_recoil_m, Zcand_q, Zcand_r
                data.sevts += 1
            end
        end
    end
    return data
end
```

Out[6]: `myanalysis! (generic function with 1 method)`

In [7]:

```
N = Threads.nthreads()
data = AnalysisData();
```

In [13]:

```
elapsed1 = @elapsed do_analysis!(data, myanalysis!, reader, events; mt=fa
println("Serial: total time: $elapsed1, $(data.pevts/elapsed1) events/s.

elapsed2 = @elapsed do_analysis!(data, myanalysis!, reader, events; mt=tr
println("MT[$N]: total time: $elapsed2, $(data.pevts/elapsed2) events/s.
println("Speeedup: $(elapsed1/elapsed2)")
```

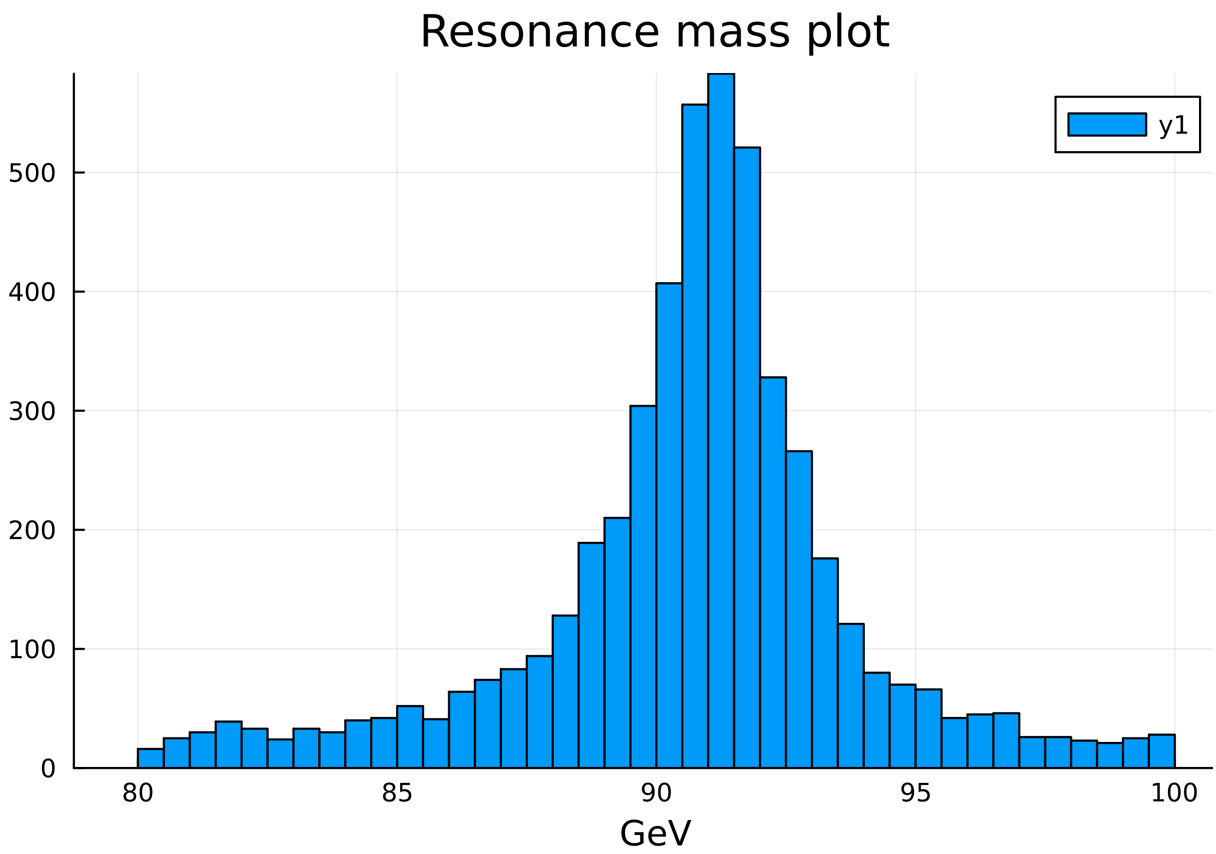
```
Serial: total time: 25.161413, 3974.339596905786 events/s. Selected event
s: 5008
MT[4]: total time: 15.496309083, 6453.14955092781 events/s. Selected event
s: 5008
Speeedup: 1.6237036100165918
```

Plot the results

In [14]:

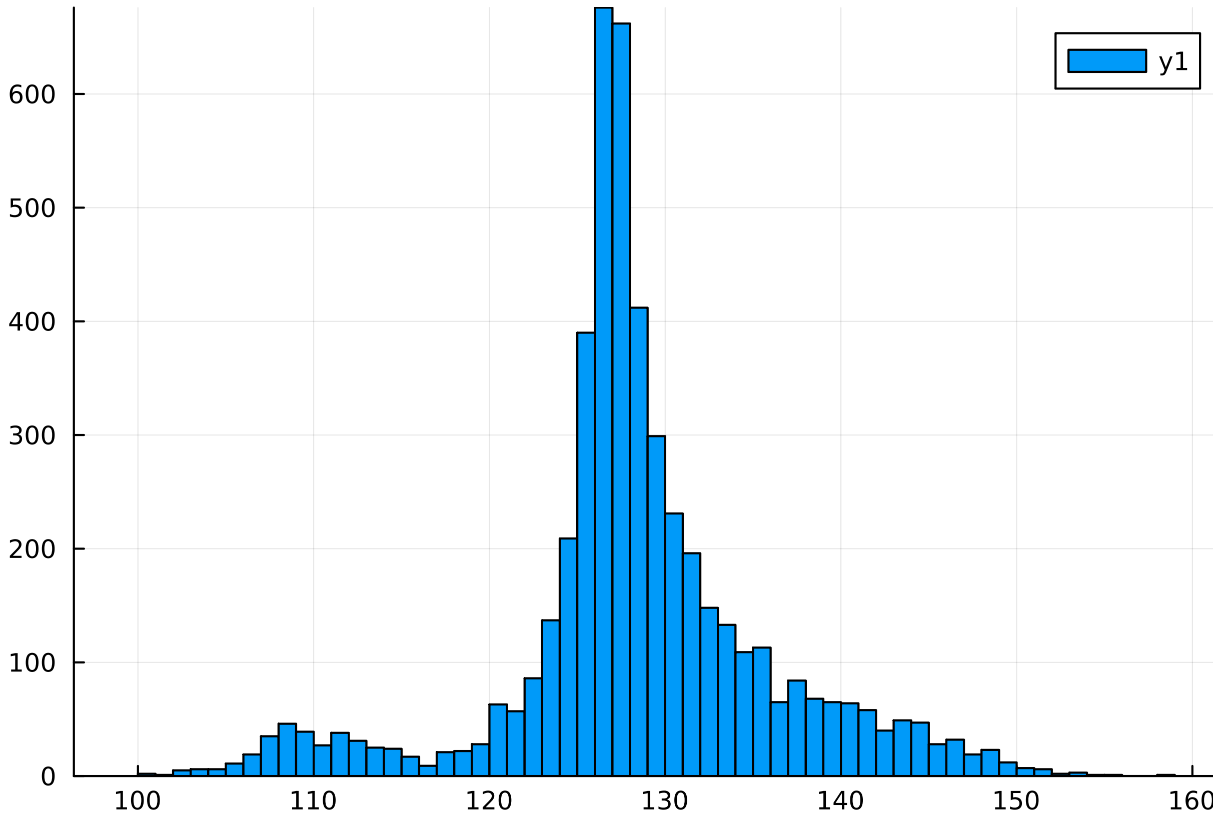
```
using Plots
histogram(data.df.Zcand_m, title="Resonance mass plot",xlabel="GeV")
```

Out[14]:



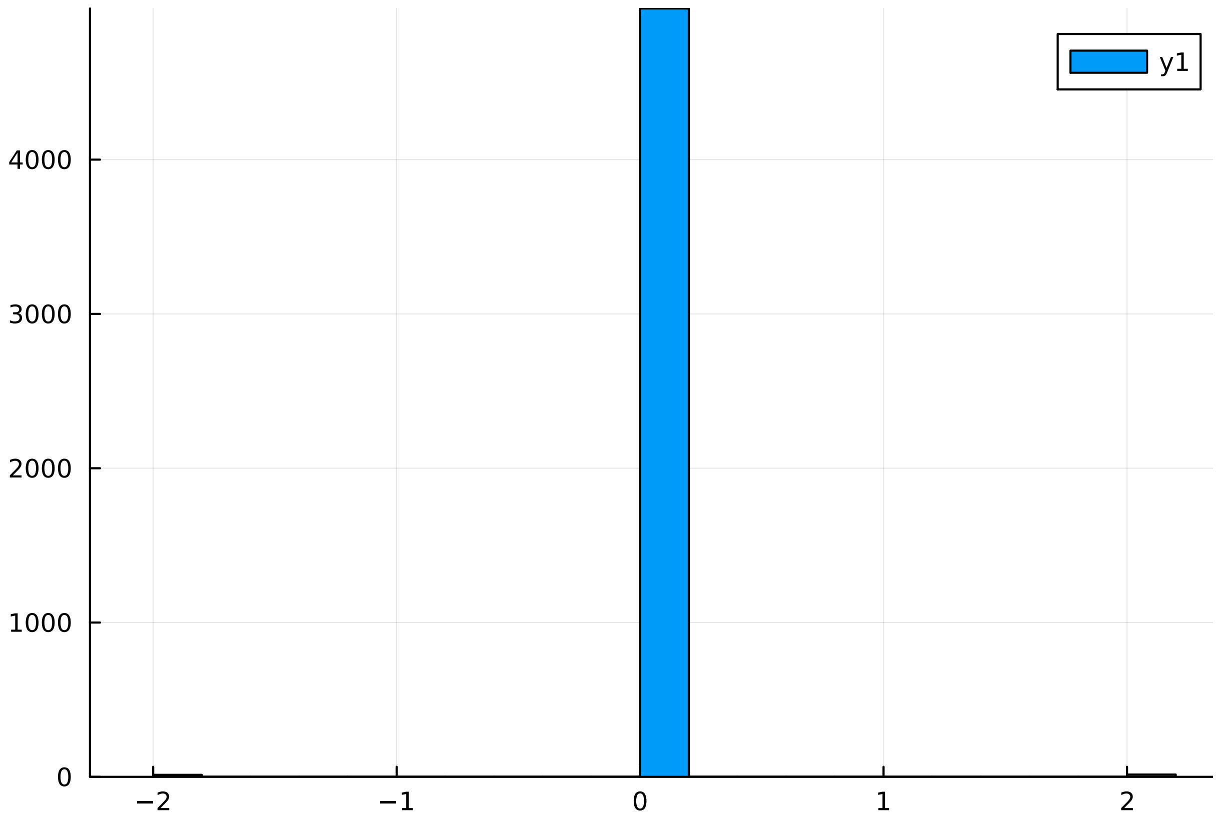
In [15]:
```
histogram(data.df.Zcand_recoil_m)
```

Out[15]:



In [16]:
```
histogram(data.df.Zcand_q)
```

Out[16]:



In [12]:

```
using Parquet2
Parquet2.writefile("m_H-recoil.parquet", data.df)
```

Out[12]:

```
Parquet2.FileWriter{IOStream}(m_H-recoil.parquet)
```

In [ ]: