Summer Student Update 3

An In Depth Look at the RDataFrame Macro

Konrad Helms

FTX Software Meeting

25.08.22

What happened so far:

- wrote many macros (with different sub-versions) for Higgs recoil mass in $e^+e^- \to Z^* \to ZH \to H\ell^+\ell^-$:

without bkg	with bkg
`higgs_recoil_edm4hep.C`	`higgs_recoil_with_bkg_edm4hep.C`
	`higgs_recoil_with_bkg_edm4hep_python.py`
	`higgs_recoil_with_bkg_edm4hep_uproot.py`
	`higgs_recoil_with_bkg_edm4hep_RDataframe.py` ✓

- macros are on GitLab (now with README), link also on INDICO page

The RDataFrame Macro

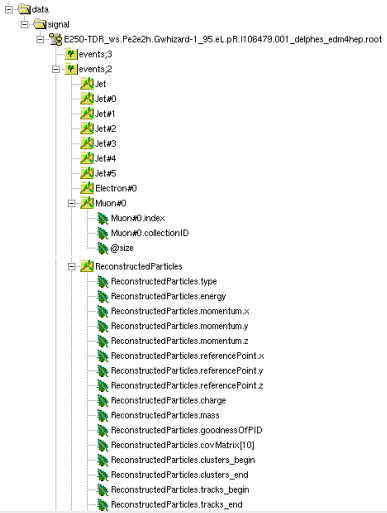
Problems:

- special characters

  $\Longrightarrow$ solution: use aliases

Incorrect:

```
df = (df.Define('idx','Muon#0.index')
.Filter('idx.size()==2'))
```



The RDataFrame Macro

Problems:

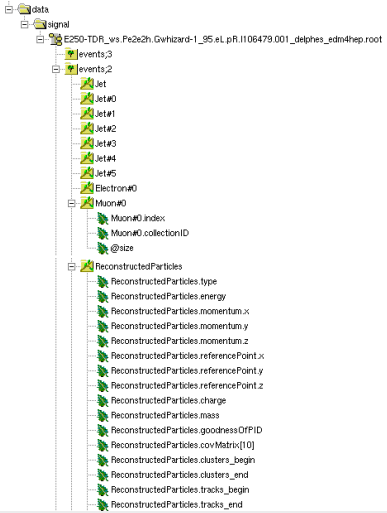
- special characters

  $\Longrightarrow$ solution: use aliases

Still incorrect:

```
df = (df.Alias('Muons','Muon#0')
.Define('idx','Muons.index')
.Filter('idx.size()==2'))
```

(reported to ROOT and will be fixed, see here)



The RDataFrame Macro

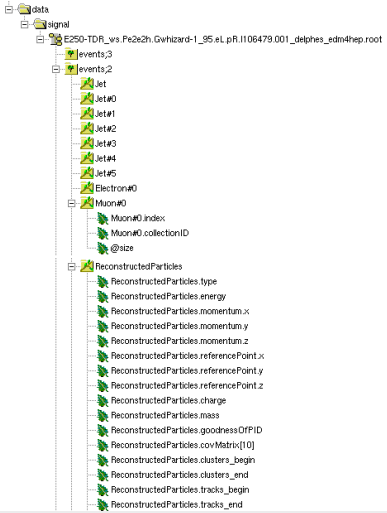
Problems:

- indexing
  - in RDataFrame only possible on leaf level
  - cannot do a 'conventional' event loop

$\Longrightarrow$ solution: utils file

Correct:

```
df = (df.Alias('Muons','Muon#0')
.Define('idx','utils::indices(Muons)')
.Filter('idx.size()==2'))
```



RDFUtils.cpp

```
ROOT::VecOps::RVec<int> indices(const ROOT::VecOps::RVec<podio::ObjectID>& objectIDs) {
    ROOT::VecOps::RVec<int> idcs;
    idcs.reserve(objectIDs.size());
    for (const auto& id : objectIDs) {
      idcs.push_back(id.index);
    }
    return idcs;
}

ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData> select(const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& recos, const
ROOT::VecOps::RVec<int>& indices) {
    RecoVec selectedRecos;
    selectedRecos.reserve(indices.size());
    for (const auto i : indices) {
        if (i > recos.size()) {
            std::cout << "error while indexing, check the inputs" << std::endl;
        }
        selectedRecos.emplace_back(recos[i]);
    }
    return selectedRecos;
}

ROOT::VecOps::RVec<float> p_x(const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& recos) {
    ROOT::VecOps::RVec<float> momentum;
    momentum.reserve(recos.size());
    for (const auto& r : recos) {
      momentum.push_back(r.momentum.x);
    }
    return momentum;
}
```

```
def makeLorentzVector(df):
    df = (df.Define("RecoMuons","utils::select(ReconstructedParticles,idx)")
            .Define("px","utils::p_x(RecoMuons)")
            .Define("py","utils::p_y(RecoMuons)")
            .Define("pz","utils::p_z(RecoMuons)")
            .Define("e","utils::e(RecoMuons)")
          )
    return df

def applyCut(df,nmuons=2):
    df = (df.Alias("Muons","Muon#0")
            .Define("idx","utils::indices(Muons)")
            .Filter(f"idx.size()=={nmuons}","Di-muon cut")
          )
    return df
```

The RDataFrame Macro

Problems:

- special characters $\Longrightarrow$ aliases
- indexing $\Longrightarrow$ need utils file that has to be compiled before

```
ROOT.gInterpreter.LoadFile("../RDFUtils.h")
ROOT.gSystem.Load("libRDFUtils.so")
```

- non intuitive ROOT-Python interface (str etc.) $\Longrightarrow$ needs patience

```
Einitial = 250. # [Gev], considering resonace here
pxinitial = 0.
angle = 0.007 # crossing angle parameter, change as needed

pxinitial = Einitial*angle
Einitial = 2.*np.sqrt((Einitial/2.)**2 + (pxinitial/2.)**2)

df = (df.Define("Einit", str(Einitial))
        .Define("pxinit", str(pxinitial))
     )
```

Runtime Comparison

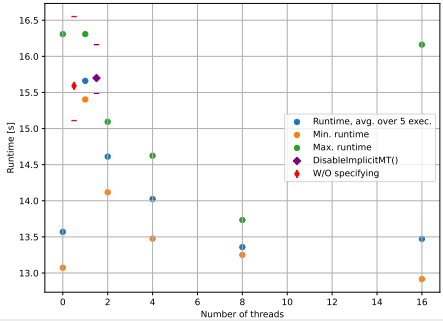
- averaged over 5 executions
- `time taskset -c <CPU Number> <run macro>`
- table shows total runtimes

Language	Macro	Avg. runtime	Min. runtime	Max. runtime
ROOT	c++ api	17.97 s	17.54 s	18.08 s
Python	Conventional evt. loop	20.20 s	19.86 *s*	20.56 s
	Uproot (lazy)	12.44 s	11.91 s	12.95 s
	Uproot (concatenate)	59.49 s	59.42 s	59.57 s
now correct !!!	**RDataFrame***	**15.60 s**	**15.11 s**	**16.55 s**
ROOT	c++ api, without bkg.	0.86 s	0.82 s	1.01 s

*: RDataFrame without `ROOT.EnableImplicitMT(...)` $\Longrightarrow$ ROOT chooses how many threads are taken

Runtime Speed Up via Multi-Threading (?)

- add: `ROOT.EnableImplicitMT(<number of threads>)`
- no speedup due to resource contention
- potential speedup bounded by Amdahl's law



To Do

- report unintuitive use of strings in DataFrames to ROOT
- write new macro to increase CPU load