FCCAnalyses*: tools and algorithms for analysis

Clement Helsens CERN-EP

Virtual FCC week, June 2021

Outline

- Introduction
- Overall Structure
- Workflow with one example

* This is not specific to FCC, but to the input data format

Introduction

FCCAnalyses: https://github.com/HEP-FCC/FCCAnalyses/

- Set of tools to for data analysis, based on LHC experience
  - Originally developed for FCC-hh physics studies
  - Was based on a CMS python analysis framework (event loop slow)
  - Was used to produced FCC-hh physics analyses published in the CDR

- Transitioned to RDataFrame
  - State of the art in HEP (considering it is ROOT based)
  - Achieved much higher performances
  - Fully compatible with EDM4hep

Common software?

- FCCAnalyses
  - Is a common tool for analyzing large amount of data using RDataFrame
  - Is composed of a library of C++ analysers and python configuration files
    - C++ analysers are developed in common
    - Python code specific to the analysis, define analysers, output variables, input samples, etc…
  - Use the produced ROOT ntuples
    - To produce, with common tools, final variables for analysis, for MVA training, and for plotting

- EDM4Hep
  - EDM4Hep is the common event data model
  - Will be used for the very long term for any simulation
    - better to start using it now!

Overall structure for case studies - 1

- Set of tools to help processing the output of 'simulation'
  - Agnostic to the type of simulation but specific reader functions are required
  - Build a common set of utility functions, algorithms for common use
  - Users to test their algorithms locally, or in dedicated repositories before publishing them

Analysis configuration
4 python scripts to configure:
1. Samples to run over (common production, EDM4Hep)
2. Functions/algorithm to call to produce variables of interests in a flat ntuple
3. Event selection and histograms definition using the flat ntuples
4. Plotting configuration

https://github.com/HEP-FCC/HEP-FCC/FCCeePhysicsPerformance/ Analysis configuration part close to the case studies and analysis specific code (like fitting, specific calculations, etc…)

Common utility functions, algorithm, etc…

C++ library

Common interface code
Sample database, RdataFrame, plotting
Python

https://github.com/HEP-FCC/FCCAnalyses/
Centrally installed on cvmfs and accessible by common setup. But totally OK to use private version.

Overall structure for case studies - 2

master FCCeePhysicsPerformance / case-studies / higgs / mH-recoil / Go to file Add file

clementhelsens change path b74cbdf 10 days ago History

..		
FCCAnalyses-config	change path	10 days ago
analysis	new configuration skim	13 days ago
images	new configuration skim	13 days ago
README.md	new configuration skim	13 days ago

- Analysis configuration -> in FCCAnalyses-config
  - 4 python scripts to configure:
    1. Samples to run over (common production, EDM4Hep)
    2. Functions/algorithm to call to produce variables of interests in a flat ntuple
    3. Event selection and histograms definition using the flat ntuples
    4. Plotting configuration

Overall structure for case studies - 3

master FCCeePhysicsPerformance / case-studies / higgs / dataframe / Go to file Add file ···

clementhelsens Update localSetup.sh ✓ 758a156 11 days ago History

analyzers	move up	4 months ago
CMakeLists.txt	move up	4 months ago
README.md	move up	4 months ago
localSetup.sh	Update localSetup.sh	11 days ago

- Code specific to the case study -> in dataframe/analyzers
  - Code your ideas here before pushing them for more general usage
  - Could be very specific to a case study
  - But a case study could also be just fine using all the functionalities/algorithm that are available in the main library

Foresee overall structure for case studies

Analysis configuration
4 python scripts to configure:
1. Samples to run over (common production, EDM4Hep)
2. Functions/algorithm to call to produce variables of interests in a flat ntuple
3. Event selection and histograms definition using the flat ntuples
4. Plotting configuration

Analysis Algorithms
Dedicated C++ algorithms not available in the common library

https://github.com/HEP-FCC/HEP-FCC/FCCeePhysicsPerformance/

Analysis configuration part close to the case studies and analysis specific code (like fitting, specific calculations, etc…)

Common utility functions, algorithm, dedicated to FCC
C++ library

Common interface code
Sample database, RdataFrame config, plotting
Python

https://github.com/HEP-FCC/FCCAnalyses/
Centrally installed on cvmfs and accessible by common setup. But totally OK to use private version.

Common utility functions, algorithm, of interest for any k4Analyses using EDM4Hep

C++ library

https://github.com/key4hep/k4Analysis
Centrally installed on cvmfs and accessible by common setup. But totally OK to use private version.

Workflow with one Example

B0 -> mumu

(took me ~45min to produce signal events and write the code, 1h to run it)

Produce signal events with EventProducer (not covered in this talk)

```
python bin/run.py --FCCee --reco --send -p p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu -n 100000
--type p8 -N 10 --condor -q microcentury --version spring2021  --detector IDEA
```

Home | Contact | FCC-ee | Les Houches | STDHEP | IDEA | IDEA 3T | IDEA Full Silicon | **Stat**

Delphes FCCee Physic events spring 2021 production (IDEA with Track Covariance full matrix lower triangle)

bd2mumu

NO	NAME	NEVENTS	NWEIGHTS	NFILES	NBAD	NEOS	SIZE (GB)	OUTPUT PATH
108	p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu	1,000,000	0	10	0	10	4.87	/eos/experiment/fcc/ee/generation /DelphesEvents/spring2021 /IDEA/p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu/

Common samples (official productions) are available on eos and can be accessed through a database. The sample name is the “key”

But still possible to run with your own files

http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_spring2021_IDEA.php

B0 -> mumu github

HEP-FCC / **FCCAnalyses**

Unwatch 10 | Star 2 | Fork 17

Code | Pull requests 3 | Actions | Projects | Wiki | Security | Insights | Settings

awkward | FCCAnalyses / examples / FCCee / flavour / **Bd2MuMu** / | Go to file | Add file | ···

This branch is 101 commits ahead, 18 commits behind master. Contribute

clementhelsens add plots ✕ d6b3b42 20 hours ago History

..

analysis.py	fix typo	20 hours ago
finalSel.py	add final sel	20 hours ago
plots.py	add plots	20 hours ago
preSel.py	add preSel for Bd2MuMu	20 hours ago

B0 -> mumu

Produce signal events with EventProducer (not covered in this talk)

```
python bin/run.py --FCCee --reco --send -p p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu -n 100000
--type p8 -N 10 --condor -q microcentury --version spring2021  --detector IDEA
```

Run first stage selection (run over 10^9 Z->bb, and 100k B0->MuMu)

```
python examples/FCCee/flavour/Bd2MuMu/preSel.py
```

Vertexing

analysis.py

```
34      #_____________________________________________________
35      def run(self):
36          df2 = (self.df
37                  ##############################################
38                  ##          Aliases for # in python         ##
39                  ##############################################
40                  .Alias("MCRecoAssociations0", "MCRecoAssociations#0.index")
41                  .Alias("MCRecoAssociations1", "MCRecoAssociations#1.index")
42                  .Alias("Particle0", "Particle#0.index")
43                  .Alias("Particle1", "Particle#1.index")
44
45                  ##############################################
46                  ##               Build MC Vertex            ##
47                  ##############################################
48                  .Define("MCVertexObject", "myUtils::get_MCVertexObject(Particle, Particle0)")
49
50                  ##############################################
51                  ##              Build Reco Vertex           ##
52                  ##############################################
53                  .Define("VertexObject", "myUtils::get_VertexObject(MCVertexObject,ReconstructedParticles,EFlowTrack_1,MCRecoAssociations0,MCRecoAssociations1)")
54
55                  ##############################################
56                  ##          Build PV var and filter         ##
57                  ##############################################
58                  .Define("EVT_hasPV",      "myUtils::hasPV(VertexObject)")
59                  .Define("EVT_NtracksPV", "myUtils::get_PV_ntracks(VertexObject)")
60                  .Define("EVT_NVertex",    "VertexObject.size()")
61                  .Filter("EVT_hasPV==1")
```

PID/B0 -> mumu

analysis.py

```
62
63          ##########################################
64          ##          Build new RecoP with PID       ##
65          ##########################################
66          .Define("RecoPartPID" ,"myUtils::PID(ReconstructedParticles, MCRecoAssociations0,MCRecoAssociations1,Particle)")
67
68          ##########################################
69          ##  Build new RecoP with PID at vertex     ##
70          ##########################################
71          .Define("RecoPartPIDAtVertex" ,"myUtils::get_RP_atVertex(RecoPartPID, VertexObject)")
72
73          ##########################################
74          ##        Build B0 -> MuMu candidates      ##
75          ##########################################
76          .Define("Bd2MuMuCandidates",         "myUtils::build_Bd2MuMu(VertexObject,RecoPartPIDAtVertex)")
77
78          ##########################################
79          ##       Filter B0 -> MuMu candidates      ##
80          ##########################################
81          .Define("EVT_NBd2MuMu",              "float(myUtils::getFCCAnalysesComposite_N(Bd2MuMuCandidates))")
82          .Filter("EVT_NBd2MuMu==1")
83
84          ##########################################
85          ##    Get the B0 -> MuMu candidate mass    ##
86          ##########################################
87          .Define("Bd2MuMu_mass",    "myUtils::getFCCAnalysesComposite_mass(Bd2MuMuCandidates)")
88
89
90      )
```

PID/B0 -> mumu

analysis.py

```
62
63          ##########################################
64          ##          Build new RecoP with PID       ##
65          ##########################################
66          .Define("RecoPartPID" ,"myUtils::PID(ReconstructedParticles, MCRecoAssociations0,MCRecoAssociations1,Particle)")
67
68          ##########################################
69          ##  Build new RecoP with PID at vertex     ##
70          ##########################################
71          .Define("RecoPartPIDAtVertex" ,"myUtils::get_RP_atVertex(RecoPartPID, VertexObject)")
72
73          ##########################################
74          ##        Build B0 -> MuMu candidates      ##
75          ##########################################
76          .Define("Bd2MuMuCandidates",        "myUtils::build_Bd2MuMu(VertexObject,RecoPartPIDAtVertex)")
77
78          ##########################################
79          ##       Filter B0 -> MuMu candidates      ##
80          ##########################################
81          .Define("EVT_NBd2MuMu",              "float(myUtils::getFCCAnalysesComposite_N(Bd2MuMuCandidates))")
82          .Filter("EVT_NBd2MuMu==1")
83
84          ##########################################
85          ##    Get the B0 -> MuMu candidate mass    ##
86          ##########################################
87          .Define("Bd2MuMu_mass",    "myUtils::getFCCAnalysesComposite_mass(Bd2MuMuCandidates)")
88
89
90      )
```

PID/B0 -> mumu

analysis.py

```
62
63          ##########################################
64          ##          Build new RecoP with PID       ##
65          ##########################################
66          .Define("RecoPartPID" ,"myUtils::PID(ReconstructedParticles, MCRecoAssociations0,MCRecoAssociations1,Particle)")
67
68          ##########################################
69          ##  Build new RecoP with PID at vertex     ##
70          ##########################################
71          .Define("RecoPartPIDAtVertex" ,"myUtils::get_RP_atVertex(RecoPartPID, VertexObject)")
72
73          ##########################################
74          ##        Build B0 -> MuMu candidates      ##
75          ##########################################
76          .Define("Bd2MuMuCandidates",         "myUtils::build_Bd2MuMu(VertexObject,RecoPartPIDAtVertex)")
77
78          ##########################################
79          ##       Filter B0 -> MuMu candidates      ##
80          ##########################################
81          .Define("EVT_NBd2MuMu",              "float(myUtils::getFCCAnalysesComposite_N(Bd2MuMuCandidates))")
82          .Filter("EVT_NBd2MuMu==1")
83
84          ##########################################
85          ##    Get the B0 -> MuMu candidate mass    ##
86          ##########################################
87          .Define("Bd2MuMu_mass",    "myUtils::getFCCAnalysesComposite_mass(Bd2MuMuCandidates)")
88
89
90      )
```

```
92          ##########################################
93          ##      select branches for output file    ##
94          ##########################################
95          branchList = ROOT.vector('string')()
96          for branchName in ["Bd2MuMu_mass" ]:
97              branchList.push_back(branchName)
98
99          df2.Snapshot("events", self.outname, branchList)
```

B0 -> mumu candidate building

```
ROOT::VecOps::RVec<FCCAnalysesComposite2> myUtils::build_Bd2MuMu(ROOT::VecOps::RVec<VertexingUtils::FCCAnalysesVertex> vertex,
                                                                 ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData> recop){

  ROOT::VecOps::RVec<FCCAnalysesComposite2> result;
  //loop over the reconstructed vertex collection
  for (size_t i=0;i<vertex.size();i++){

    //not consider PV, exactly 2 tracks
    if (vertex.at(i).vertex.primary==1)continue;
    if (vertex.at(i).ntracks!=2)       continue;

    //2 tracks id as muons
    int charge_Bd=0;
    int nobj_Bd=0;
    for (auto &r:vertex.at(i).reco_ind){
      if (recop.at(r).type==13 ){
        nobj_Bd+=1;
        charge_Bd+=recop.at(r).charge;
      }
    }

    //select candidates with exactly 2 muons and charge 0
    if (nobj_Bd!=2)   continue;
    if (charge_Bd!=0) continue;

    //build a composite vertex
    FCCAnalysesComposite2 comp;
    comp.vertex   = i;
    comp.particle = build_tlv(recop,vertex.at(i).reco_ind);
    comp.charge   = charge_Bd;

    //add the composite vertex to the collection
    result.push_back(comp);
  }
  return result;
}
```

myUtils.cc

Run pre-selection

preSel.py

```python
#python examples/FCCee/flavour/Bd2MuMu/preSel.py

from config.common_defaults import deffccdicts
import config.runDataFrameBatch as rdf
import os

basedir=os.path.join(os.getenv('FCCDICTSDIR', deffccdicts), '') + "yaml/FCCee/spring2021/IDEA/"
outdir="/eos/experiment/fcc/ee/analyses/case-studies/flavour/Bd2MuMu/flatNtuples/spring2021/Batch/"
NUM_CPUS=8
output_list=[]

fraction=1.

inputana="/afs/cern.ch/user/h/helsens/FCCsoft/HEP-FCC/FCCAnalyses/examples/FCCee/flavour/Bd2MuMu/analysis.py"
process_list=['p8_ee_Zbb_ecm91']
myana=rdf.runDataFrameBatch(basedir,process_list, outlist=output_list)
myana.run(ncpu=NUM_CPUS,fraction=fraction, chunks=50 ,outDir=outdir, inputana=inputana)


process_list=['p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu']
import config.runDataFrame as rdf2
myana=rdf2.runDataFrame(basedir,process_list, outlist=output_list)
myana.run(ncpu=NUM_CPUS,fraction=fraction ,outDir=outdir)

```

B0 -> mumu

Produce signal events with EventProducer (not covered in this talk)

```
python bin/run.py --FCCee --reco --send -p p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu -n 100000
--type p8 -N 10 --condor -q microcentury --version spring2021  --detector IDEA
```

Run first stage selection (run over 10^9 Z->bb, and 100k B0->MuMu)

```
python examples/FCCee/flavour/Bd2MuMu/preSel.py
```

Run final selection and make histograms

```
python examples/FCCee/flavour/Bd2MuMu/finalSel.py
```

Run final-selection

finalSel.py

```python
#python examples/FCCee/flavour/Bd2MuMu/finalSel.py

from config.common_defaults import deffccdicts
import sys, os
import ROOT

###Input directory where the files produced at the pre-selection level are
baseDir  ="/eos/experiment/fcc/ee/analyses/case-studies/flavour/Bd2MuMu/flatNtuples/spring2021/Batch/"

###Link to the dictonary that contains all the cross section informations etc...
procDict = os.path.join(os.getenv('FCCDICTSDIR', deffccdicts), '') + "FCCee_procDict_spring2021_IDEA.json"

process_list=['p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu',
              'p8_ee_Zbb_ecm91'
              ]

define_list={}

###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":"Bd2MuMu_mass>0"}

###Dictionary for the ouput variable/hitograms. The key is the name of the variable in the output files. "name" is the nam
variables = {
    "EVT_CandMass"       :{"name":"Bd2MuMu_mass","title":"mass [GeV]","bin":300,"xmin":0,"xmax":6.},
    "EVT_CandMass_zoom"  :{"name":"Bd2MuMu_mass","title":"mass [GeV]","bin":100,"xmin":5.,"xmax":6.}
}

###Number of CPUs to use
NUM_CPUS = 4

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

B0 -> mumu

Produce signal events with EventProducer (not covered in this talk)

```
python bin/run.py --FCCee --reco --send -p p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu -n 100000
--type p8 -N 10 --condor -q microcentury --version spring2021  --detector IDEA
```

Run first stage selection (run over 10^9 Z->bb, and 100k B0->MuMu)

```
python examples/FCCee/flavour/Bd2MuMu/preSel.py
```

Run final selection and make histograms

```
python examples/FCCee/flavour/Bd2MuMu/finalSel.py
```

Produce the plots

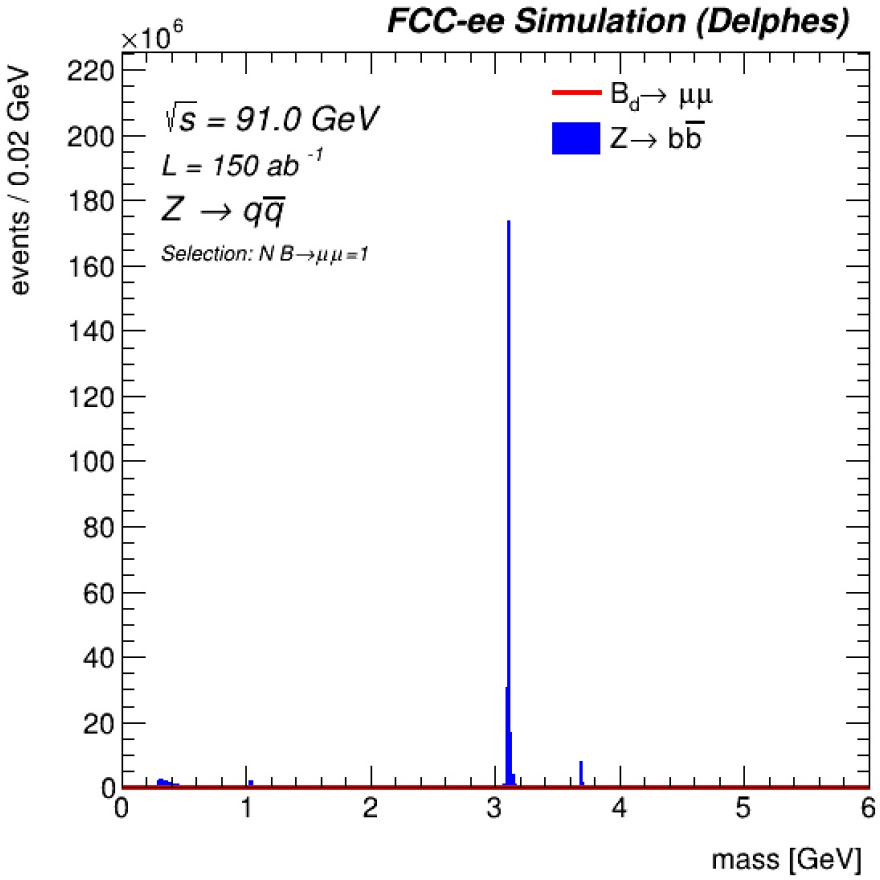
```
python config/doPlots.py examples/FCCee/flavour/Bd2MuMu/plots.py
```

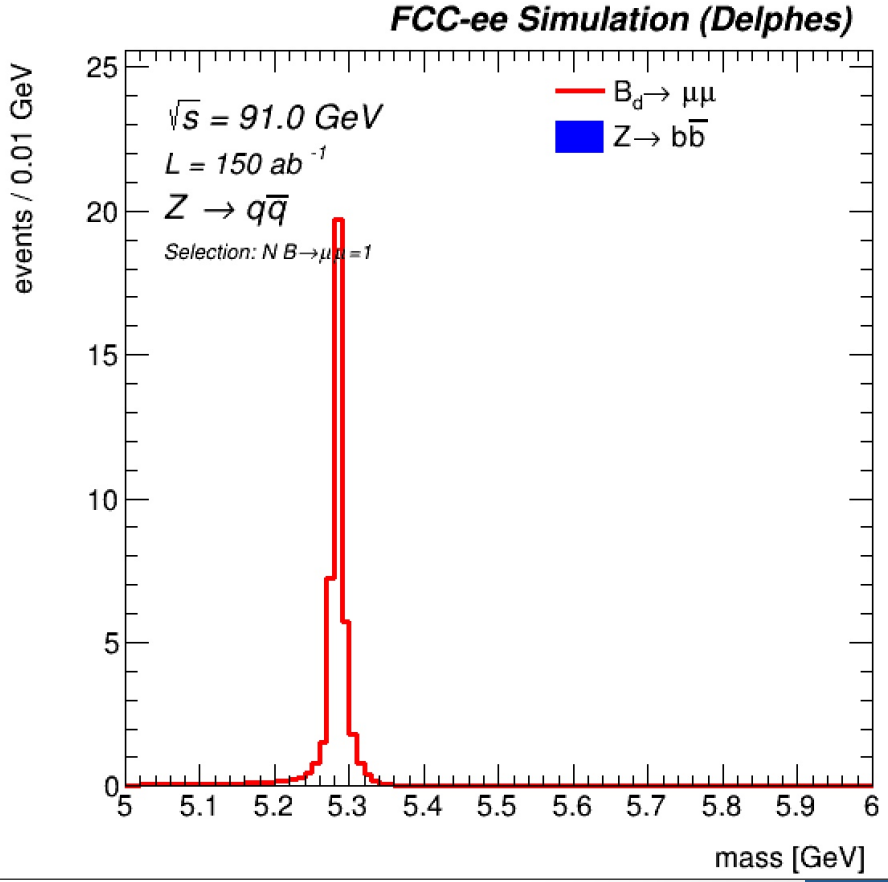
Run plots

plots.py

```python
import ROOT

# global parameters
intLumi        = 150000000. #in pb-1
ana_tex        = "Z #rightarrow q#bar{q}"
delphesVersion = "3.4.2"
energy         = 91.0
collider       = "FCC-ee"
inputDir       = "/eos/experiment/fcc/ee/analyses/case-studies/flavour/Bd2MuMu/flatNtuples/spring2021/Batch/"
formats        = ['png','pdf']
yaxis          = ['lin','log']
stacksig       = ['nostack']
outdir         = 'plots_Bd2MuMu/'
variables      = ["EVT_CandMass","EVT_CandMass_zoom"]

scaleSig=1.
###Dictonnary with the analysis name as a key, and the list of selections to be plotted for this analysis.
###The name of the selections should be the same than in the final selection
selections = {}
selections['Bd2MuMu']       = ["sel0"]

extralabel = {}
extralabel['sel0'] = "Selection: N B#rightarrow#mu#mu=1"

colors = {}
colors['Z_bb']  = ROOT.kBlue
colors['Z_Bd']  = ROOT.kRed

plots = {}
plots['Bd2MuMu'] = {'signal':{'Z_Bd':['p8_ee_Zbb_ecm91_EvtGen_Bd2MuMu']},
                    'backgrounds':{'Z_bb':['p8_ee_Zbb_ecm91']}
                   }

legend = {}
legend['Z_Bd']    = 'B_{d}#rightarrow #mu#mu'
legend['Z_bb']    = 'Z#rightarrow b#bar{b}'
```

Candidate mass plots





Summary

- Easy to use analysis code to produce root ntuples based on RDataFrame
- Extremely powerful and flexible
- Already actively used: flavour, higgs and top physics at FCC-ee
- Large scale usage with Bc->tauNu analysis (~10^{10} events, 12h processing on batch with 9000HS06, not sure the fraction of it I got though)
- Already contains lot's of functionalities
  - Jet clustering, vertexing, thrust/sphericity axis, simple PID (more accurate to come), etc…

- Started to move FCCAnalyses from HEP-FCC to Key4Hep/k4Analyses
  - https://github.com/key4hep/k4Analysis
  - New schema needs to be defined in the next weeks (where the various codes have to be)
  - Split C++ and python