FCCAnalyses today

Juraj Smieško (CERN)

7th FCC Physics Workshop

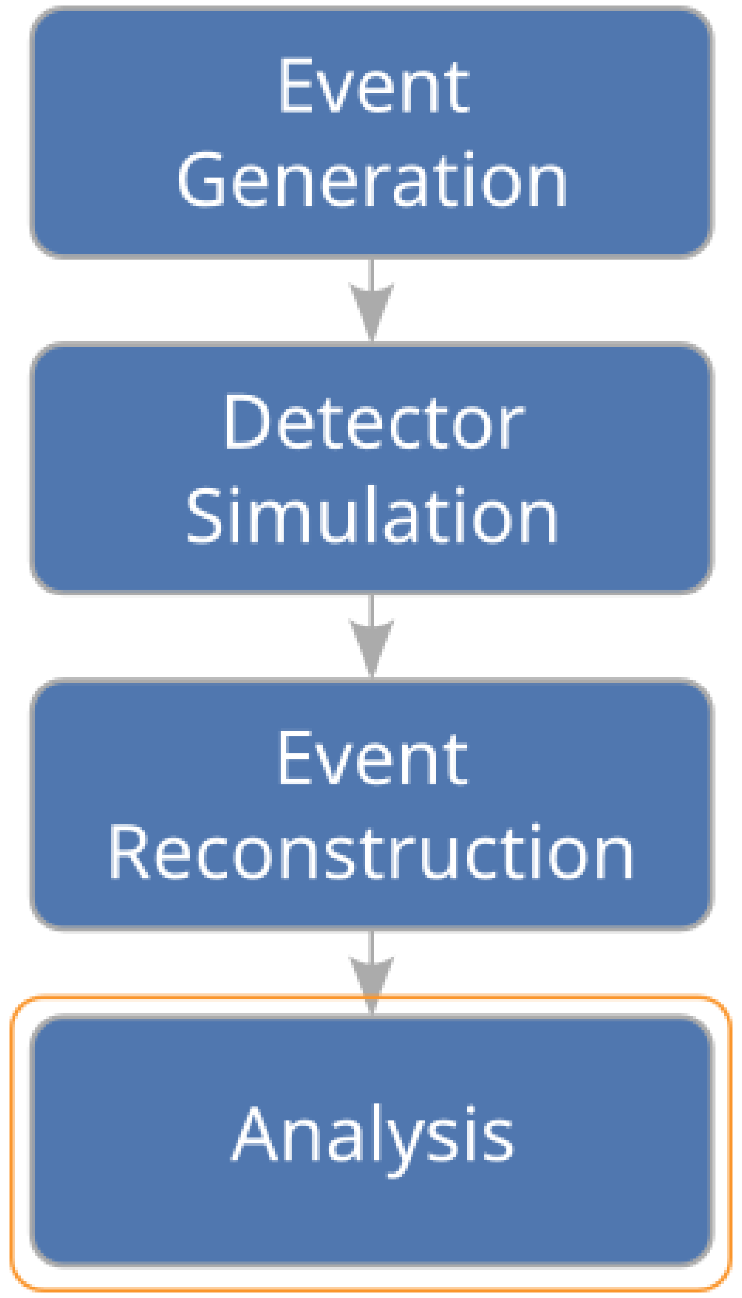
Annecy, 01 Feb 2024

FCCAnalyses Scope

Goal of the framework is to aid the users in obtaining the desired physics results from the reconstructed objects

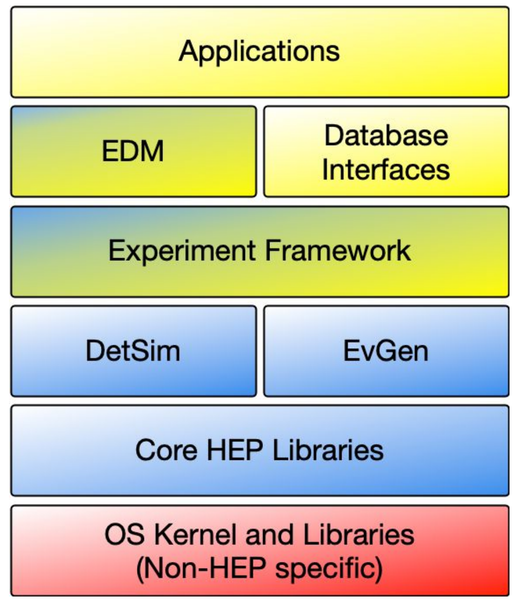
Framework requirements:

- Efficiency — Make quick turn-around possible
- Flexibility — Allow heavy customization
- Ease of use — Should not be hard to start using
- Scalable — Seamlessly handle from small to large datasets



Key4hep

- Set of common software packages, tools, and standards for different Detector concepts
- Common for FCC, CLIC/ILC, CEPC, EIC, ...
- Individual participants can mix and match their stack
- Main ingredients:
  - Data processing framework: Gaudi
  - Event data model: EDM4hep
  - Detector description: DD4hep
  - Software distribution: Spack



Applications

EDM

Database Interfaces

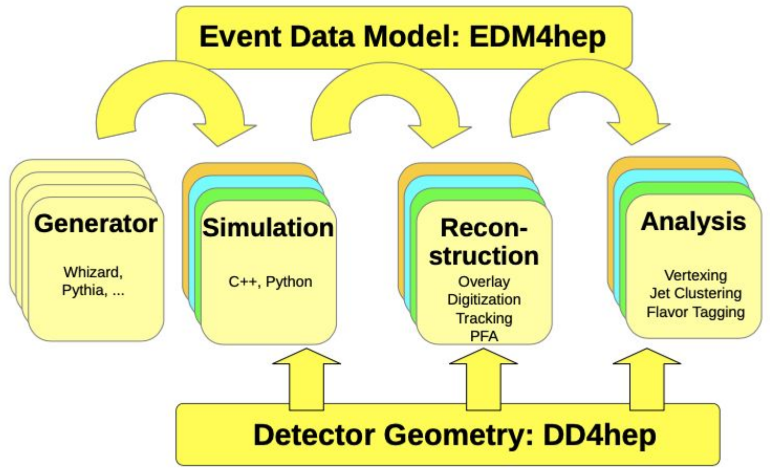
Experiment Framework

DetSim

EvGen

Core HEP Libraries

OS Kernel and Libraries (Non-HEP specific)



Event Data Model: EDM4hep

Generator

Whizard, Pythia, ...

Simulation

C++, Python

Recon-struction

Overlay

Digitization

Tracking

PFA

Analysis

Vertexing

Jet Clustering

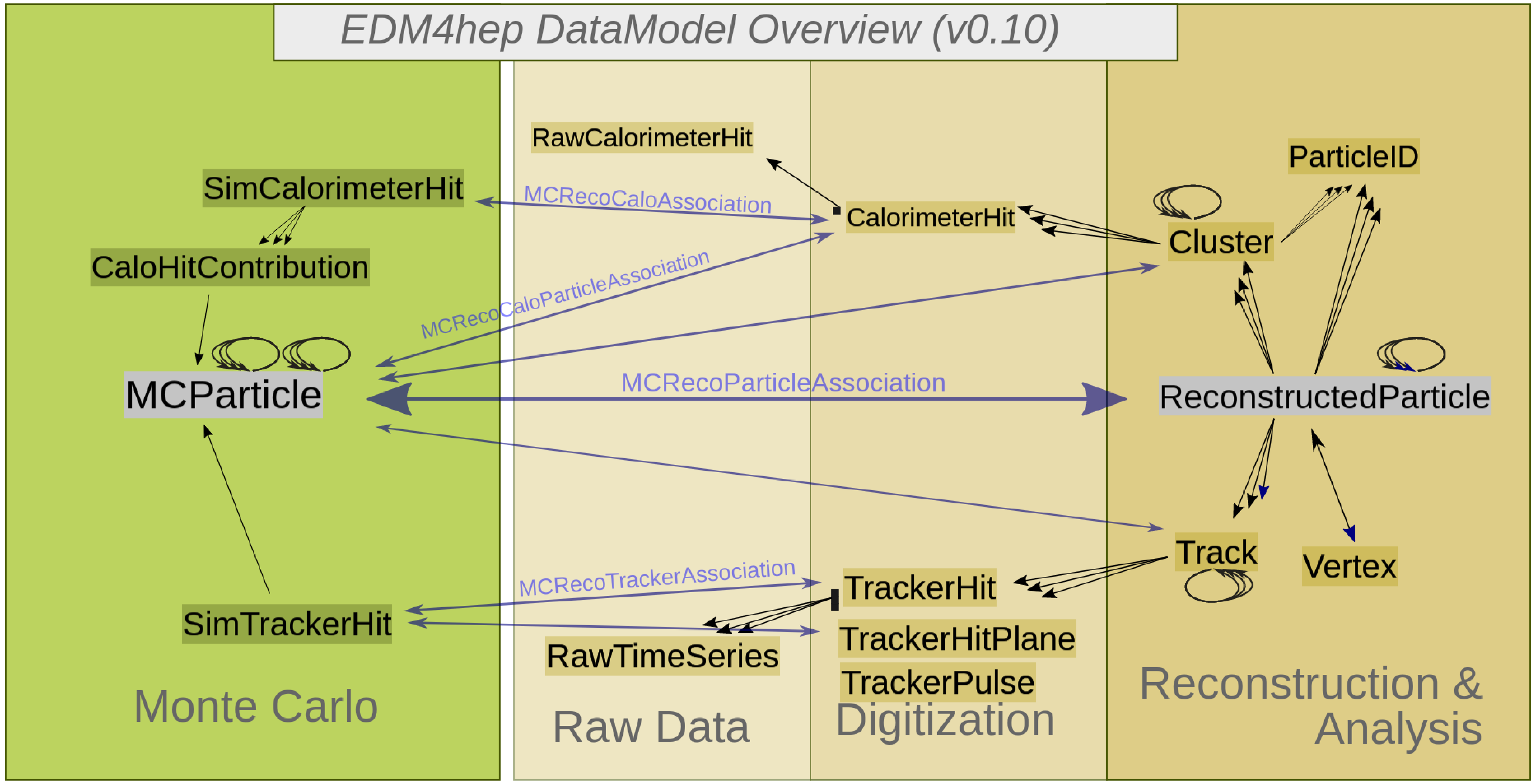
Flavor Tagging

Detector Geometry: DD4hep

EDM4hep I.

Describes event data with the set of standard objects.

- Specification in a single YAML file
- Generated with the help of Podio



EDM4hep II.

Example object:

```
#-------------  CalorimeterHit
edm4hep::CalorimeterHit:
  Description: "Calorimeter hit"
  Author : "F.Gaede, DESY"
  Members:
    - uint64_t cellID             //detector specific (geometrical) cell id.
    - float energy                //energy of the hit in [GeV].
    - float energyError           //error of the hit energy in [GeV].
    - float time                  //time of the hit in [ns].
    - edm4hep::Vector3f position //position of the hit in world coordinates in [mm].
    - int32_t type                //type of hit. Mapping of integer types to names via collection parameters "CalorimeterHitTypeNames" an
```

- Current version: `v0.10.3`
- Objects can be extended / new created
- Bi-weekly discussion: Indico

Datasets

Plethora of processes are pre-generated and available from EOS

- Two main production campaigns in use:
  - Spring 2021
  - Winter 2023
- Processes are identified by its name, e.g.: `p8_ee_WW_ecm240`
- The production Database browsable at: fcc-physics-events.web.cern.ch
- Example: Delphes events, IDEA, FCCee, winter 2023
- EOS directory: `/eos/experiment/fcc/...`
- Generation handled by EventProducer
  - Heads up: Will change soon (Dirac, iLCDirac)

EOS Space

Various intermediate files of common interest can be stored at: `/eos/experiment/fcc/ee/analyses_storage/...`

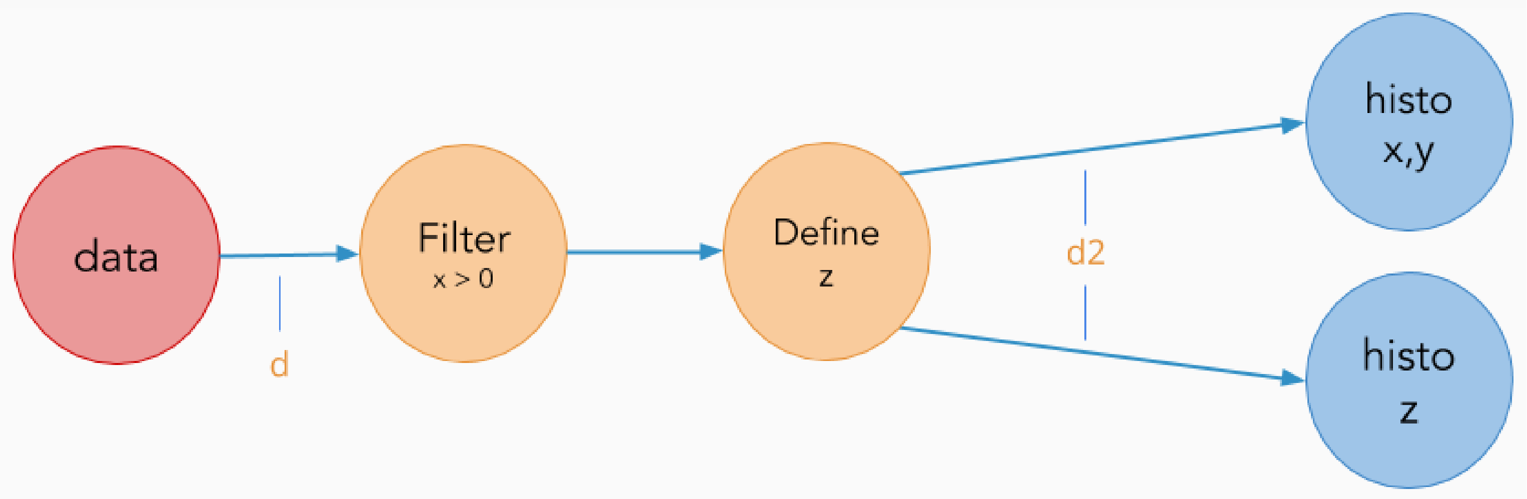
in four subfolders:

- BSM
- EW_and_QCD
- flavor
- Higgs_and_TOP

Access and quotas:

- Read access is is granted to anyone
- Write access needs to be granted: Ask your convener :)
- Total quota for all four directories is `200TB`
- ATM only part of the quota is allocated

ROOT RDataFrame



- Describes processing of data as actions on table columns
  - Defines of new columns
  - Filter rules
  - Result definitions (histogram, graph)
- The actions are lazily evaluated
- Multi threading is available out of the box
- Optimized for bulk processing
- Allows integration of existing C++ libraries

Functional Approach

- The operations on the dataframe happen with small stateless functions:

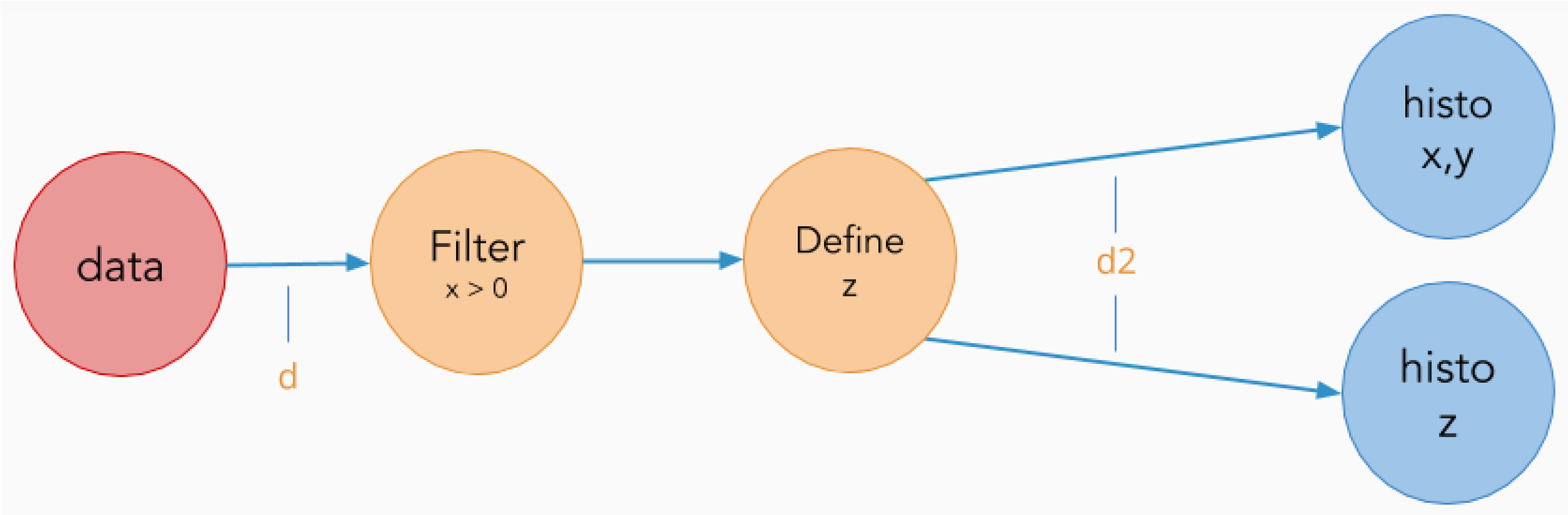
```cpp
float getMass(const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& in) {
  ROOT::Math::LorentzVector<ROOT::Math::PxPyPzE4D<double>> result;

  for (auto & p: in) {
    ROOT::Math::LorentzVector<ROOT::Math::PxPyPzE4D<double>> tmp;
    tmp.SetPxPyPzE(p.momentum.x, p.momentum.y, p.momentum.z, p.energy);
    result+=tmp;
  }

  return result.M();
}
```

- or with structs, which have internal state:

```cpp
/// Get the number of particles in a given hemisphere (defined by it's angle wrt to axis). Returns 3 values: total, charged, neutra
struct getAxisN {
public:
  getAxisN(bool arg_pos=0);
  ROOT::VecOps::RVec<int> operator() (const ROOT::VecOps::RVec<float> & angle,
                                      const ROOT::VecOps::RVec<float> & charge);
private:
  bool _pos; /// Which hemisphere to select, false/0=cosTheta<0 true/1=cosTheta>0. Default=0
};
```

Analysis as a graph

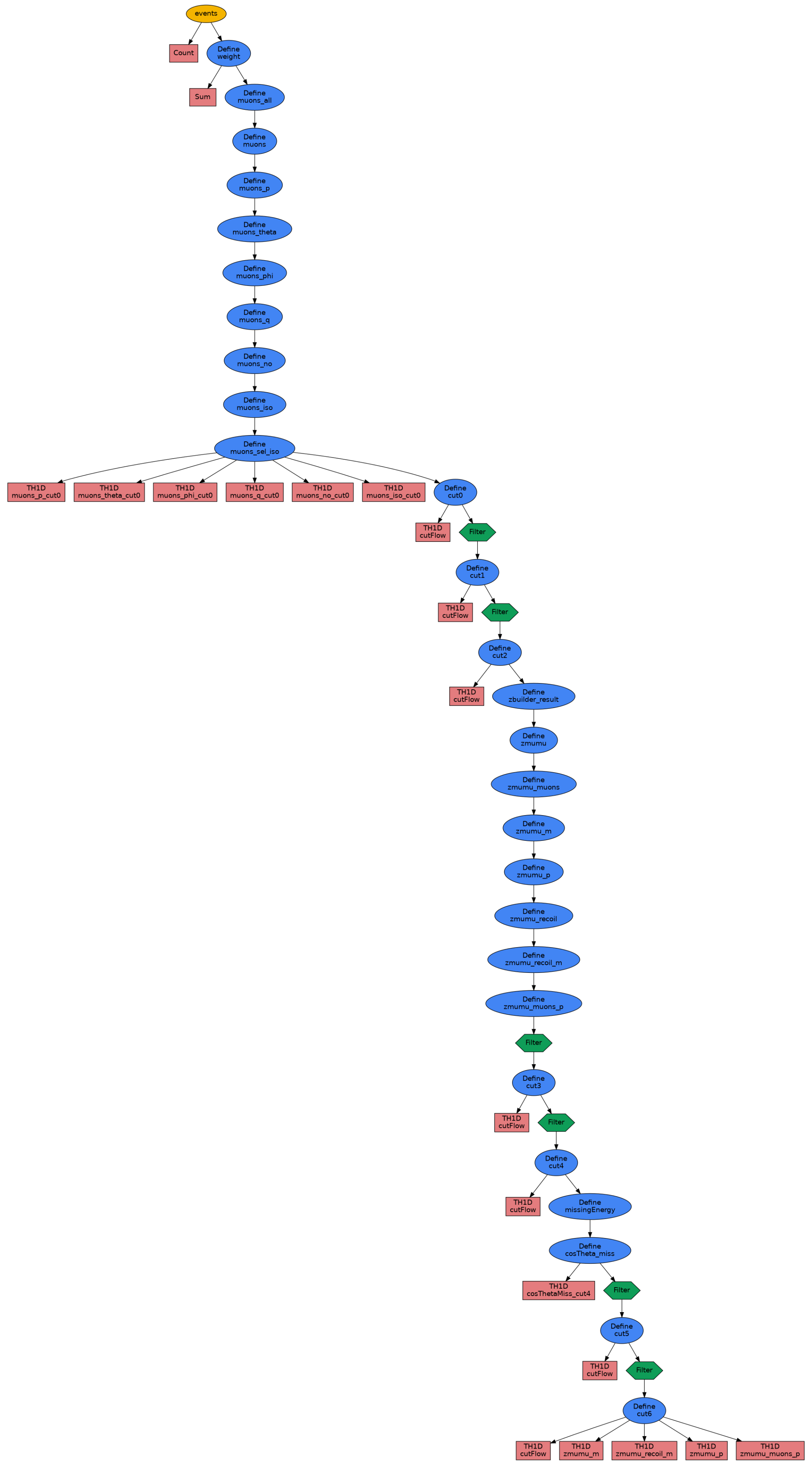
Analysis can be imagined as a graph composed out of building blocks



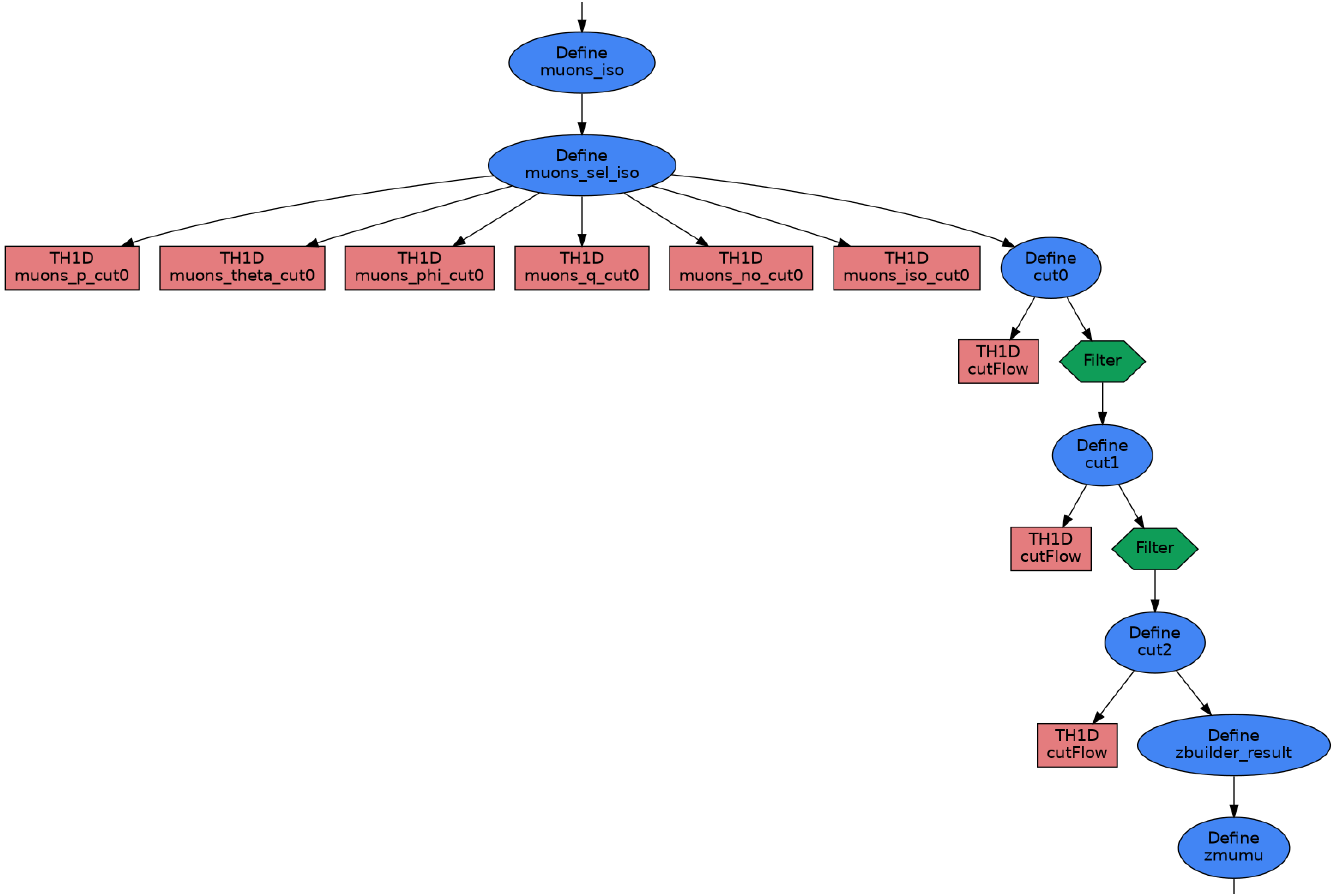
To generate graph of your analysis:

```
fccanalyses run analysis_script.py --graph
```

Analysis as a graph



Analysis as a graph



Integration with Existing Tools

- Boundary between reconstruction and analysis blurred
  - Especially for full-sim
  - Sometimes it is more advantageous to use Gaudi Algorithm [Juan's talk]
- Many tools/libraries created over the years
  - Most are integrated into the Key4hep stack
- RDataFrame C++ based, integrated into Python
- Available libraries:
  - ROOT — together with RDataFrame
  - ACTS — track reconstruction tools
  - ONNX — neural network exchange format
  - FastJet — jet finding package
  - DD4hep — detector description
  - Delphes — fast simulation

Distribution

Latest release of FCCAnalyses is `v0.8.0`

How to get FCCAnalyses:

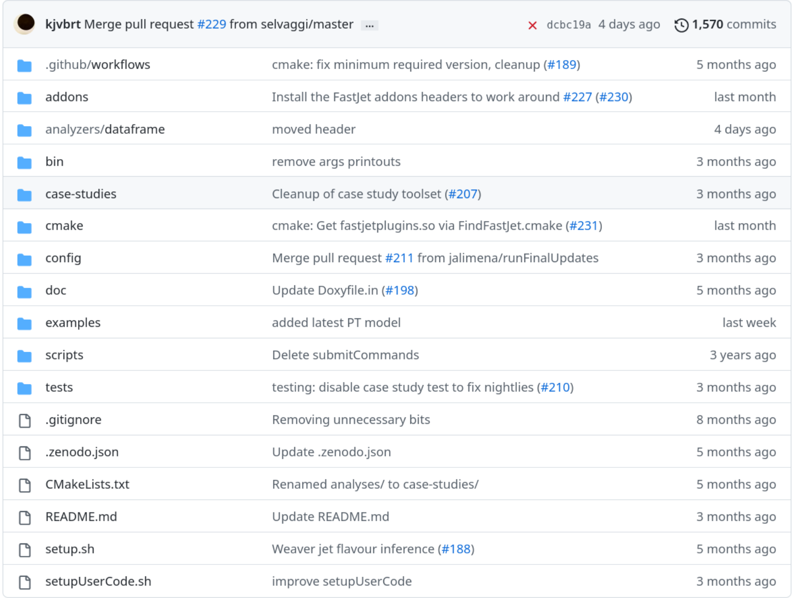
- As a package in the stable/nightlies Key4hep stack
  - Allows to quickly put together small analysis
  - `fccanalysis run ana_script.py` + `analysers.h`
- As a package in the nightlies Key4hep stack
- By checking out `master` branch
  - Allows greater customization
  - Requires discipline
  - Hint: Keep your master in sync with upstream (use rebase or merge)
  - Main branch `master` → `main`
  - Main development branch should be always buildable
- Build in spack / key4hep-spack (~200 packages, ~2h)
- CVMFS + Docker/Podman

Key4hep platforms: CentOS 7, AlmaLinux 9, Ubuntu 22.04

Ecosystem

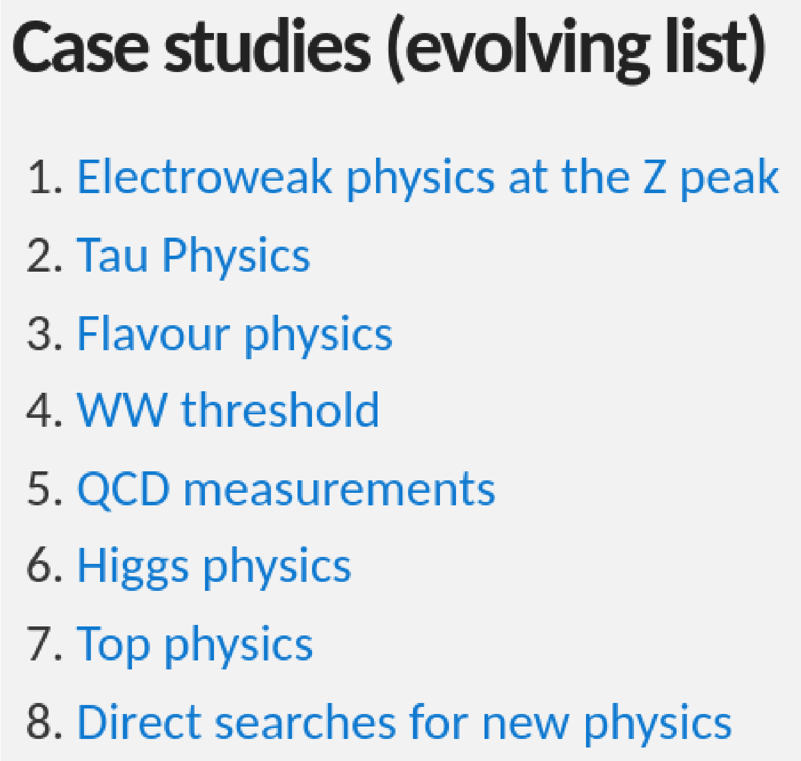
Analysis spread through two repositories:

- FCCAnalyses
  - Repository of common tools and algorithms
  - General analysis code in analyzers
  - Steering of the analysis (RDataFrame)
  - Access to the dataset (meta)data
  - Running over large datasets / on batch
  - Experimetal machinery for case studies
- FCCeePhysicsPerformance
  - Main place for the abstracts
  - Contains very specific analysis code
    - Or prototypes of tools of common interest to be eventually moved to FCCAnalysis
  - (Proto)package repository



kjvbrt Merge pull request #229 from selvaggi/master ... ✕ dcbc19a 4 days ago 1,570 commits

.github/workflows	cmake: fix minimum required version, cleanup (#189)	5 months ago
addons	Install the FastJet addons headers to work around #227 (#230)	last month
analyzers/dataframe	moved header	4 days ago
bin	remove args printouts	3 months ago
case-studies	Cleanup of case study toolset (#207)	3 months ago
cmake	cmake: Get fastjetplugins.so via FindFastJet.cmake (#231)	last month
config	Merge pull request #211 from jalimena/runFinalUpdates	3 months ago
doc	Update Doxyfile.in (#198)	5 months ago
examples	added latest PT model	last week
scripts	Delete submitCommands	3 years ago
tests	testing: disable case study test to fix nightlies (#210)	3 months ago
.gitignore	Removing unnecessary bits	8 months ago
.zenodo.json	Update .zenodo.json	5 months ago
CMakeLists.txt	Renamed analyses/ to case-studies/	5 months ago
README.md	Update README.md	3 months ago
setup.sh	Weaver jet flavour inference (#188)	5 months ago
setupUserCode.sh	improve setupUserCode	3 months ago



Case studies (evolving list)

1. Electroweak physics at the Z peak
2. Tau Physics
3. Flavour physics
4. WW threshold
5. QCD measurements
6. Higgs physics
7. Top physics
8. Direct searches for new physics

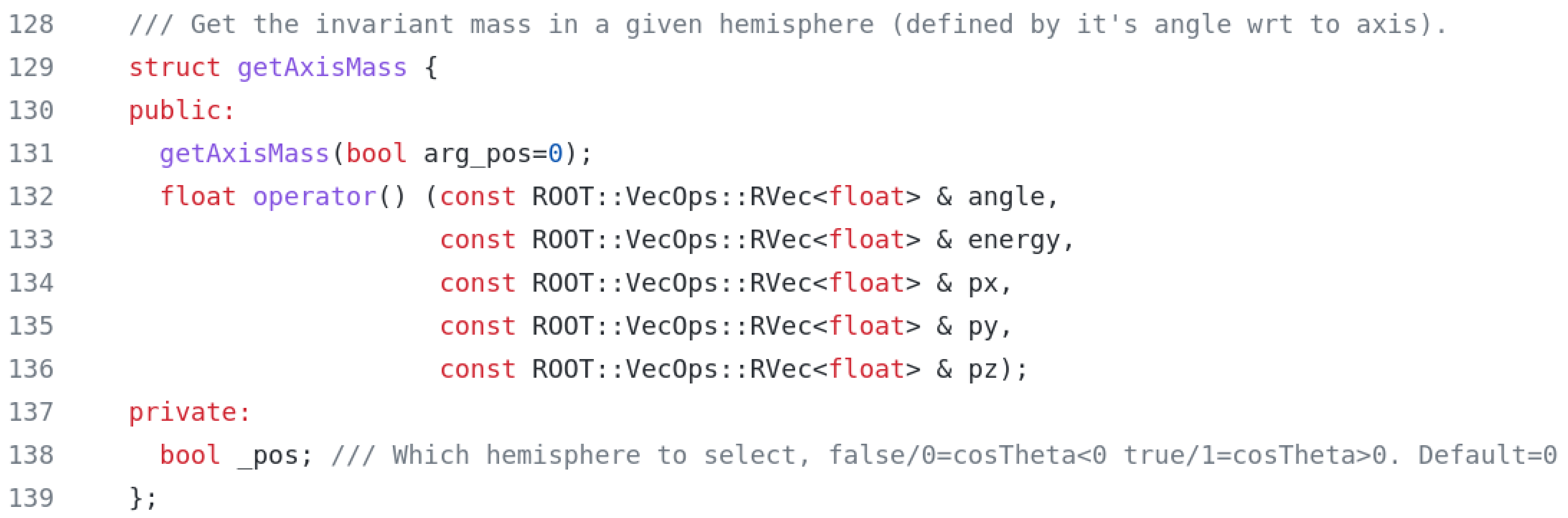
Analysis Architecture I.

One can write and run the analysis in several ways

- Managed mode: `fccanalysis run my_ana.py`
  - The RDataFrame frame is managed by the framework
  - User provides Python analysis script with compulsory attributes
  - Libraries are loaded automatically
  - Dataset metadata are loaded from remote location — CVMFS/HTTP server
  - Batch submission on HTCondor
  - Customization: Possible at the level of analyzer functions
  - Intend for: Quick analysis, no advanced analyzer functions

Writing an analyzer function

- Analyzer function is a C++ function or struct
- Typically and analyzer is a `struct` which operates on an EDM4hep object
- Optional dependencies for analyzers can be: FastJet, DD4hep, ACTS and ONNX
- ROOT RDataFrame needs to be aware of the analyzer function
  - Provided as a string
  - Loaded and JITed by the `ROOT.gInterpreter`
  - Compiled in the library



```cpp
/// Get the invariant mass in a given hemisphere (defined by it's angle wrt to axis).
struct getAxisMass {
public:
  getAxisMass(bool arg_pos=0);
  float operator() (const ROOT::VecOps::RVec<float> & angle,
                    const ROOT::VecOps::RVec<float> & energy,
                    const ROOT::VecOps::RVec<float> & px,
                    const ROOT::VecOps::RVec<float> & py,
                    const ROOT::VecOps::RVec<float> & pz);
private:
  bool _pos; /// Which hemisphere to select, false/0=cosTheta<0 true/1=cosTheta>0. Default=0
};
```

Workflow

- The complete analysis in managed mode is divided into three steps (example):
  - `analysis_stage1.py`, ... — pre-selection stages, analysis dependent, usually runs on batch
  - `analysis_final.py` — final selection, produces final variables
  - `analysis_plots.py` — produces plots from histograms/TTrees
- or into two with the help of Histmaker (example):
  - The pre-selection stages and final stage are combined together
  - Plotting step
- Disclaimer: Plotting facilities are rudimentary, improvements are welcome :)

Analysis Architecture II.

One can write and run an analysis in several ways

- Standalone mode: `python my_ana.py`
  - The RDataFrame frame is managed by the user
  - Can leverage the FCCAnalyses library of analyzer functions
  - The analysis can be written as a Python script or C++ program
  - Loading of the libraries is handled by the user
  - Dataset metadata have to be handled manually
  - Batch submission is not provided
  - Customization: Creation and steering of the RDataFrame
  - Intended for: Advanced users
- Ntupleizer style:
  - Intend is to create flat trees and continue without the frameworks help

Improvements

Making FCCAnalyses more robust framework

- Global reorganization of the internal structure --- fccanalysis + sub-commands
- Synchronized logging functionality across the whole framework (Python + RDataFrame + ROOT)
- Created man pages (terminal + web)
- General safety and robustness
- Reviving FCCAnalyses package in Key4hep stack
- Testing the whole analysis chain

fccanalysis sub-commands

```
[jsmiesko@death-machine FCCAnalyses (master =)]$ fccanalysis --help

...

sub-commands:
  sub-command           one of the available sub-commands
    init                generate a RDataFrame based FCC analysis
    build               build and install local analysis
    test                test whole or a part of the analysis framework
    pin                 pin fccanalyses to the current version of Key4hep stack
    run                 run a RDataFrame based FCC analysis
    final               run a RDataFrame based FCC analysis final configuration
    plots               run a RDataFrame based FCC analysis plot configuration
```

Logging functionality

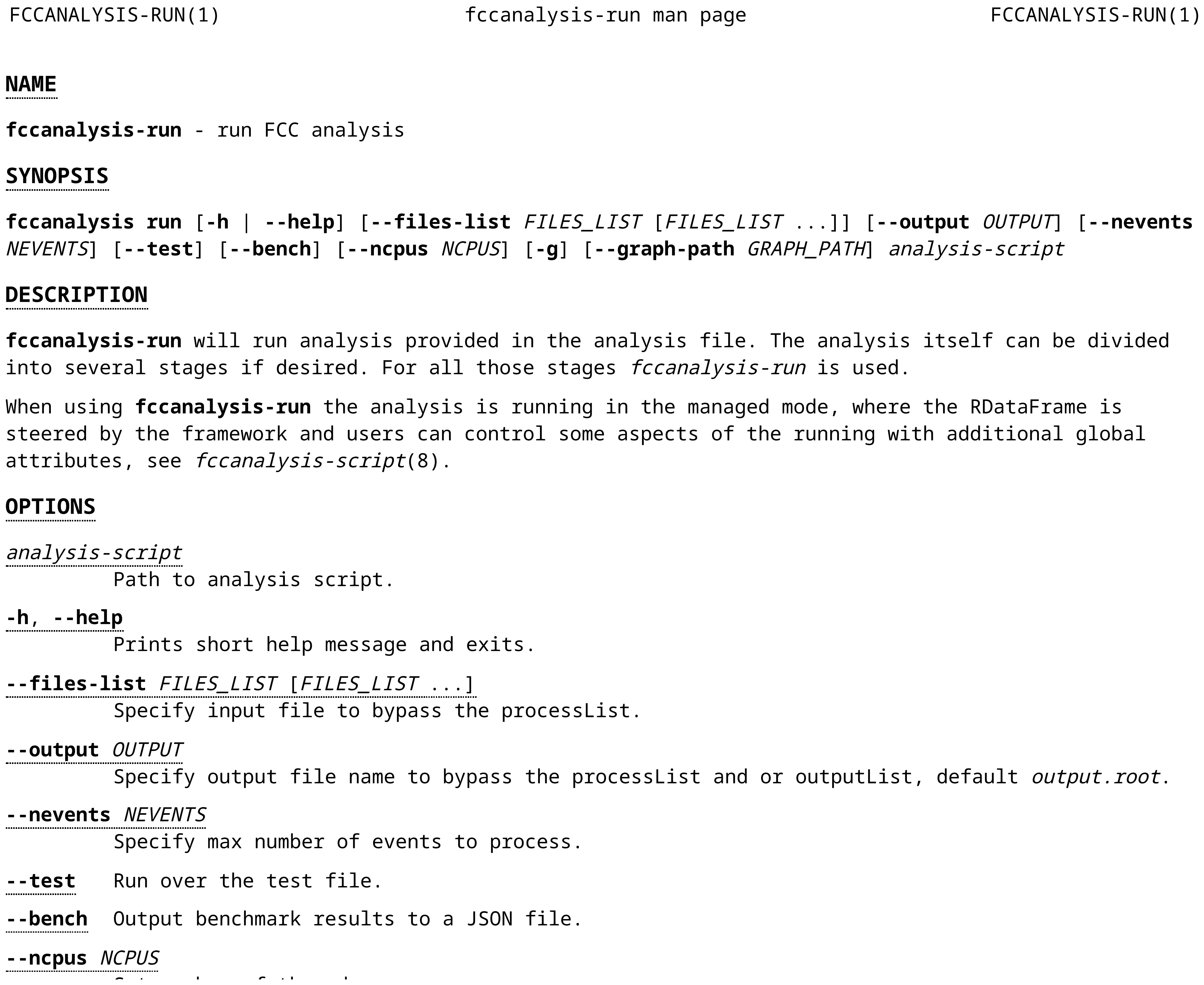
Select verbosity level:

```
[jsmiesko@death-machine FCCAnalyses (master =)]$ fccanalysis --help
usage: fccanalysis [-h] [-v | -vv | -vvv] sub-command ...

FCCAnalyses v0.8.0

options:
  -h, --help            show this help message and exit
  -v, --verbose         make output verbose
  -vv, --more-verbose   make output more verbose
  -vvv, --most-verbose  make output even more verbose
```

Get something out of the analyzer:

```
#include "RLogger.hxx"
R__LOG_INFO(ROOT.Detail.RDF.RDFLogChannel(), "Info message")
```

Manual pages



FCCANALYSIS-RUN(1) fccanalysis-run man page FCCANALYSIS-RUN(1)

NAME

fccanalysis-run - run FCC analysis

SYNOPSIS

fccanalysis run [**-h** | **--help**] [**--files-list** *FILES_LIST* [*FILES_LIST* ...]] [**--output** *OUTPUT*] [**--nevents** *NEVENTS*] [**--test**] [**--bench**] [**--ncpus** *NCPUS*] [**-g**] [**--graph-path** *GRAPH_PATH*] *analysis-script*

DESCRIPTION

fccanalysis-run will run analysis provided in the analysis file. The analysis itself can be divided into several stages if desired. For all those stages *fccanalysis-run* is used.

When using **fccanalysis-run** the analysis is running in the managed mode, where the RDataFrame is steered by the framework and users can control some aspects of the running with additional global attributes, see *fccanalysis-script*(8).

OPTIONS

analysis-script
    Path to analysis script.

-h, **--help**
    Prints short help message and exits.

--files-list *FILES_LIST* [*FILES_LIST* ...]
    Specify input file to bypass the processList.

--output *OUTPUT*
    Specify output file name to bypass the processList and or outputList, default *output.root*.

--nevents *NEVENTS*
    Specify max number of events to process.

--test Run over the test file.

--bench Output benchmark results to a JSON file.

--ncpus *NCPUS*
    Set number of threads

Testing FCCAnalyses package

```
source "${FCCTESTS_STACK}"

RNDMSTR="$(sed 's/[-]//g' < /proc/sys/kernel/random/uuid | head -c 12)"
WORKDIR="${FCCTESTS_TMPDIR}/fccanalyses-stack-full-analysis-${RNDMSTR}"

mkdir -p "${WORKDIR}" || exit 1
cd "${WORKDIR}" || exit 1

fccanalysis run ${FCCANALYSES}/../share/examples/examples/FCCee/higgs/mH-recoil/mumu/analysis_stage1.py || exit 1
fccanalysis run ${FCCANALYSES}/../share/examples/examples/FCCee/higgs/mH-recoil/mumu/analysis_stage2.py || exit 1
fccanalysis final ${FCCANALYSES}/../share/examples/examples/FCCee/higgs/mH-recoil/mumu/analysis_final.py || exit 1
fccanalysis plots ${FCCANALYSES}/../share/examples/examples/FCCee/higgs/mH-recoil/mumu/analysis_plots.py
```

- Multiple FCCAnalyses tests running every morning
  - Plan to integrate them into Key4hep validation

Work in Progress

Unlocking full potential of ROOT RDataFrame and EDM4hep inside FCCAnalyses framework

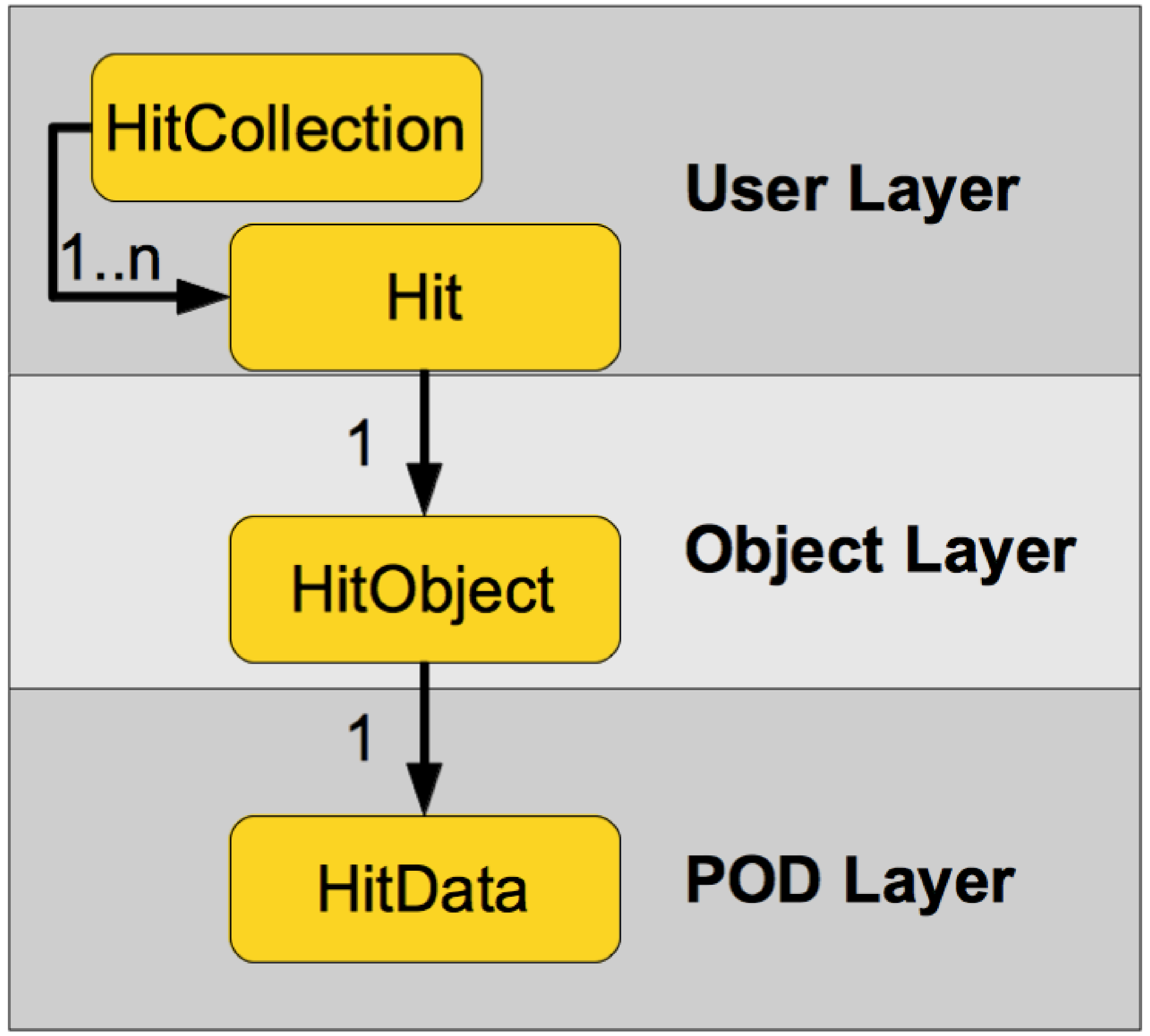
- EDM4hep RDataSource
- Sample metadata (Dirac)
- Import FCCAnalyses / analysis compartmentalization
- Standard library of the analyzers
- Slow or fast decay of the EDM4hep in the analysis
- Test / Validation facilities
- Event visualization

EDM4hep RDataSource

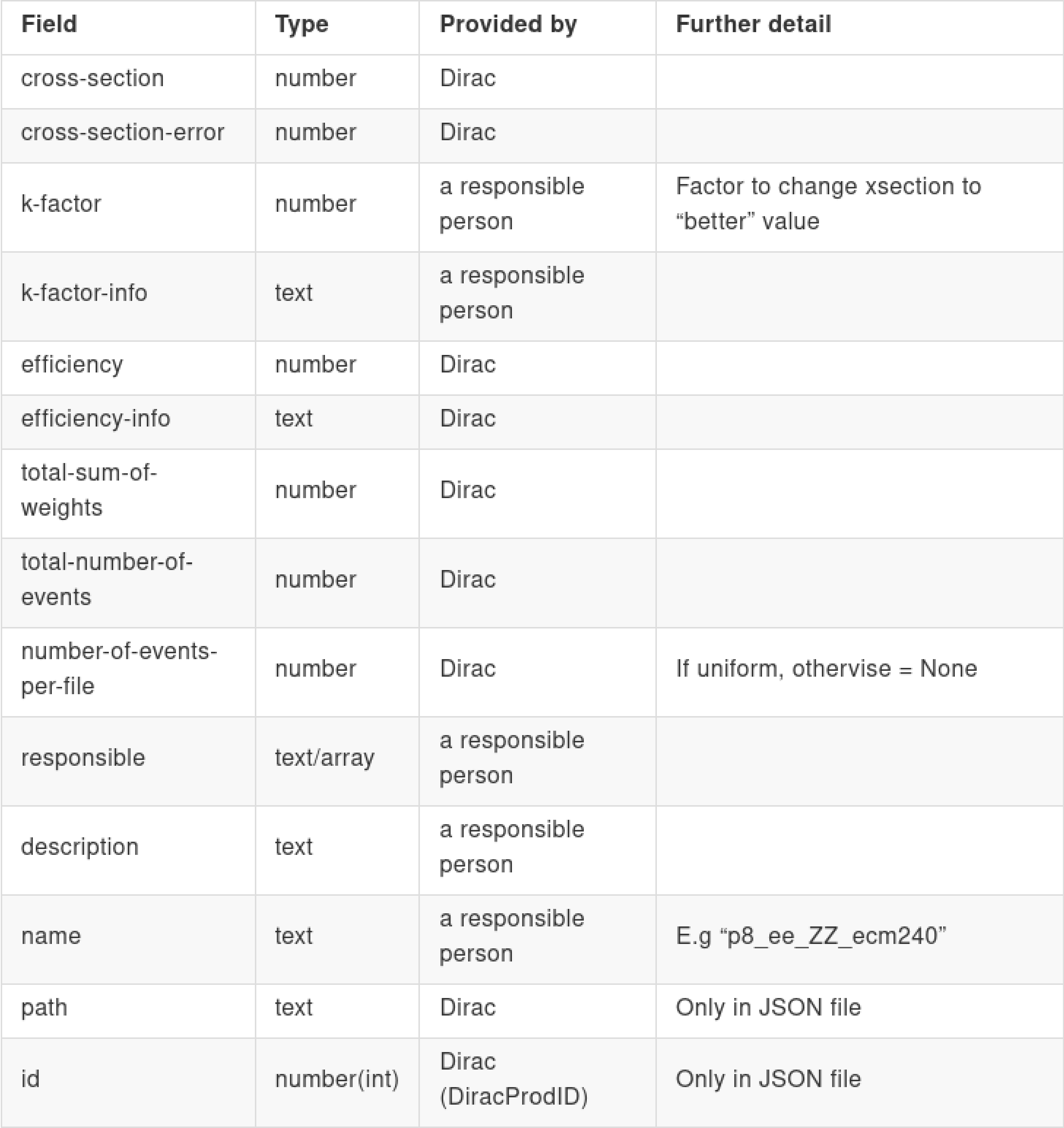
Preserving EDM4hep Associations in RDataFrame

- Podio/EDM4hep has several layers
- Highest layer provides associations
- Having associations greatly improves writing/understanding of the analyzers
- Majority of the analyzers needs adjustment

For more details about EDM4hep, see Thomas' talk



Sample metadata



Field	Type	Provided by	Further detail
cross-section	number	Dirac	
cross-section-error	number	Dirac	
k-factor	number	a responsible person	Factor to change xsection to “better” value
k-factor-info	text	a responsible person	
efficiency	number	Dirac	
efficiency-info	text	Dirac	
total-sum-of-weights	number	Dirac	
total-number-of-events	number	Dirac	
number-of-events-per-file	number	Dirac	If uniform, othervise = None
responsible	text/array	a responsible person	
description	text	a responsible person	
name	text	a responsible person	E.g “p8_ee_ZZ_ecm240”
path	text	Dirac	Only in JSON file
id	number(int)	Dirac (DiracProdID)	Only in JSON file

Sample output path example

`/eos/experiment/fcc/prod/fcc/ee/winter2023/91.19gev/Zbb/idea/delphes/00012345/`

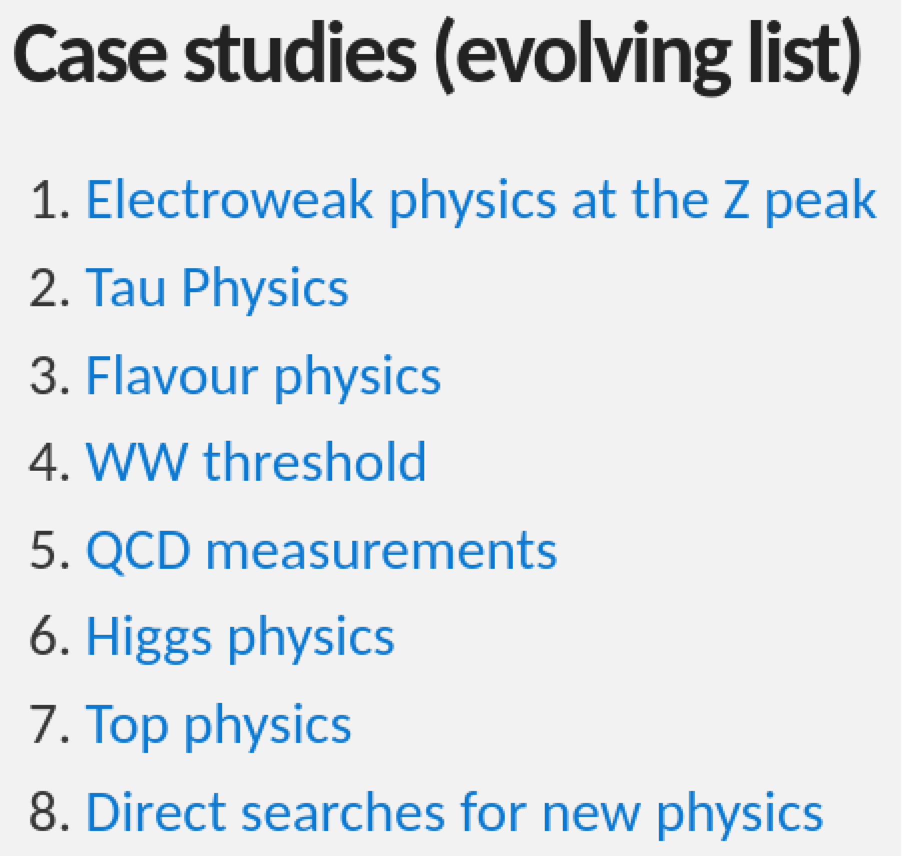
See also Lorenzo's talk

import FCCAnalyses

```python
import FCCAnalyses
import ROOT

ROOT.gROOT.SetBatch(True)

def main():
    '''
    Example analysis entry point
    '''

    sample = FCCAnalyses.Sample('p8_ee_WW_ecm240')

    FCCAnalyses.register_analyzers('examples/FCCee/import/AddAnalyzers.h')

    dframe = fccana.get_dataframe(sample)
    dframe2 = dframe.Define("particles", "gen_particles()")
    dframe3 = dframe2.Define("particles_pt", "MCParticle::get_pt(particles)")
    hist = dframe3.Histo1D("particles_pt")
    hist.Print()

    canvas = ROOT.TCanvas("canvas", "", 450, 450)
    hist.Draw()
    canvas.Print('test.pdf')

if __name__ == '__main__':
    main()
```

FCCAnalyses library

- Vertexing
- ACTS vertex finder
- Event variables
- Calorimeter hit/cluster variables
- Reconstructed/MC particle operations
- Flavour tagging
- Jet clustering/constituents

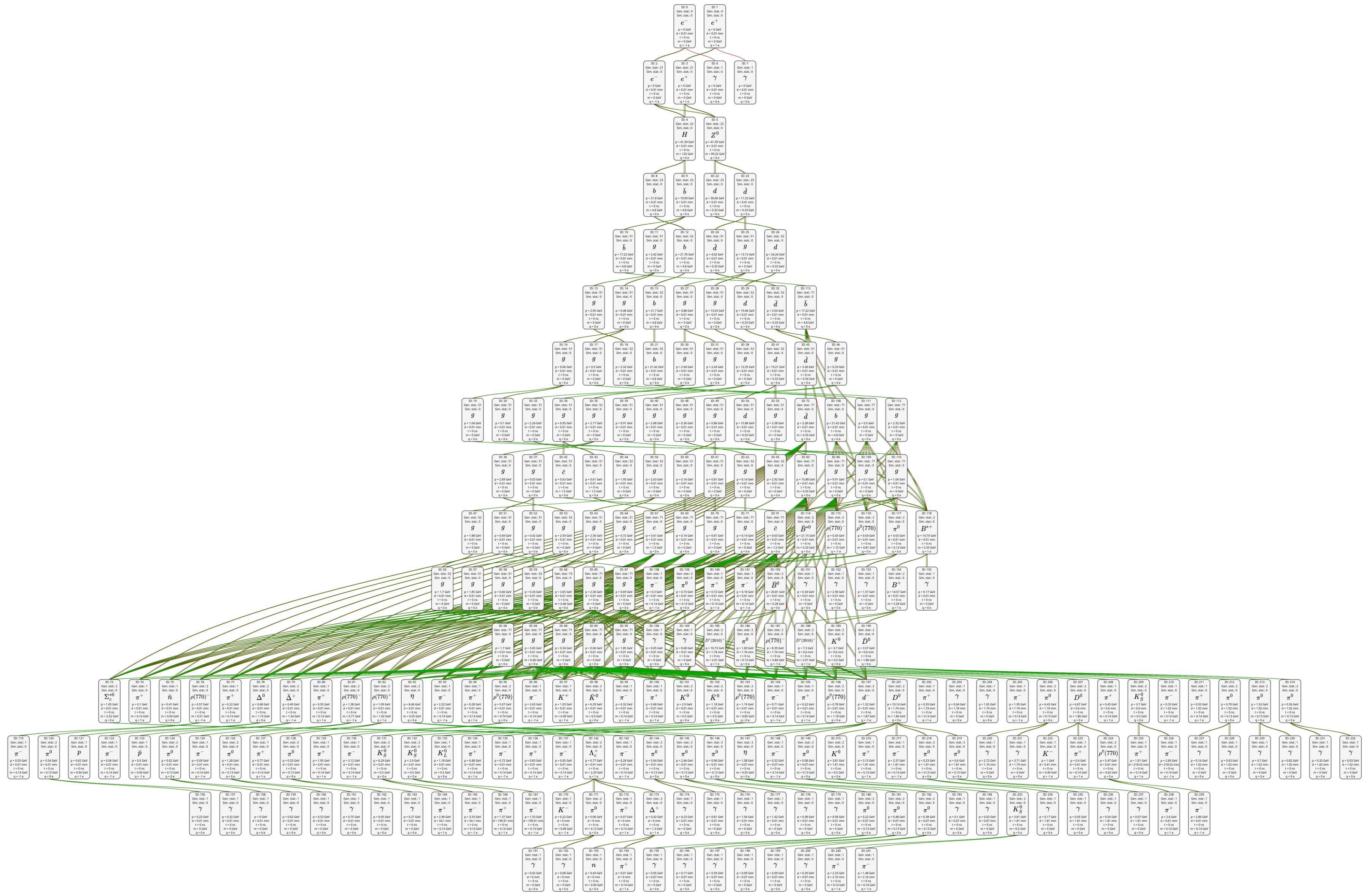


Case studies (evolving list)

1. Electroweak physics at the Z peak
2. Tau Physics
3. Flavour physics
4. WW threshold
5. QCD measurements
6. Higgs physics
7. Top physics
8. Direct searches for new physics

Event visualization

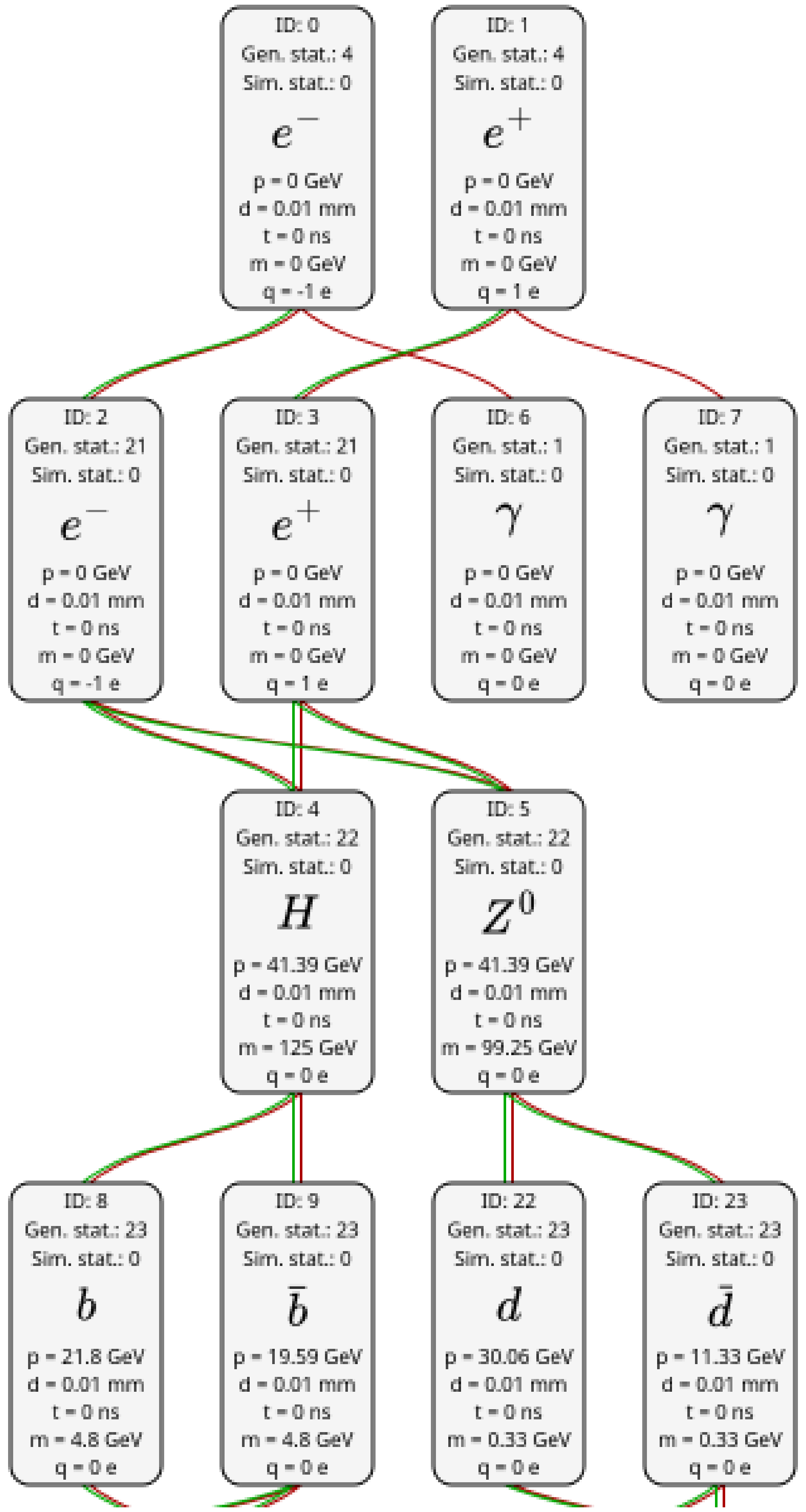
Pythia 8 | ee → ZH @ 240 GeV



To visualize your MC Particle tree, do:

- `source /cvmfs/sw.hsf.org/key4hep/setup.sh`
- `edm4hep2json -l Particle -n 10 /eos/experiment/fcc/ee/generation/DelphesEvents/winter2023/IDEA/p8_ee_WW_ecm240/events_059793334.root -o p8_ee_WW_ecm240.json`
- visit: https://key4hep.github.io/dmx/ and upload your `.json` file

Event visualization



Pythia 8 | ee → ZH @ 240 GeV

Documentation

There are several sources of documentation

- FCC Tutorials: https://hep-fcc.github.io/fcc-tutorials/
  - Focused on providing a tutorial on a specific topic
- Code reference: https://hep-fcc.github.io/FCCAnalyses/doc/latest/index.html
  - Provides details about implementation of individual analyzers
- Manual pages:
  - Info about commands directly in the terminal: `man fccanalysis`
- FCCAnalyses website, FCCSW website

Conclusions & Outlook

- The combination of EDM4hep and RDataFrame works
  - Possibility to integrate range of existing libraries
  - ML integration need more thought
- Writing an analysis without compilation prefered
- Started focusing on the full simulation detector studies
  - Access to the detector description through the framework
- Bi-weekly meeting focused on framework development, but more importantly on the analysis development
  - First meeting: 7 Feb 2024, 11:00 AM

See you at the meeting!

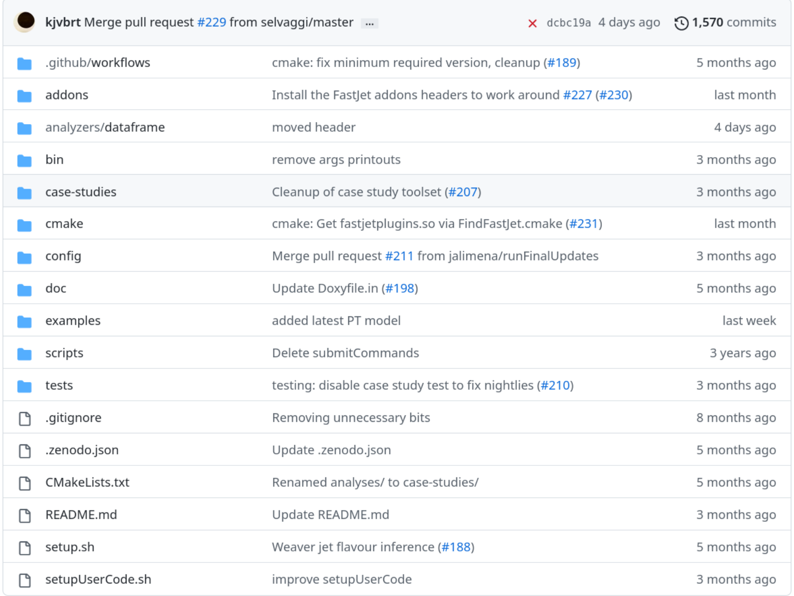
Babyface from Toy Story, Pixar

Backup

Ecosystem

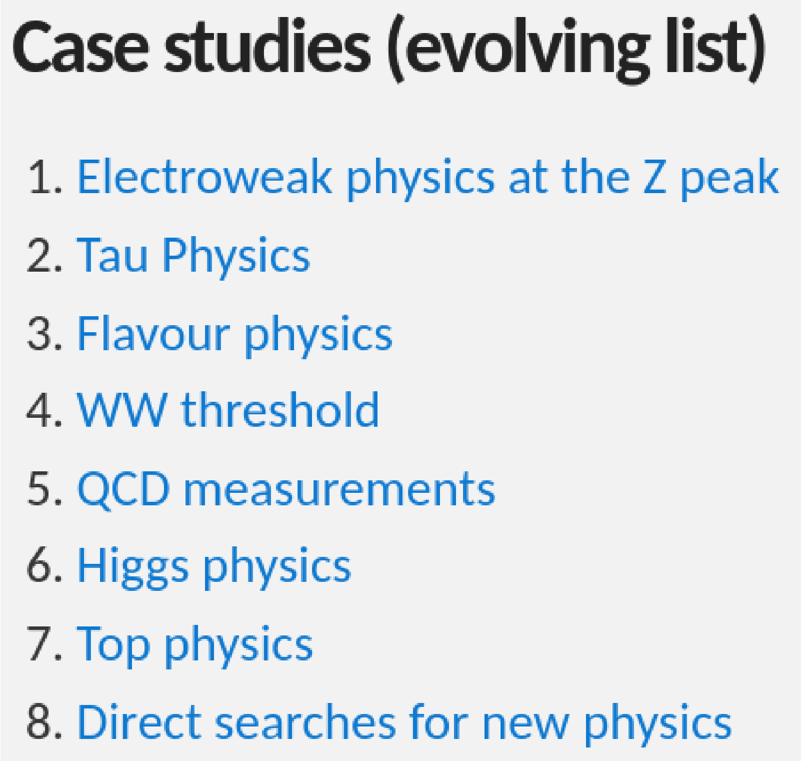
Analysis spread through two repositories:

- FCCAnalyses
  - Repository of common tools and algorithms
  - General analysis code in analyzers
  - Steering of the analysis (RDataFrame)
  - Access to the dataset (meta)data
  - Running over large datasets / on batch
  - Experimetal machinery for case studies
- FCCeePhysicsPerformance
  - Main place for the abstracts
  - Contains very specific analysis code
    - Or prototypes of tools of common interest to be eventually moved to FCCAnalysis
  - (Proto)package repository



kjvbrt Merge pull request #229 from selvaggi/master ... ✕ dcbc19a 4 days ago 1,570 commits

.github/workflows	cmake: fix minimum required version, cleanup (#189)	5 months ago
addons	Install the FastJet addons headers to work around #227 (#230)	last month
analyzers/dataframe	moved header	4 days ago
bin	remove args printouts	3 months ago
case-studies	Cleanup of case study toolset (#207)	3 months ago
cmake	cmake: Get fastjetplugins.so via FindFastJet.cmake (#231)	last month
config	Merge pull request #211 from jalimena/runFinalUpdates	3 months ago
doc	Update Doxyfile.in (#198)	5 months ago
examples	added latest PT model	last week
scripts	Delete submitCommands	3 years ago
tests	testing: disable case study test to fix nightlies (#210)	3 months ago
.gitignore	Removing unnecessary bits	8 months ago
.zenodo.json	Update .zenodo.json	5 months ago
CMakeLists.txt	Renamed analyses/ to case-studies/	5 months ago
README.md	Update README.md	3 months ago
setup.sh	Weaver jet flavour inference (#188)	5 months ago
setupUserCode.sh	improve setupUserCode	3 months ago



Case studies (evolving list)

1. Electroweak physics at the Z peak
2. Tau Physics
3. Flavour physics
4. WW threshold
5. QCD measurements
6. Higgs physics
7. Top physics
8. Direct searches for new physics

FCCAnalyses vs. Coffea/Coffea-casa

- Provides similar set of features to FCCAnalyses
- Dataframe in coffea, Orchestration in coffea-casa
- User interface purely pythonic
- Integrated into python package ecosystem
- FCCAnalysis purpose build for FCC
- Integration with SWAN and Dask

FCCAnalyses batch submissions

- FCCAnalyses allows users to submit their jobs onto HTCondor
- It bootstraps itself with use of scripts in subprocesses
- Framework creates two files
  - Shell script with `fccanalysis` command
  - Condor configuration file
- There is also possibility to add user provided Condor parameters
- Condor environment now isolated from machine where the submission was done
- Revised tracking across chunks/stages done with the variable in the ROOT file

Code formatting

- Currently, there is wide range of styles used
- End goal: Make the analyzers better organized
  - They are building blocks of the analysis
- Created CI to check every commit
- LLVM Style selected based on popularity
- Only changed lines are checked

Updated vertexing

- Vertexing done with the help of code from Franco B.
- Introduces dependency on Delphes
- Introduces new analyzers: `SmearedTracksdNdx`, `SmearedTracksTOF`
- Simplifies Delphes–EDM4hep unit gymnastic
- Adds examples for $\mathrm{B_s}$ to $\mathrm{D_s}$ K

Building of FCCAnalyses

- FCCAnalyses is a package in the Key4hep stack
- Advanced users can work directly on their forks
  - Allows to keep the analysis "cutting edge"
  - Requires discipline
- Added helper sub-command: `fccanalysis build`
- Current distribution mechanisms:
  - Using released version in Key4hep stack
  - Separate git repository + stable Key4hep stack
  - Separate git repository + nightlies stack

Key4hep stack pin

- FCCAnalyses is developed on top of Key4hep stack
- Sometimes depends on specific version of the package
- Added helper sub-command: `fccanalysis pin`
- Will pin the analysis to a specific version of the Key4hep stack
  - There is no patch mechanism in the Key4hep stack