Introduction to collaborative services and basf2

What is it and why do we need it

Speaker: Ilya Komarov for Belle II Summer School at BNL
Slides are heritage of StarterKit workshops
(Jake Bennet, Sam Cunliffe, I.K., Umberto Tamponi, Hannah Wakeling, Anže Zupanc)

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY

First things first

How to get a help and where is all the code?

On this workshop we will talk about the software.
Before we start, let's clarify two most important things:

Where to find documentation?

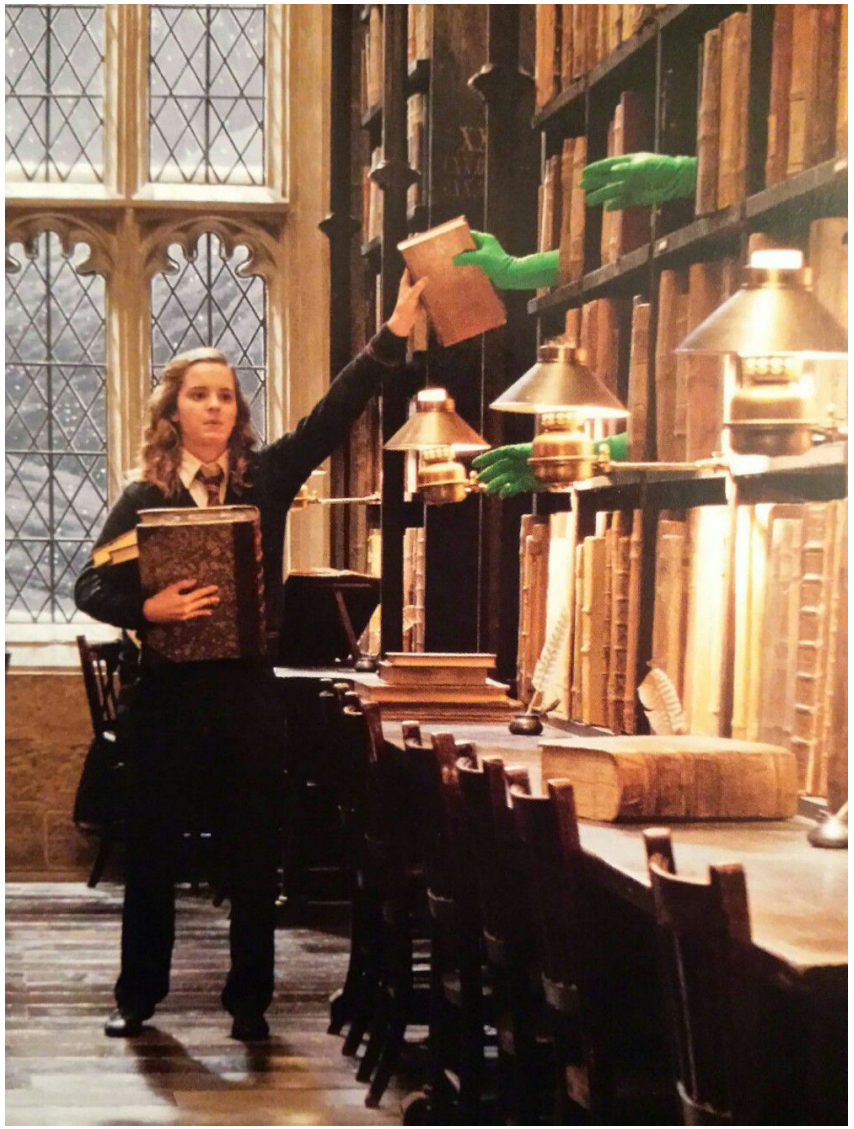
- User-friendly documentation:
  software.belle2.org
- Code itself:
  stash.desy.de
  (Check also tutorials at anaysis/examples/tutorial)

Where to get help?

- Q&A:
  questions.belle2.org
- Chat:
  chat.belle2.org

Tip: try search!

search.belle2.org



Our support is so good that it is almost magical.

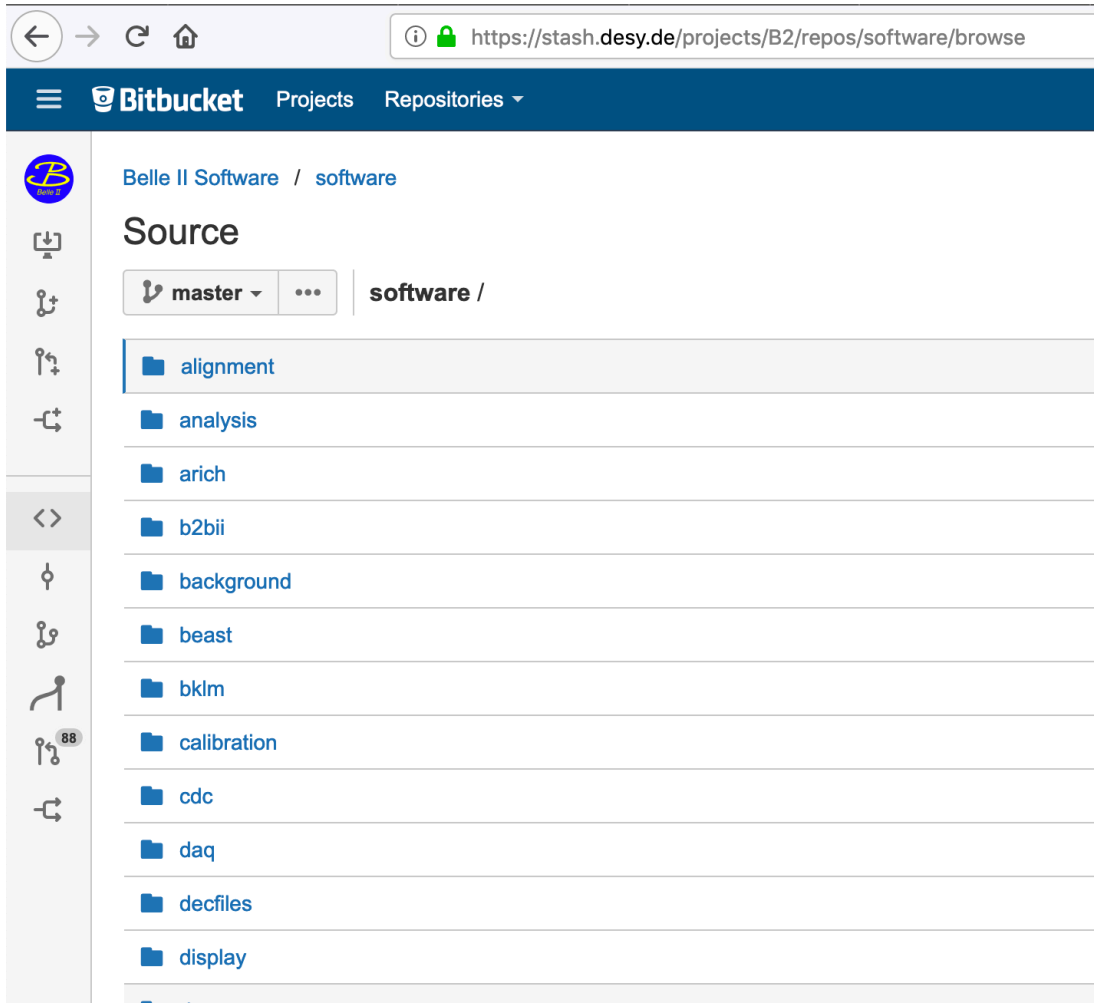
Ok, so what is BASF2?

BASF2 is an acronym standing for

Belle II
Analysis
Software
Framework
2 - do discriminate from BASF
(try guess what is BASF. Hint: it's not chemical giant)

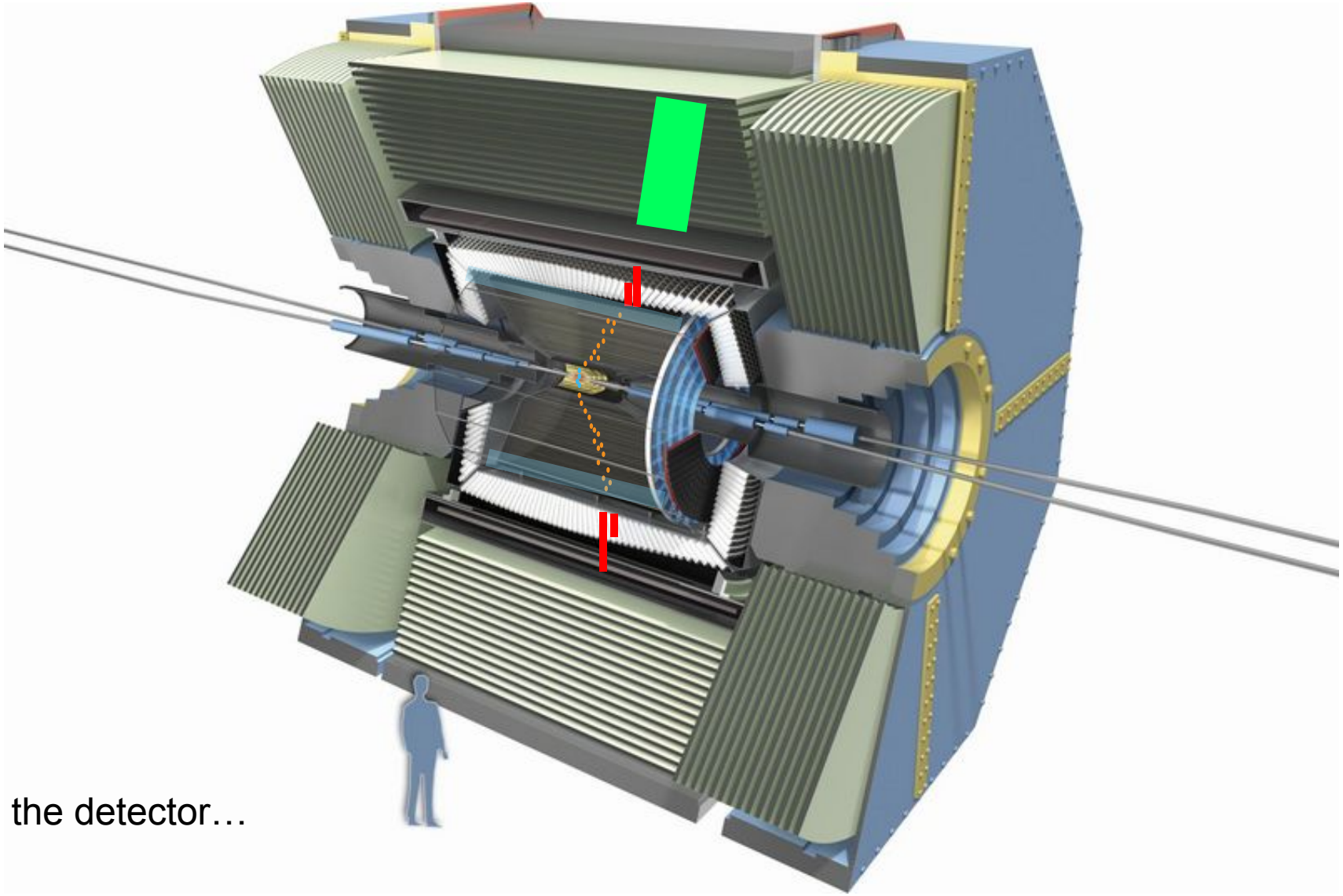
It consists of 41 packages that provide almost all software needs of the experiment: from interpretation of signals from the detector to high-level validation of the performance.

It is written in C++ with python user interfaces.



So what does it do exactly?

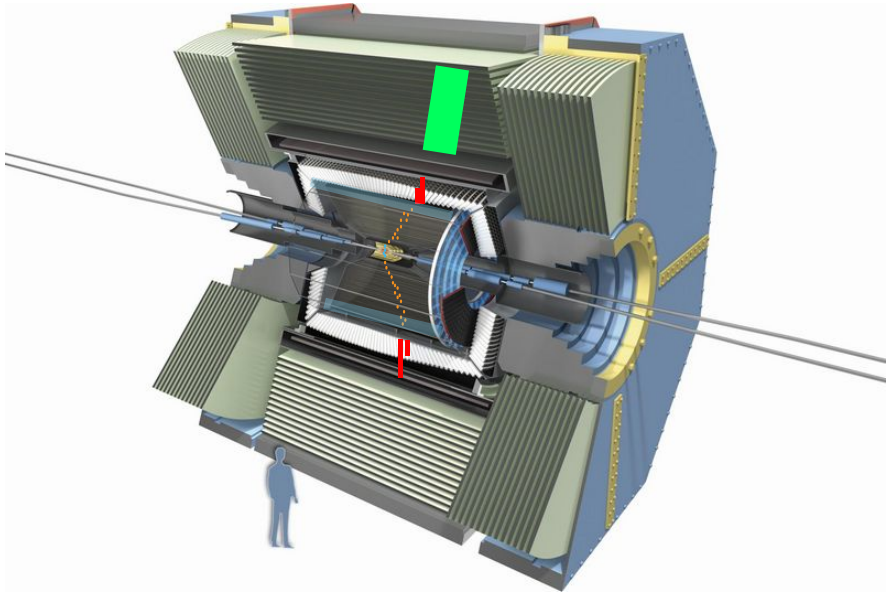
And what will we learn at this workshop?



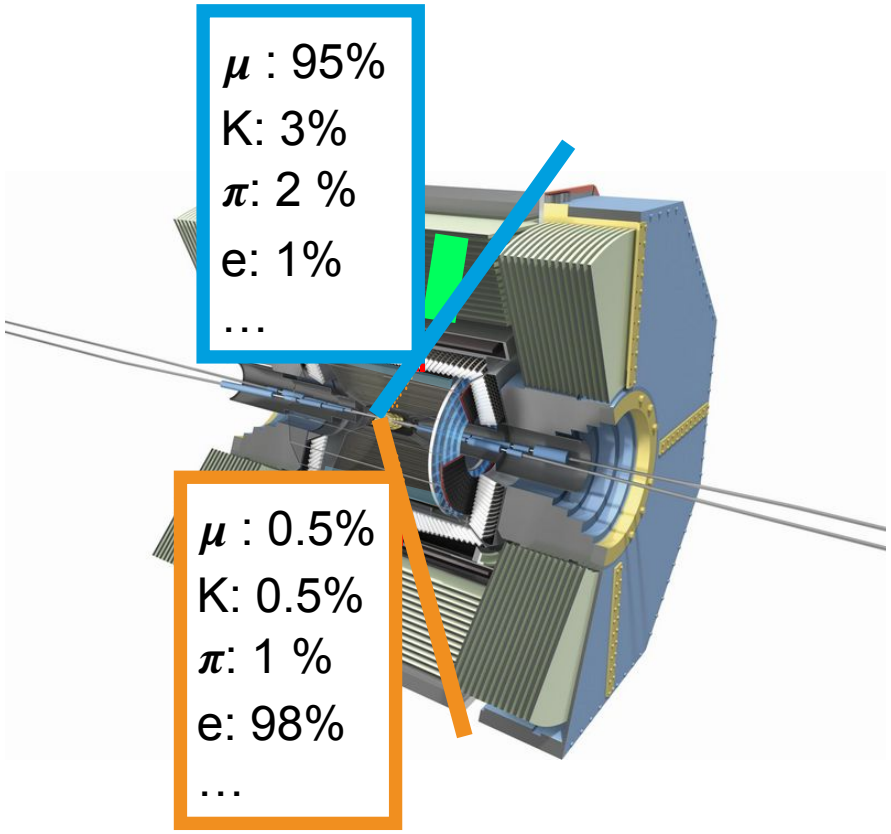
After the collision,
We have signals in the detector…

So what does it do exactly?

And what will we learn at this workshop?



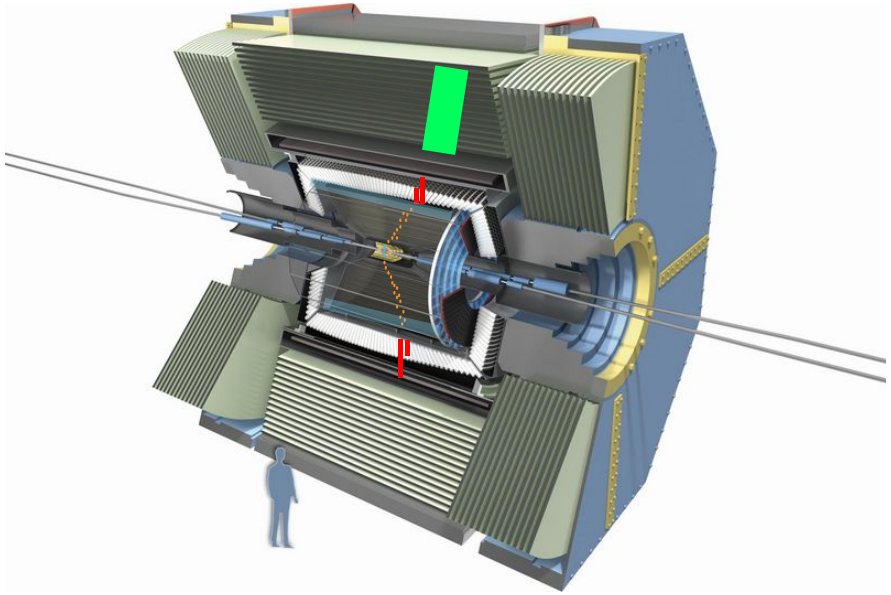
Reconstruction



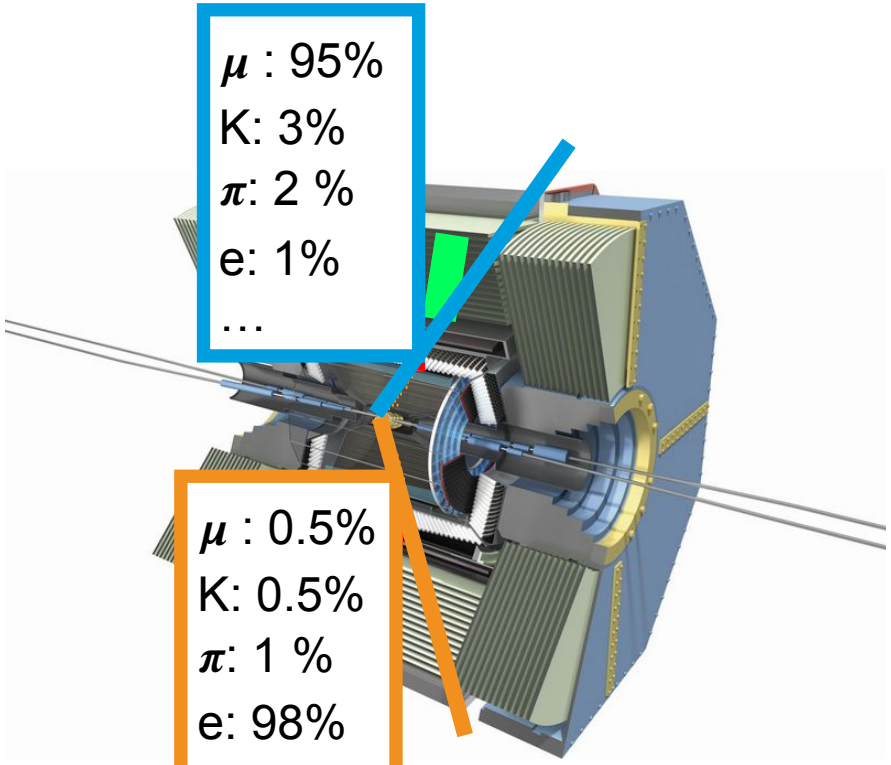
BASF2 reconstructs objects from those signals

So what does it do exactly?

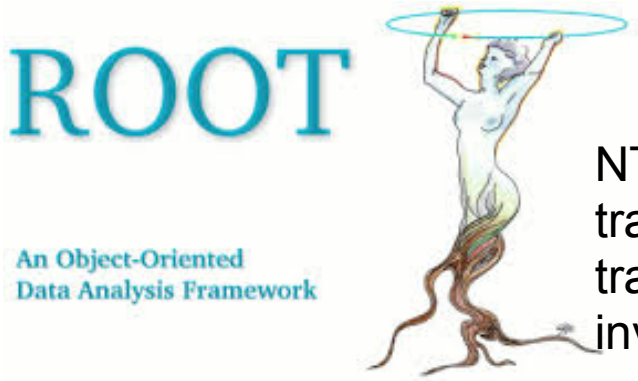
And what will we learn at this workshop?



Reconstruction



Analysis

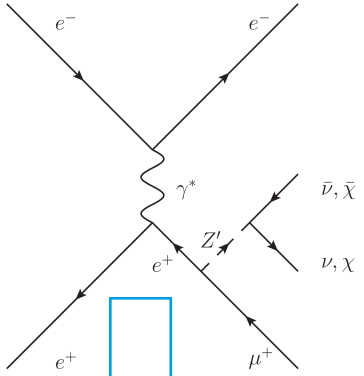


NTuple
track1_p
track2_p
invariant_mass
…

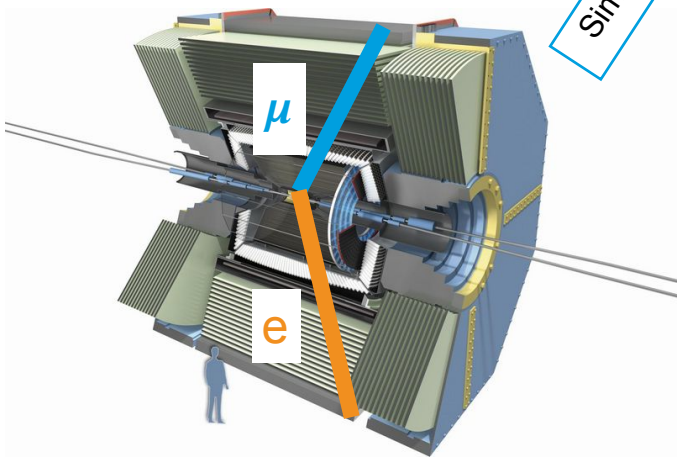
Analysts, using basf2, analyse event and write out ntuple

So what does it do exactly?

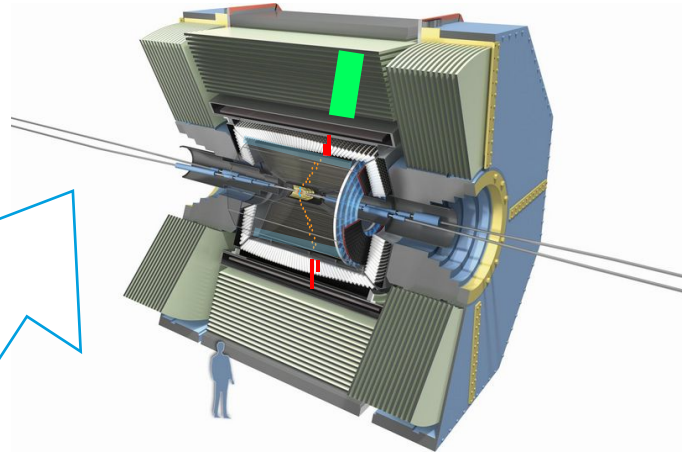
And what will we learn at this workshop?



Generation

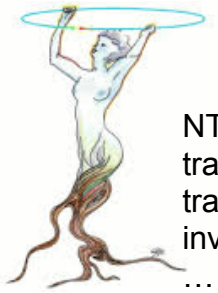


Simulation



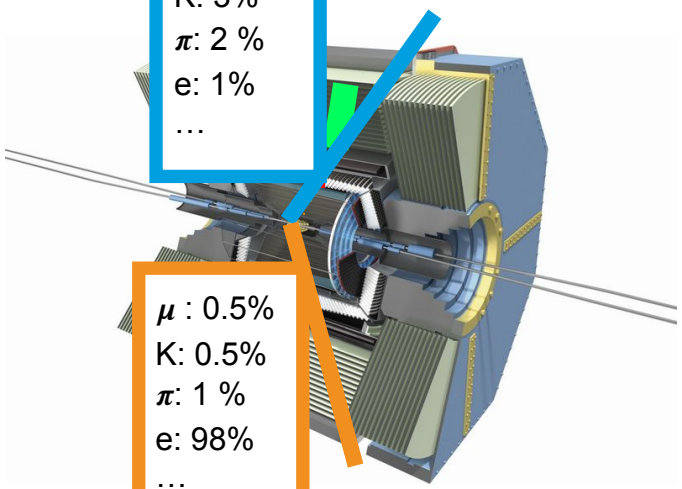
ROOT

An Object-Oriented Data Analysis Framework



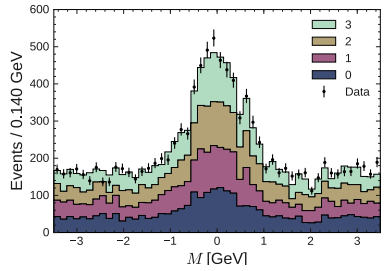
NTuple
track1_p
track2_p
invariant_mass
…

Reconstruction



Analysis

Offline analysis

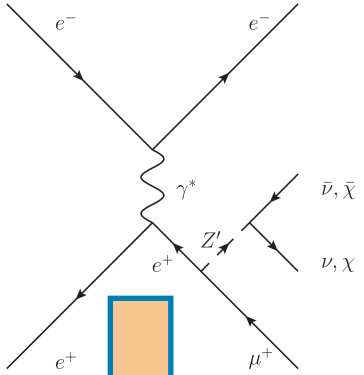


Your result

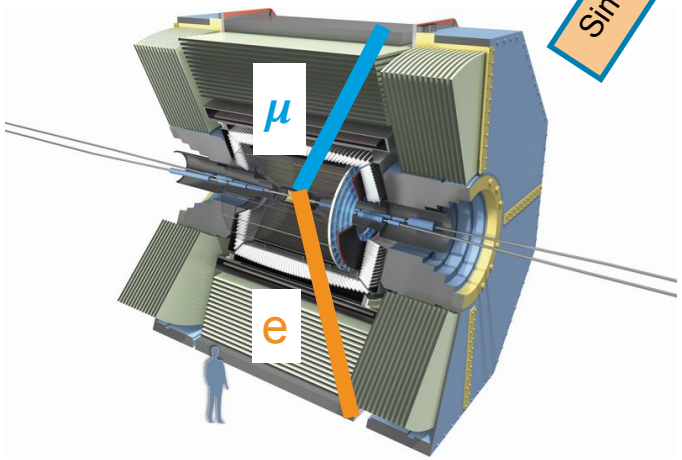
We need to add two extra steps to analyse simulations: generation and simulation

So what does it do exactly?

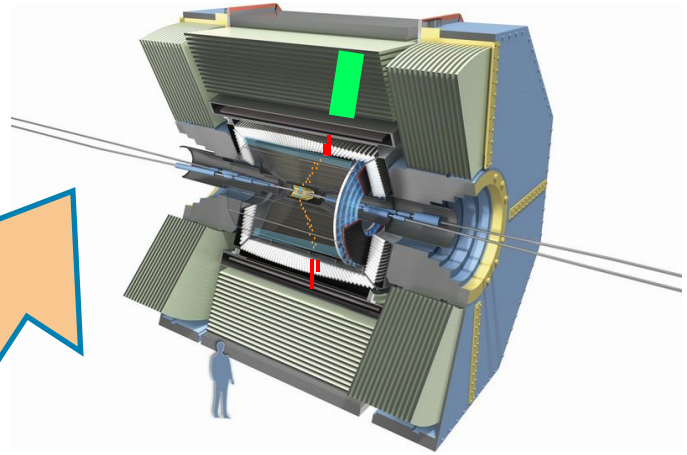
And what will we learn at this workshop?



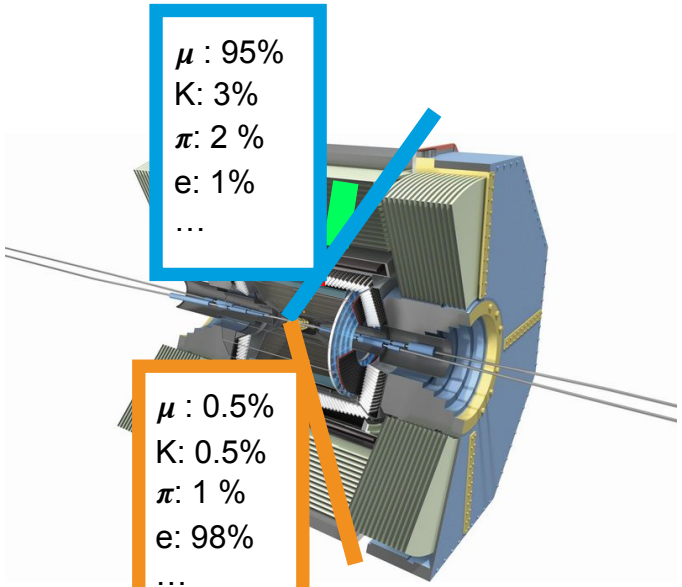
Generation



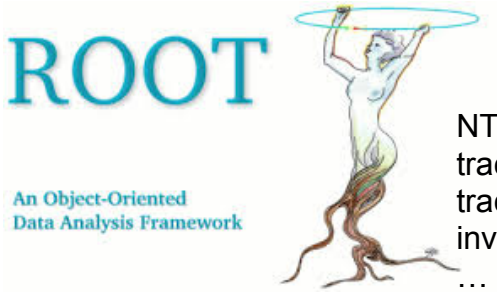
Simulation



Reconstruction

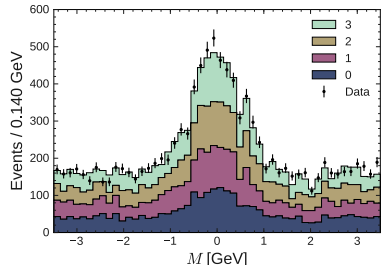


Analysis



NTuple
track1_p
track2_p
invariant_mass
…

Offline analysis

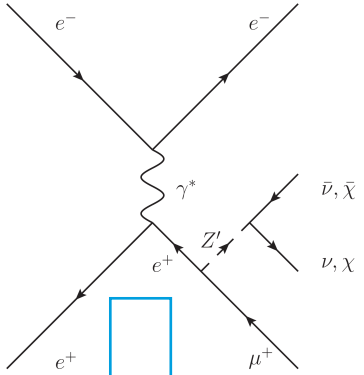


Your result

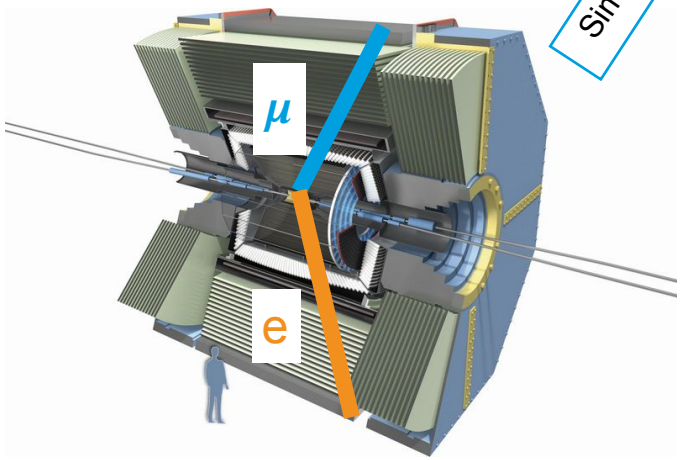
These steps are done by BASF2

So what does it do exactly?

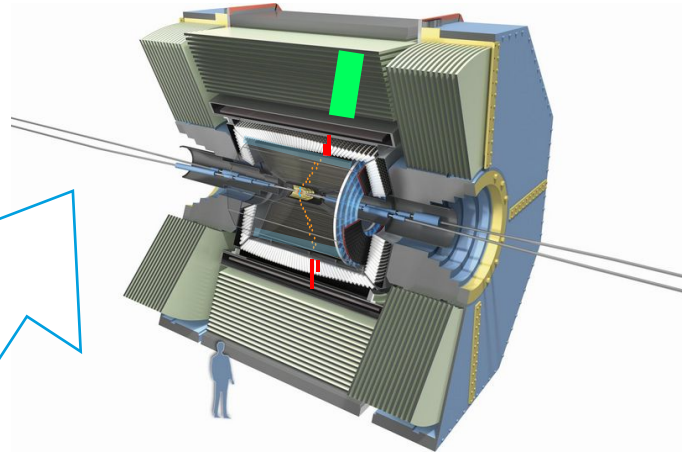
And what will we learn at this workshop?



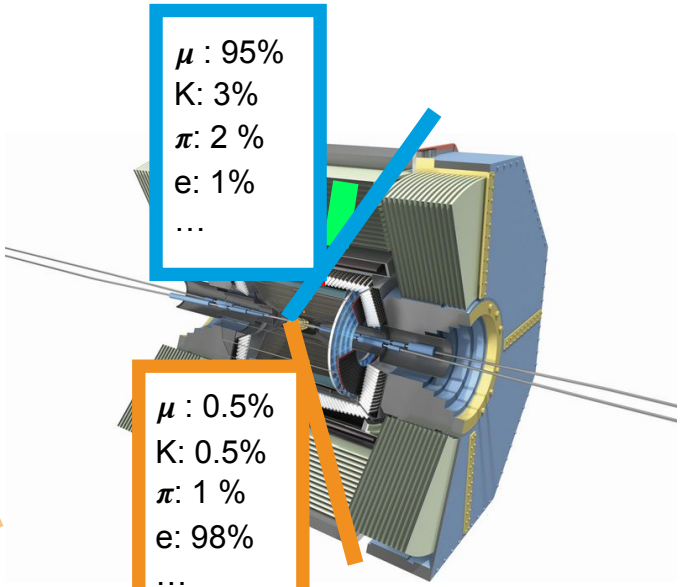
Generation



Simulation



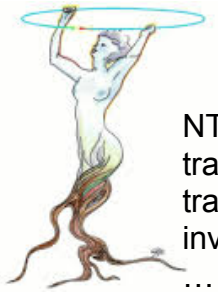
Reconstruction



Analysis

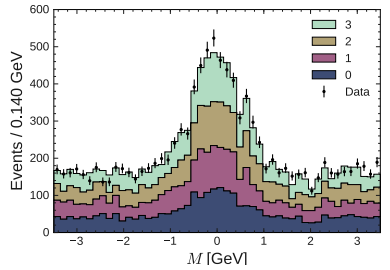
ROOT

An Object-Oriented Data Analysis Framework



NTuple
track1_p
track2_p
invariant_mass
…

Offline analysis



Your result

These steps are usually done by analyst. We will cover them.

The code

`basf2` is C++14 "**under the hood**"

- Packages contain C++ **modules** to manipulate data.

- In analysis: we have code to build **particles** from primitive objects (like tracks and calorimeter clusters).
- We also calculate physics quantities, and apply cuts.

Python 3.6 code for **steering**

- Load and configure C++ modules
  - analysis modules and modules from other packages
- Also python does *some* high-level analysis tasks.

- You will write a fair bit of python during the workshop.

The code

basf2 is C++14 "**under the hood**"

Packages contain C++ **modules** to process data.

In analysis: we usually build **particles** from primitive objects (like tracks and calorimeter clusters).
We also calculate physics quantities, and apply cuts.

Heavy lifting

Python 3.6 code for **steering**

- Load and configure C++ modules
  - analysis modules from other packages
- Also python code for high-level analysis tasks.

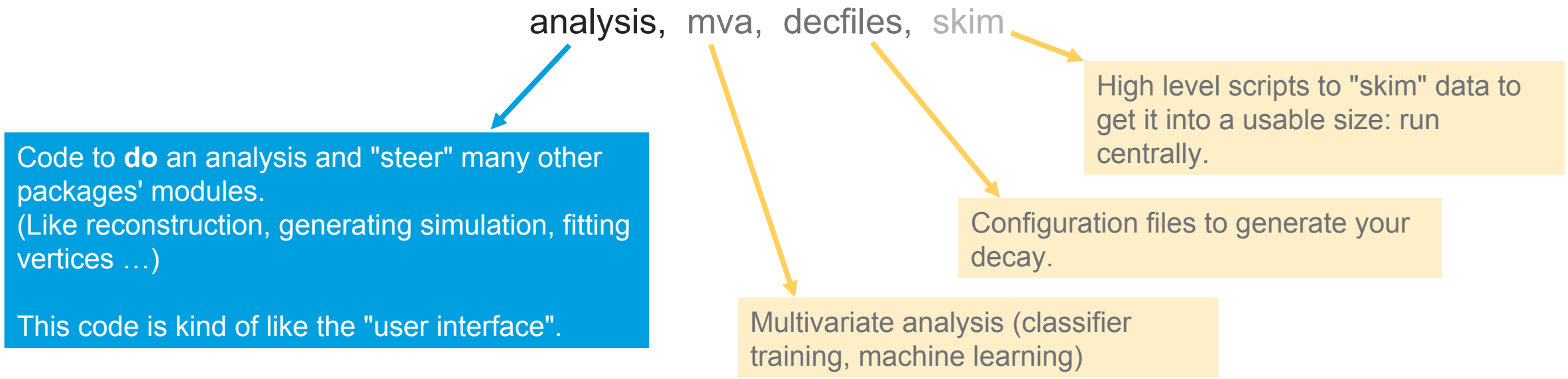
Readable scripts

- You will write a fair bit of python during the workshop.

What is the analysis package?

- Our software is organised into "packages".
  - https://stash.desy.de/projects/B2/repos/software/browse
  - There are packages for subdetectors, tracking, simulation…
  - As a student/postdoc/collaborator you might work on some of them.

- BUT! When you want to do a physics measurement. You really only care about:



Modules, paths, the DataStore and how to steer them all

Modules, paths, the DataStore

What do we need to process the data?

Kuhr, Pulvermacher, Ritter, Hauth, Braun
Comput. Softw. Big Sci. 3 (2019) no.1

- 1) A set of classes (modules) that process the data
  → **BASF2 module**

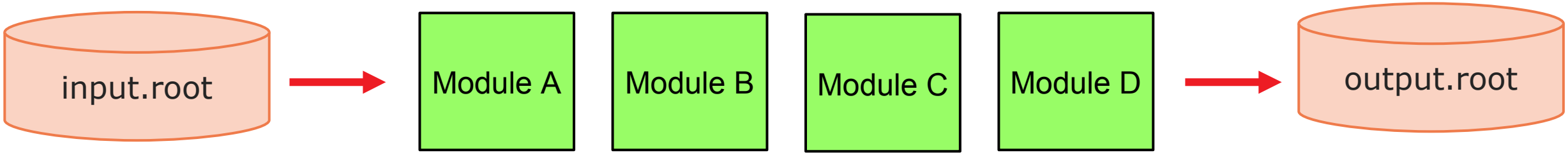
A module is written in C++ or Python and derived from a `Module` base class that defines the following interface methods:

- `initialize()`: called before the processing of events to initialize the module.
- `beginRun()`: called each time before a sequence of events of a new run is processed, e.g., to initialize run-dependent data structures like monitoring histograms.
- `event()`: called for each processed event.
- `endRun()`: called each time after a sequence of events of the same run is processed, e.g., to collect run-summary information.
- `terminate()`: called after the processing of all events.

Modules, paths, the DataStore

What do we need to process the data?

- 1) A set of classes (modules) that process the data
  → **BASF2 module**

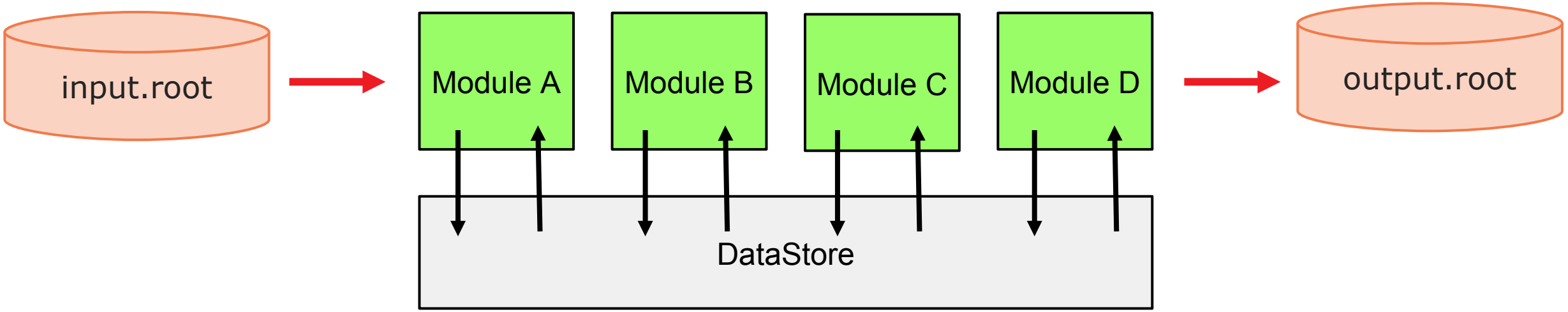


Modules, paths, the DataStore

What do we need to process the data?

- 1) A set of classes (modules) that process the data
  → **BASF2 module**

2) A set of classes (dataobjects) that hold the data and allow module to pass thing one to the other
  → **BASF2 dataObject**

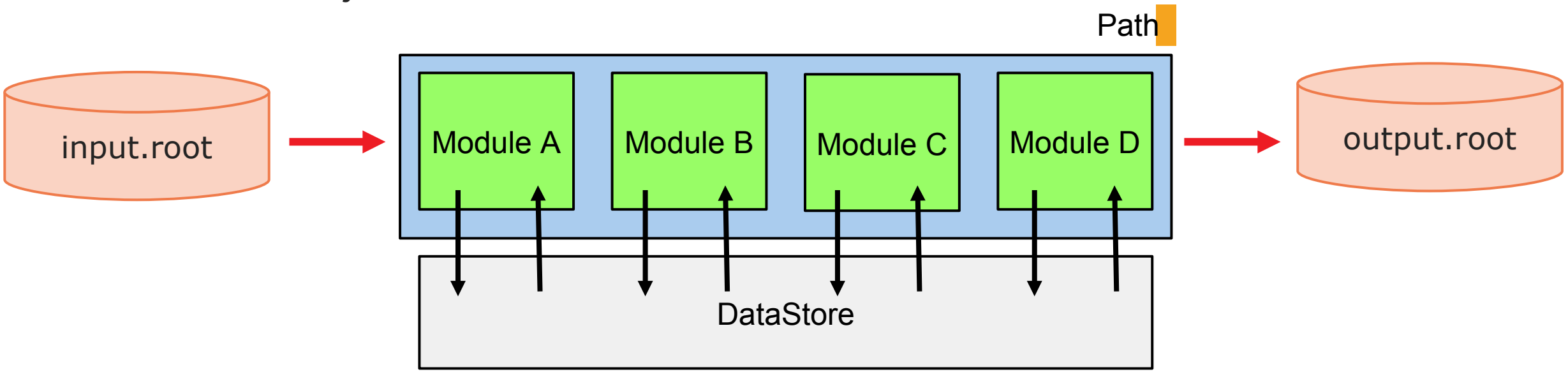


Modules, paths, the DataStore

What do we need to process the data?

- 1) A set of classes (modules) that process the data
  → **BASF2 module**

2) A set of classes (dataobjects) that hold the data and allow module to pass thing one to the other
  → **BASF2 dataObject**



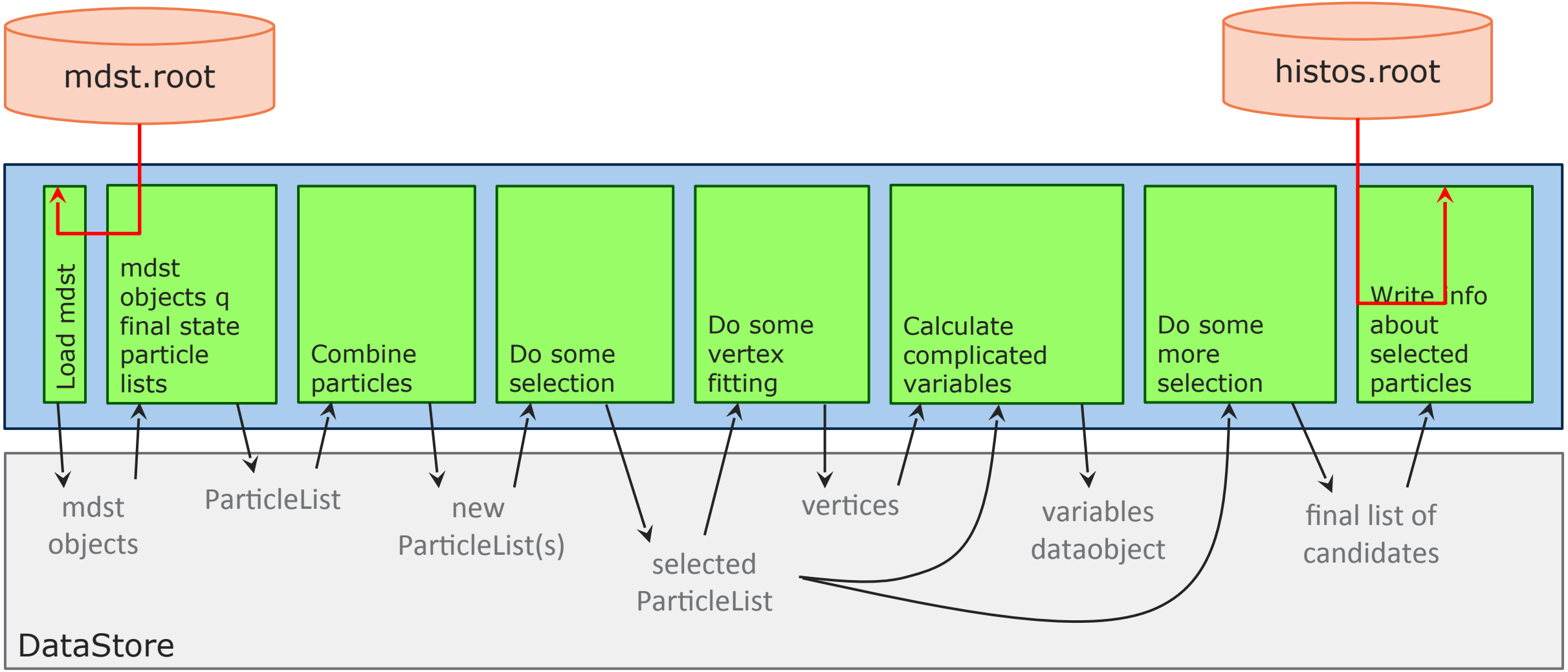
3) An order in which the modules must be executed
  → **BASF2 path**

What does a steering file look like?

How to I implement all this?

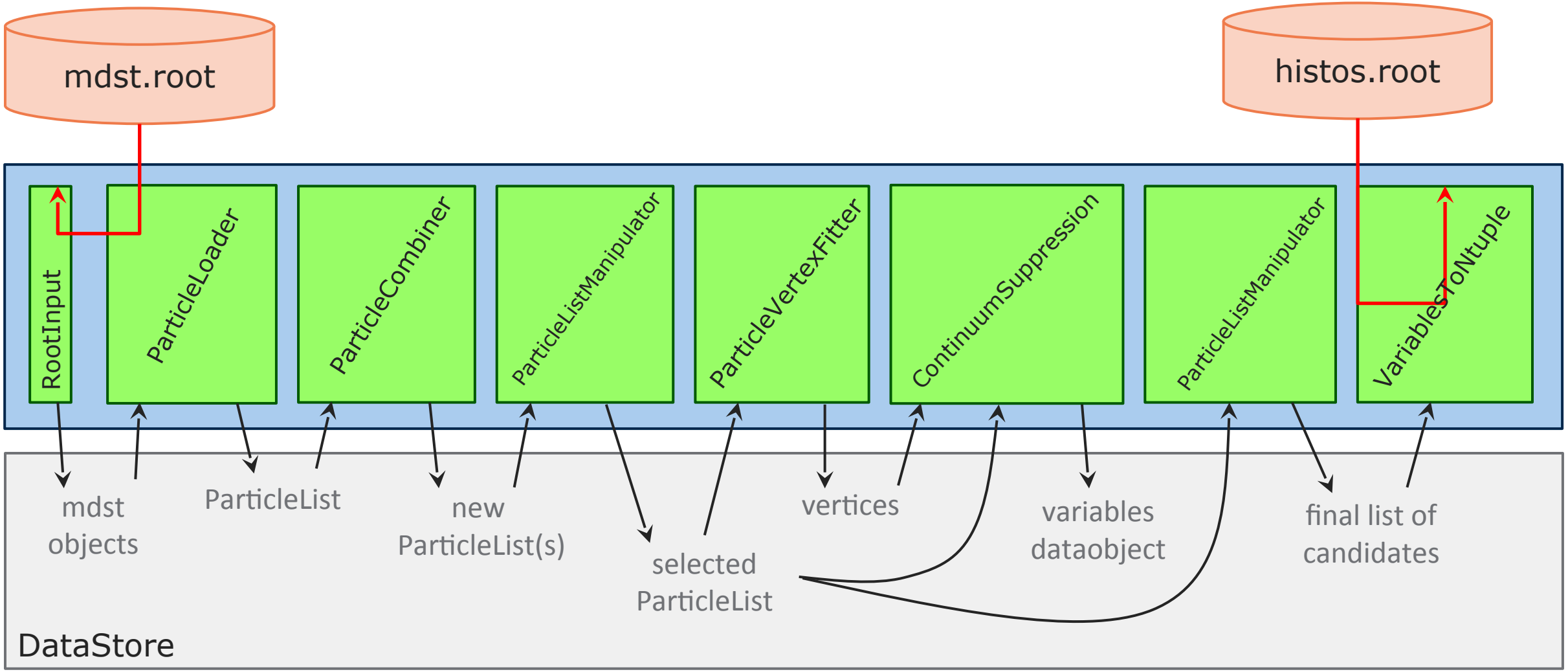
- A steering file is a python script that runs
  → the modules that you need
  → in the order you need
  → on the data you need

A typical path for an analysis job



A typical path for an analysis job

Now with the real names for the modules



A word about file types and why you should use the analysis package

File types

That basf2 can read and/or create

- A **dst** is contains basf2 objects which will populate a **DataStore**.
  - **d**ata **s**ummary **t**able
  - Basically: a special ROOT file.

- The data for physics analysis are "**mdst**".
  - **m**ini **d**ata **s**ummary **t**able.
  - Same structure of a **dst, but with much less information**
  - Input to your analysis package scripts

- The calibration & performance are **“cdst”**
  - **c**alibration **m**ini **d**ata **s**ummary **t**able.
  - mdst + digits

- At the end of your analysis chain you will write out a "normal" root file containing a TTree, TNtuple, or histograms

A relevant question
https://questions.belle2.org/question/219

Objects allowed in an ***m***dst:
https://goo.gl/AB15Ud

Why use the analysis package?

Can I read the mdst by myself?

mdst are basically root trees containing lists of:

→ Track
→ TrackFitResult
→ V0
→ PIDLikelihood
→ ECLCluster
→ KLMCluster
→ KlID
→ TRGSummary
→ SoftwareTriggerResult
→ (MCParticle)

The analysis package has modules to convert these
Into more friendly quantities like
→ Particle
→ PID probability
→ Vertex position
→ photons
→ …..

Why use the analysis package?

Can I read the mdst by myself?

Can I open the mdst with my own, custom-made macro and run the analysis?

Why use the analysis package?

Can I read the mdst by myself?

Can I open the mdst with my own, custom-made macro and run the analysis?

NO

The mdst contain also the relations bewteen the objects stored in it, which are not trivially handeled by a stand-alone root macro. **Use always basf2-based code.**

Why use the analysis package?

Can I read the mdst by myself?

Can I open the mdst with my own, custom-made macro and run the analysis?

NO

The mdst contain also the relations bewteen the objects stored in it, which are not trivially handeled by a stand-alone root macro. **Use always basf2-based code.**

Should I write my own module that loops over reconstructed objects like the ECLClusters and do the analysis (i.e. Belle-style)?

Why use the analysis package?

Can I read the mdst by myself?

Can I open the mdst with my own, custom-made macro and run the analysis?

NO

The mdst contain also the relations bewteen the objects stored in it, which are not trivially handeled by a stand-alone root macro. **Use always basf2-based code.**

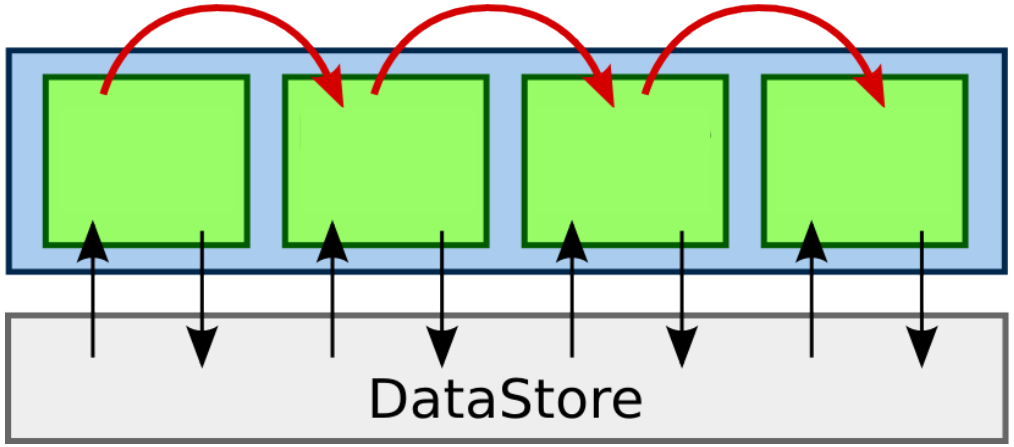
Should I write my own module that loops over reconstructed objects like the ECLClusters and do the analysis (i.e. Belle-style)?

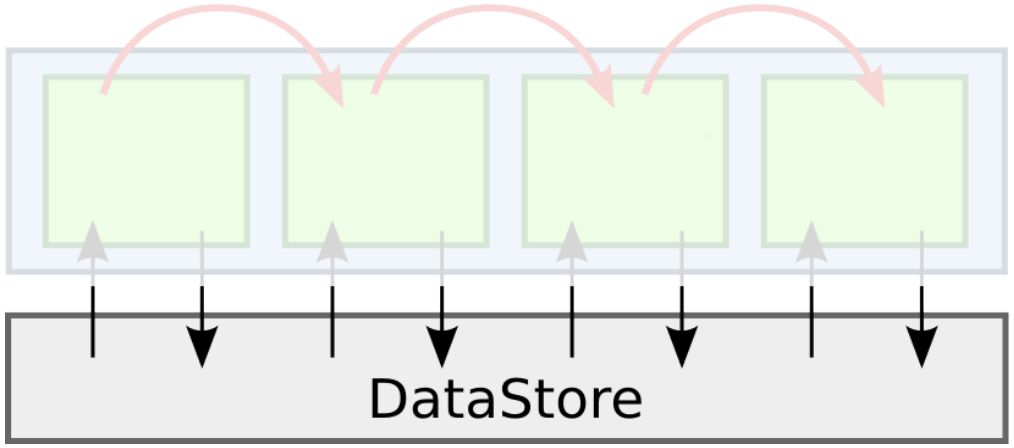
NO

The relation bewteen analysis object (particles) and the reconstructed objects is not always trivial.
One particle may have many trackFitResults
The ECLClusters are not photons.
Use the modules provided by a detector expert

Candidate/particle based analysis

Before we get started, here's this diagram again





ou can take a look at examples i
<package>/dataobjects

- The datastore contains the **dataobjects**
- At the level of analysis, the main dataobject is: **ParticleList**

Particle-based analysis

- Take particle lists
- Build up decay parents from daughters
- Make ***candidate***s for your decay of interest
- Filter/cut/keep.

- You might have more than one candidate per event.
  - We deal with this after the fact.
  - This is fine. I promise.
  - https://arxiv.org/abs/1703.01128

The Particle class

It's not crucial to understand the details

- A common representation of all particle types
  - Charged: e / μ / π / K / p [built up from track + hypothesis]
  - γ [built up from ECLClusters + !Track]
  - K0L [KLMClusters]
  - Composite particles: π^0 / K^0_S / D / B [built up combinations]

- Data members of the class are **common to all particle types**: mass, momentum, position, PDG code, …

- Information which is only relevant to certain kinds of particle is saved in separate **analysis package dataobjects** and accessible by **relations**.
  - e.g. ContinuumSuppression
  - …. FlavorTaggerInfo

ParticleList

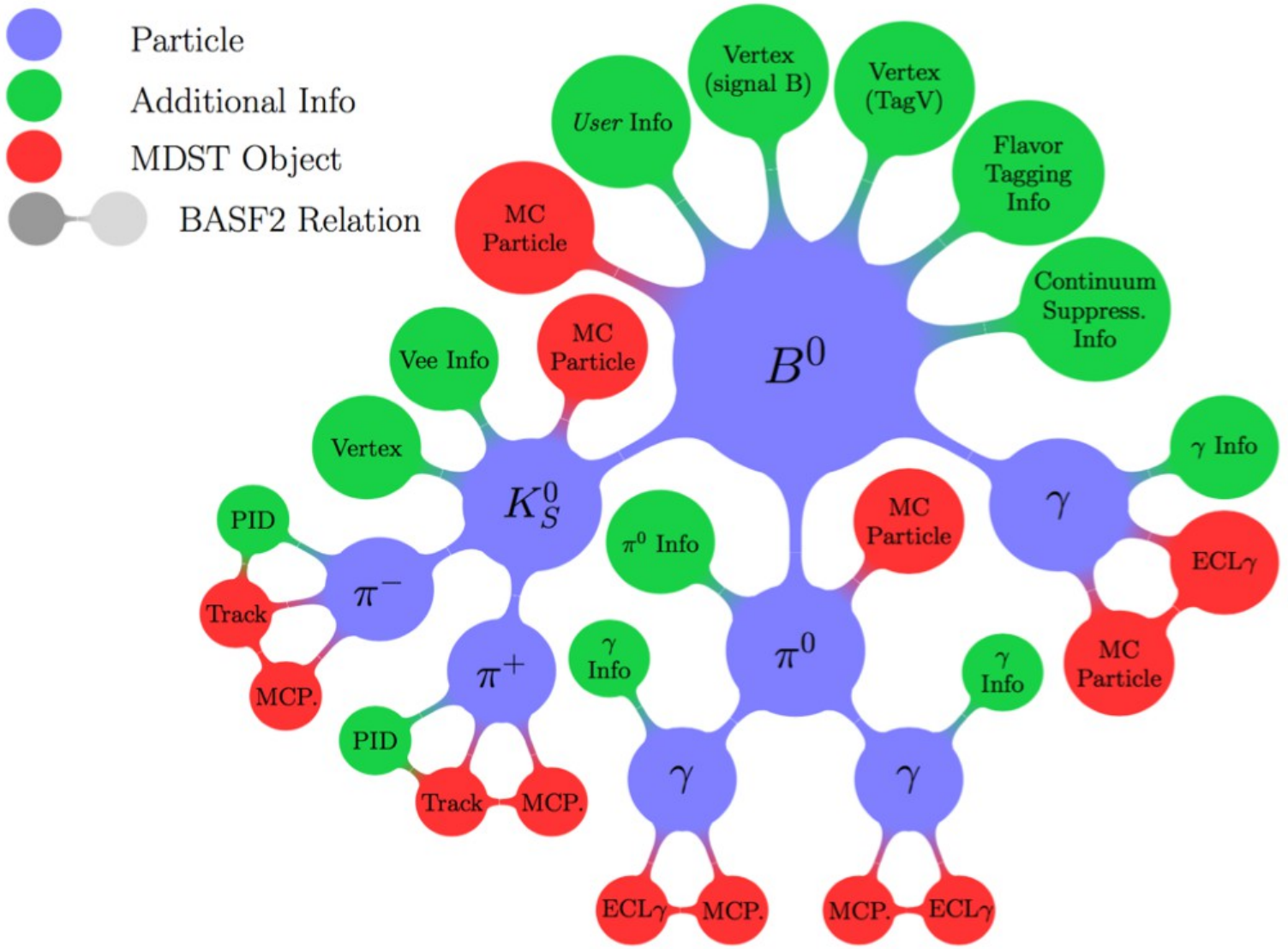
- A group of all particles and anti-particles that belong together logically.
  - e.g. K^{*0} s (decaying to K and π with invariant mass in a certain window)
- Can only store particles of the same PDG code (can be different decay modes).
- Doesn't have ownership of the **Particle** objects.
- **ParticleList** is the **dataobject** on which analysis modules operate.

- The physics-performance group provides **Standard Particle Lists** for which quality benchmarks exist, and systematics will be provided.
  - I really recommend you use these.
  - … But you should know what they do.

Q: you've already seen the standard particle lists in action in these slides… where?

Q: can you find the benchmarks… in the documentation? In the code?

Some more details on Particles, ParticleLists and Relations



- At each stage we build relations between the dataobjects
  - Like vertex information, ContinuumSuppression → all related to Particles
  - Particles themselves related to primitive mdst objects (clusters, tracks)

Nomenclature

Nomenclature

https://confluence.desy.de/display/BI/Main+Glossary

- **Experiment** (chunk of data-taking ~months).
- **Run** (chunk of data-taking w/ stable beams ~hours),
- Event.

- **TRG** the hardware trigger (group, device, DAQ)
- **L1** the hardware trigger (used interchangeably)

- SoftwareTrigger / **HLT** (the software trigger)
-
- **basf2** “Belle 2 analysis software framework” “the software”
- **gbasf2** “The grid job submission tool” “computing”

Hands-on time:

101 example of script analysing $B \to D\pi$ decays.