RIVET + JETSCAPE Part - 1

Raghav Kunnawalkam Elayavalli (they/them) [Vanderbilt University]
Christal Martin, Shannon Harris, Tanner Mengal, Joesph Beller and Christine Nattrass [UTK]
Special thanks to Antonio Silva [ISU]
Slides thanks to - Christian Bierlich [Lund], Louie Corpe [CERN]

JETSCAPE Summer School 2023, 20th July

What is RIVET?
Your first Analysis!
Running RIVET + ROOT

Power to the people!

Shannon Harris

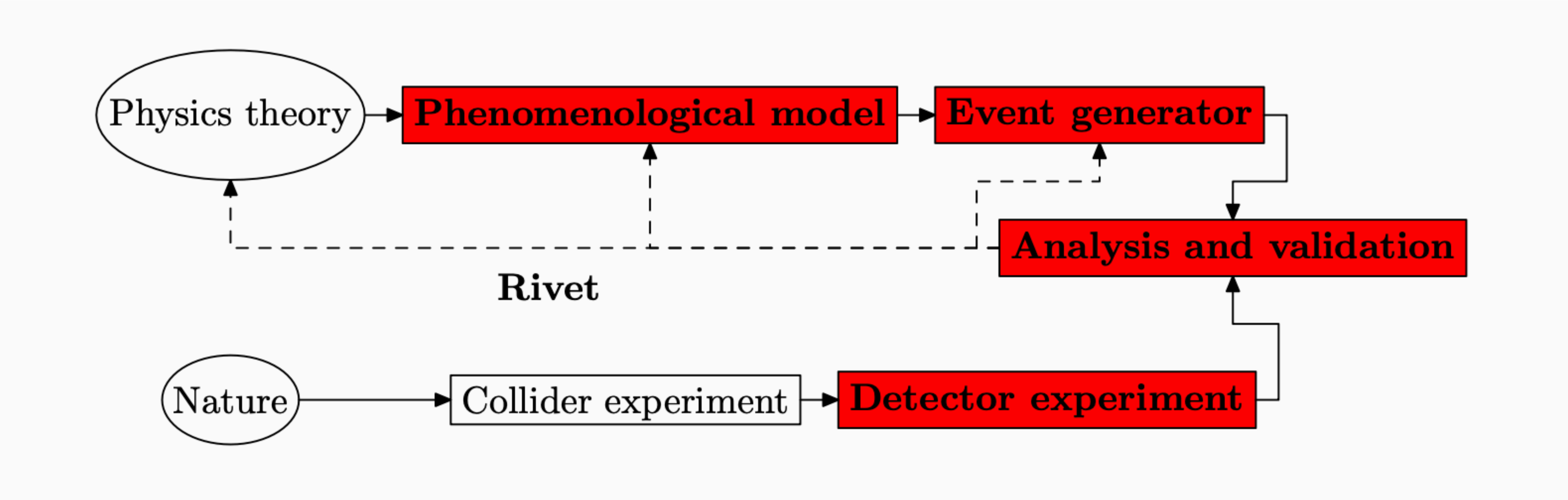
Tanner Mengel

Joesph Beller

Antonio DaSilva

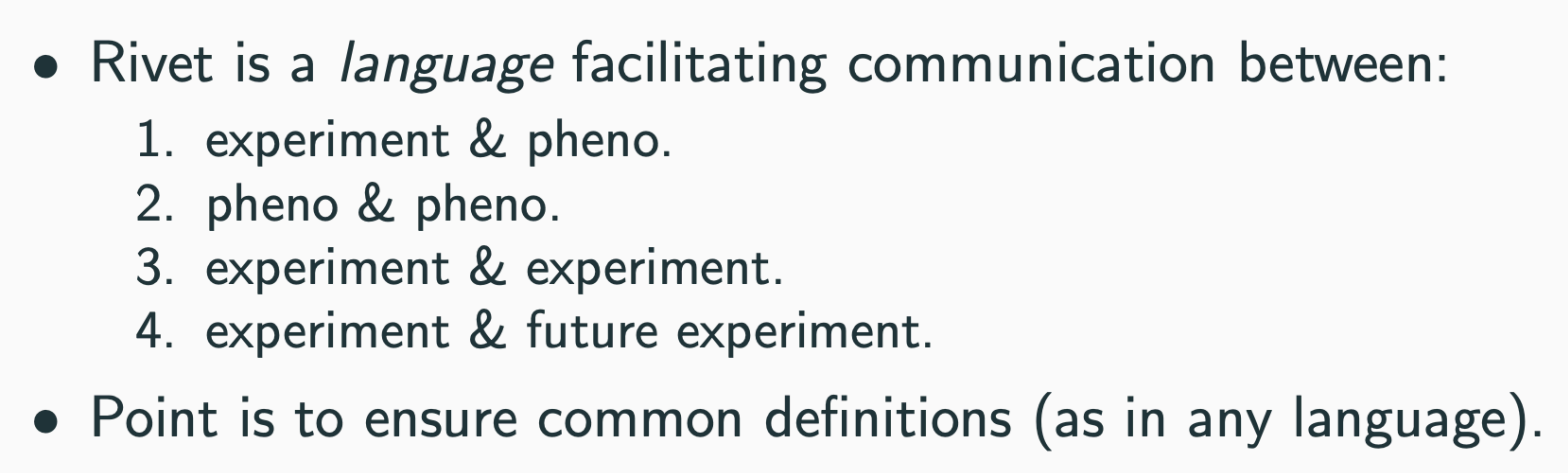
Christal Martin

What is RIVET?



Robust Independent Validation of Experiment and Theory

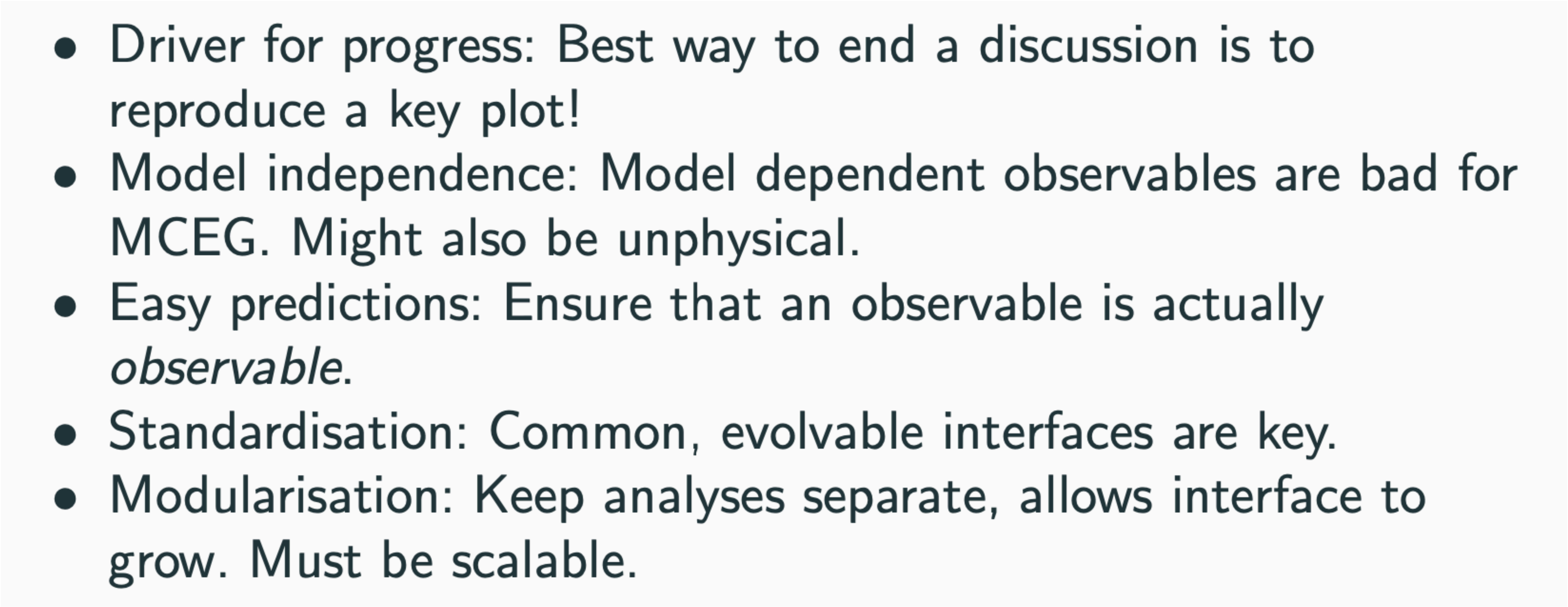
What is RIVET?



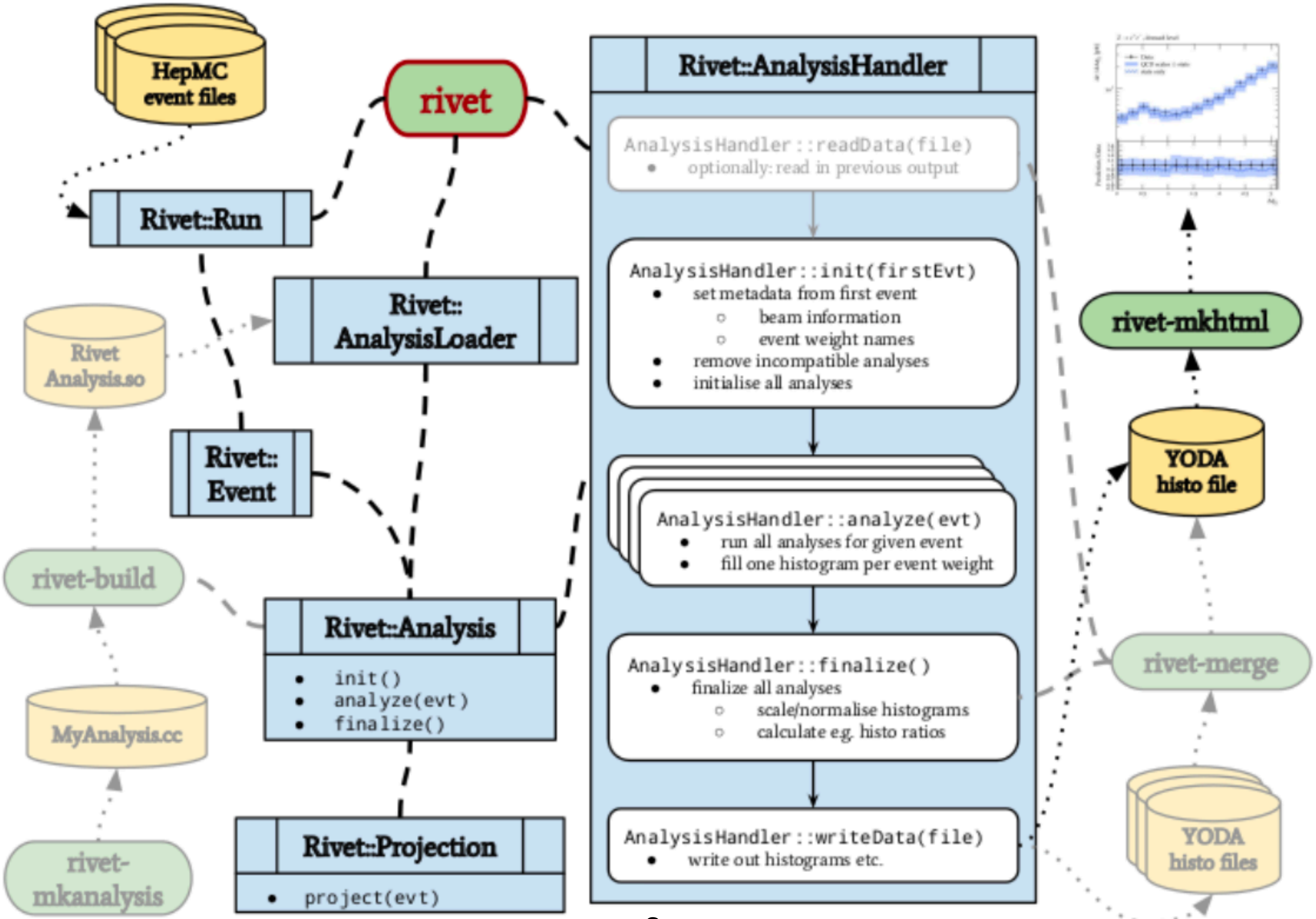
- Rivet is a *language* facilitating communication between:
  1. experiment & pheno.
  2. pheno & pheno.
  3. experiment & experiment.
  4. experiment & future experiment.
- Point is to ensure common definitions (as in any language).

Phenomenology here refers to an implementation of a theory prescription of a phenomenon towards calculating something that can be measured

Why do we need RIVET?



- Driver for progress: Best way to end a discussion is to reproduce a key plot!
- Model independence: Model dependent observables are bad for MCEG. Might also be unphysical.
- Easy predictions: Ensure that an observable is actually *observable*.
- Standardisation: Common, evolvable interfaces are key.
- Modularisation: Keep analyses separate, allows interface to grow. Must be scalable.



HepMC event files

rivet

Rivet::Run

Rivet::AnalysisLoader

Rivet Analysis.so

Rivet::Event

rivet-build

MyAnalysis.cc

rivet-mkanalysis

Rivet::Analysis

- init()
- analyze(evt)
- finalize()

Rivet::Projection

- project(evt)

Rivet::AnalysisHandler

AnalysisHandler::readData(file)
- optionally: read in previous output

AnalysisHandler::init(firstEvt)
- set metadata from first event
  - beam information
  - event weight names
- remove incompatible analyses
- initialise all analyses

AnalysisHandler::analyze(evt)
- run all analyses for given event
- fill one histogram per event weight

AnalysisHandler::finalize()
- finalize all analyses
  - scale/normalise histograms
  - calculate e.g. histo ratios

AnalysisHandler::writeData(file)
- write out histograms etc.

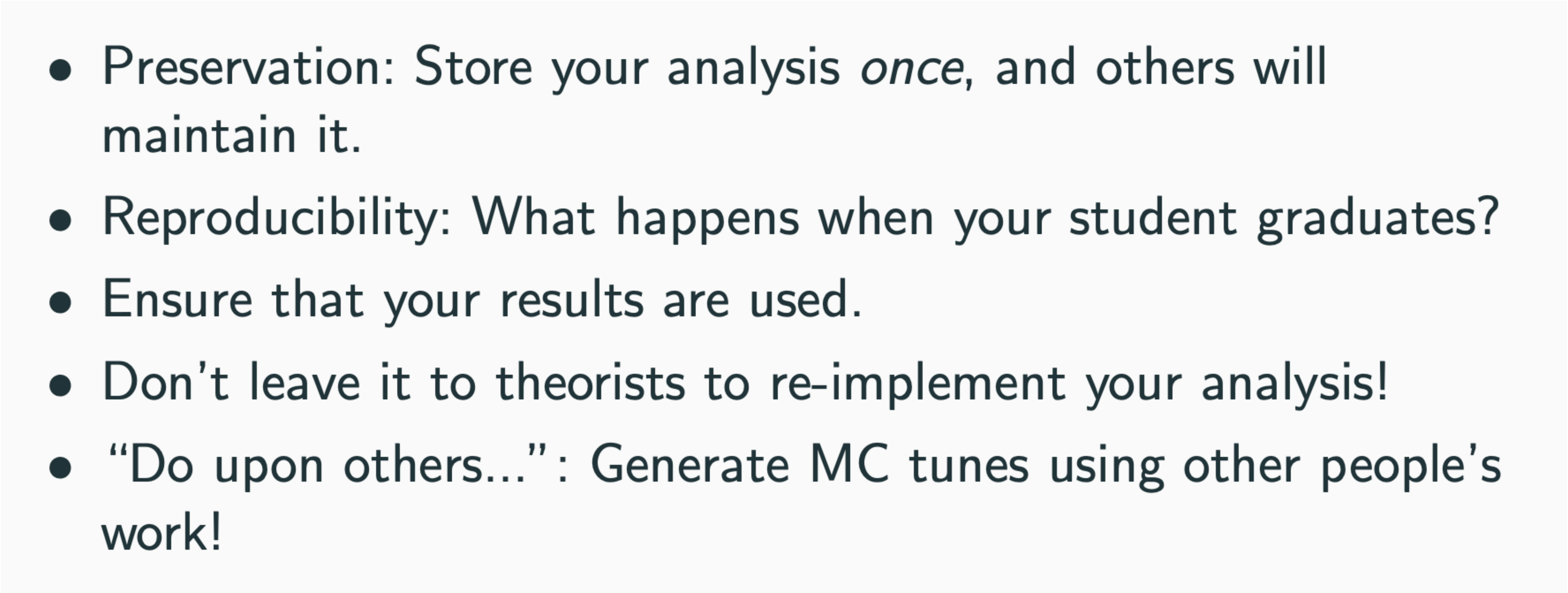
rivet-mkhtml

YODA histo file

rivet-merge

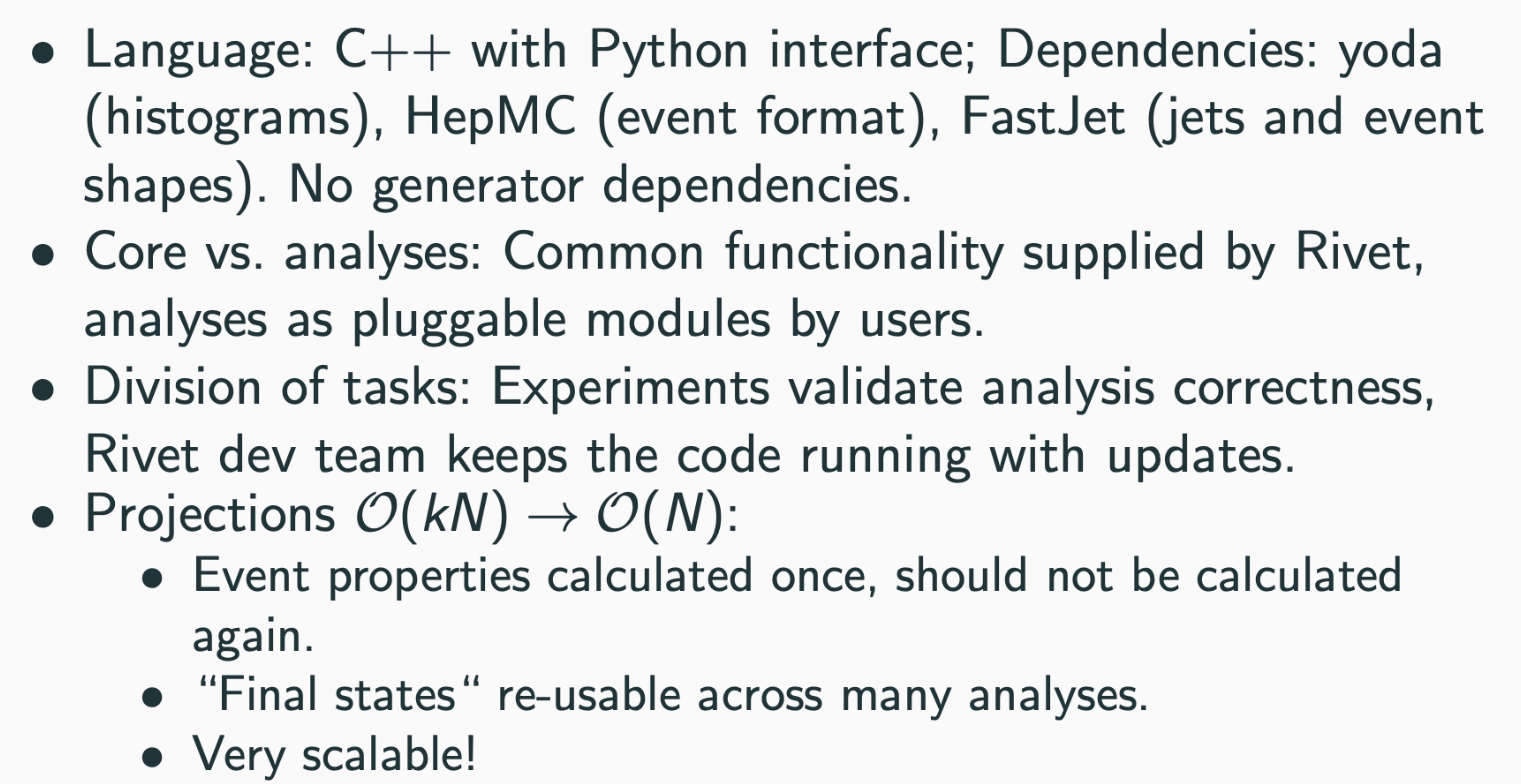
YODA histo files

Why should experimentalists learn it?



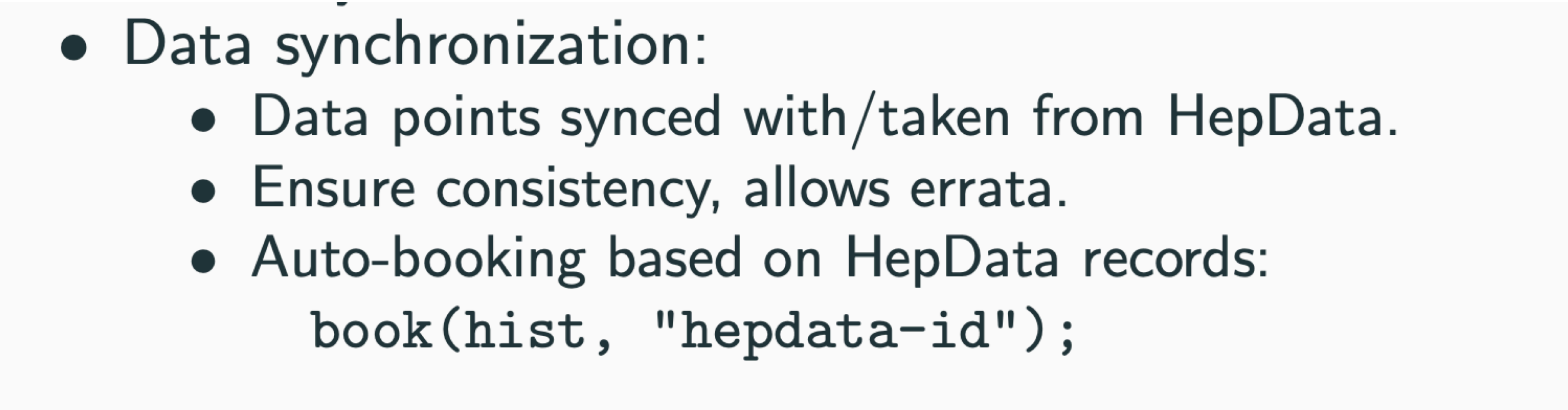
- Preservation: Store your analysis *once*, and others will maintain it.
- Reproducibility: What happens when your student graduates?
- Ensure that your results are used.
- Don't leave it to theorists to re-implement your analysis!
- "Do upon others...": Generate MC tunes using other people's work!

Why should theorists learn it?



- Language: C++ with Python interface; Dependencies: yoda (histograms), HepMC (event format), FastJet (jets and event shapes). No generator dependencies.
- Core vs. analyses: Common functionality supplied by Rivet, analyses as pluggable modules by users.
- Division of tasks: Experiments validate analysis correctness, Rivet dev team keeps the code running with updates.
- Projections $\mathcal{O}(kN) \to \mathcal{O}(N)$:
    - Event properties calculated once, should not be calculated again.
    - “Final states“ re-usable across many analyses.
    - Very scalable!

Why should theorists learn it?



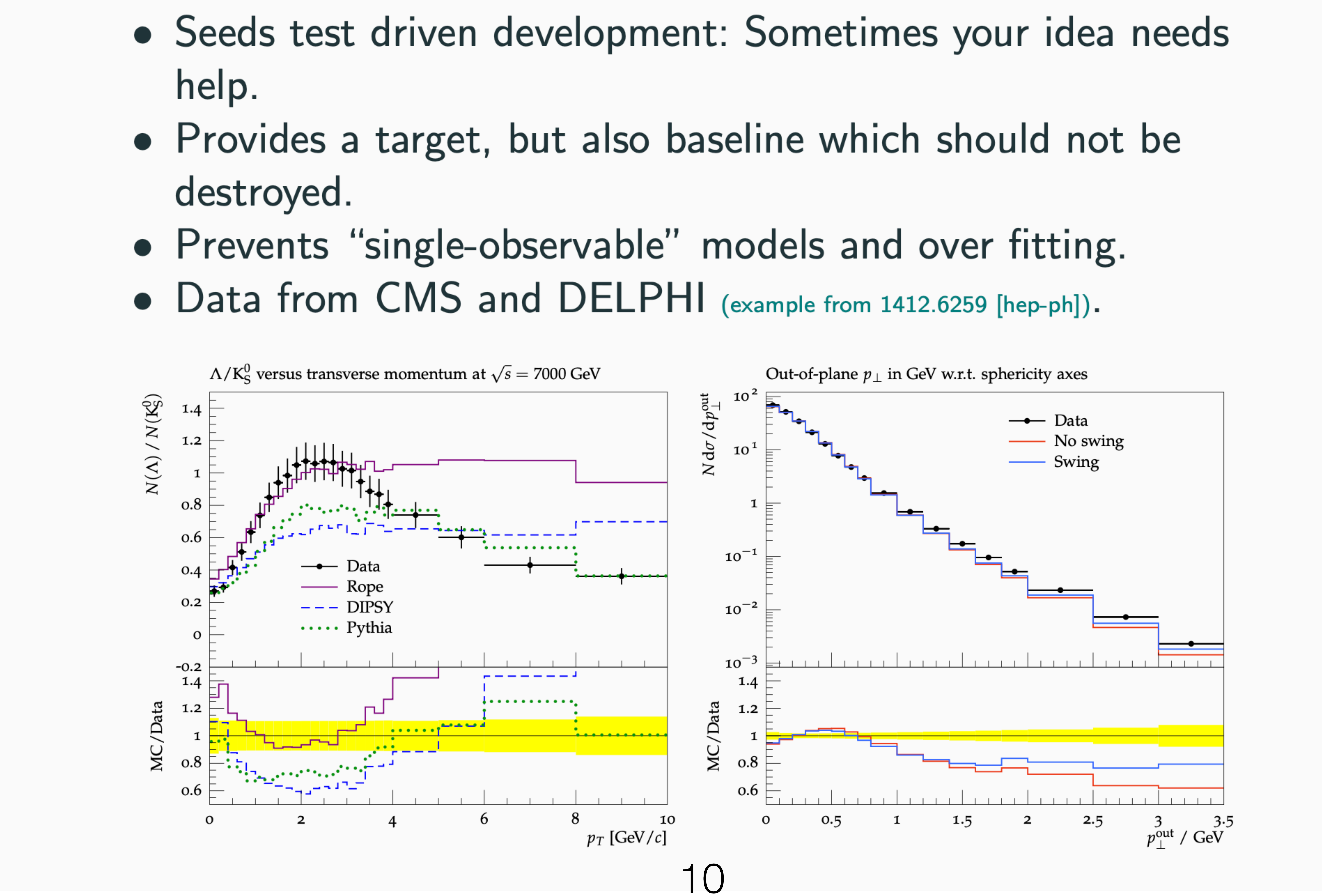
- Data synchronization:
  - Data points synced with/taken from HepData.
  - Ensure consistency, allows errata.
  - Auto-booking based on HepData records:

    ```
    book(hist, "hepdata-id");
    ```

This allows one to directly compare the measurement to your calculations

For today - we will skip this part and come back to it next week!

Uses of RIVET

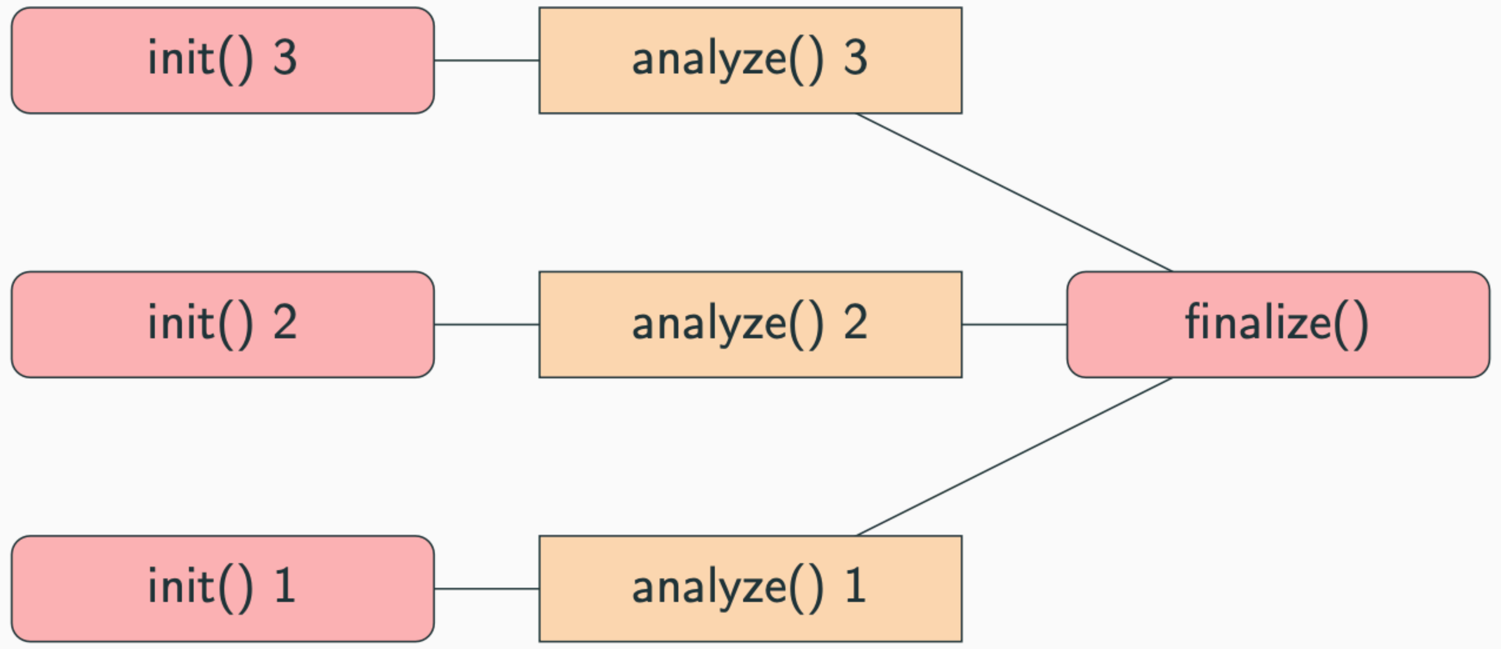


- Seeds test driven development: Sometimes your idea needs help.
- Provides a target, but also baseline which should not be destroyed.
- Prevents “single-observable” models and over fitting.
- Data from CMS and DELPHI (example from 1412.6259 [hep-ph]).

In HEP -

“Big data” I: perfect run combination

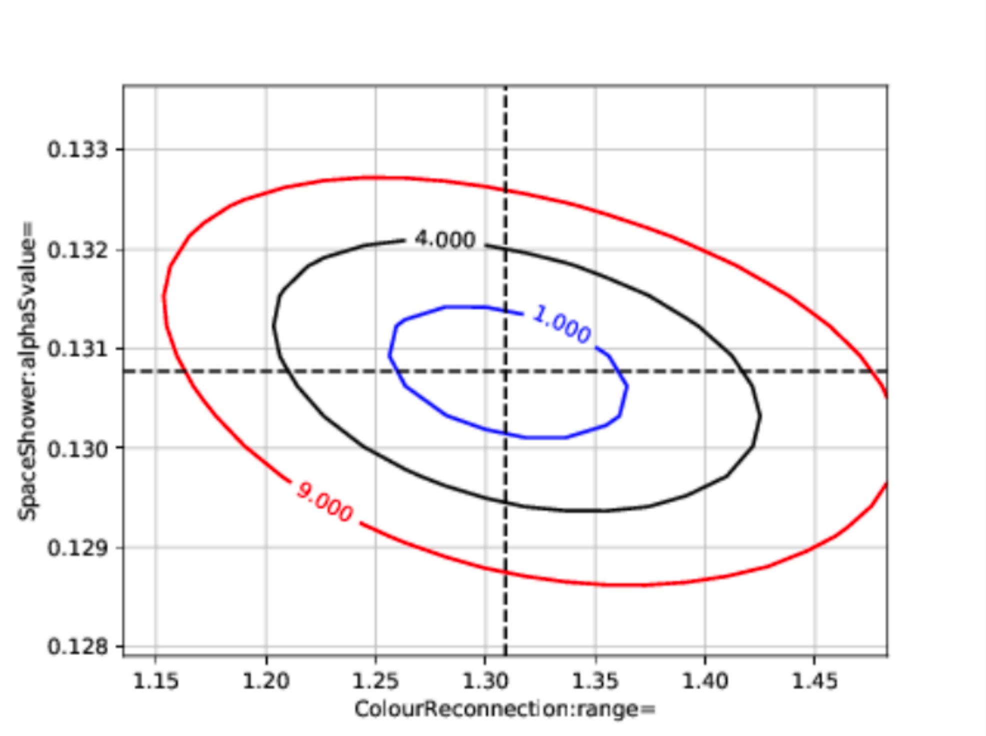
- Parallelization is necessary but potentially difficult.
- Old solution `yoda-merge` only for special cases.
- Consider: flavour ratios, R_{AA}, flow...
- Solution: `rivet-merge` before finalization.



- Let analyser implement merging → *perfect run combination*.

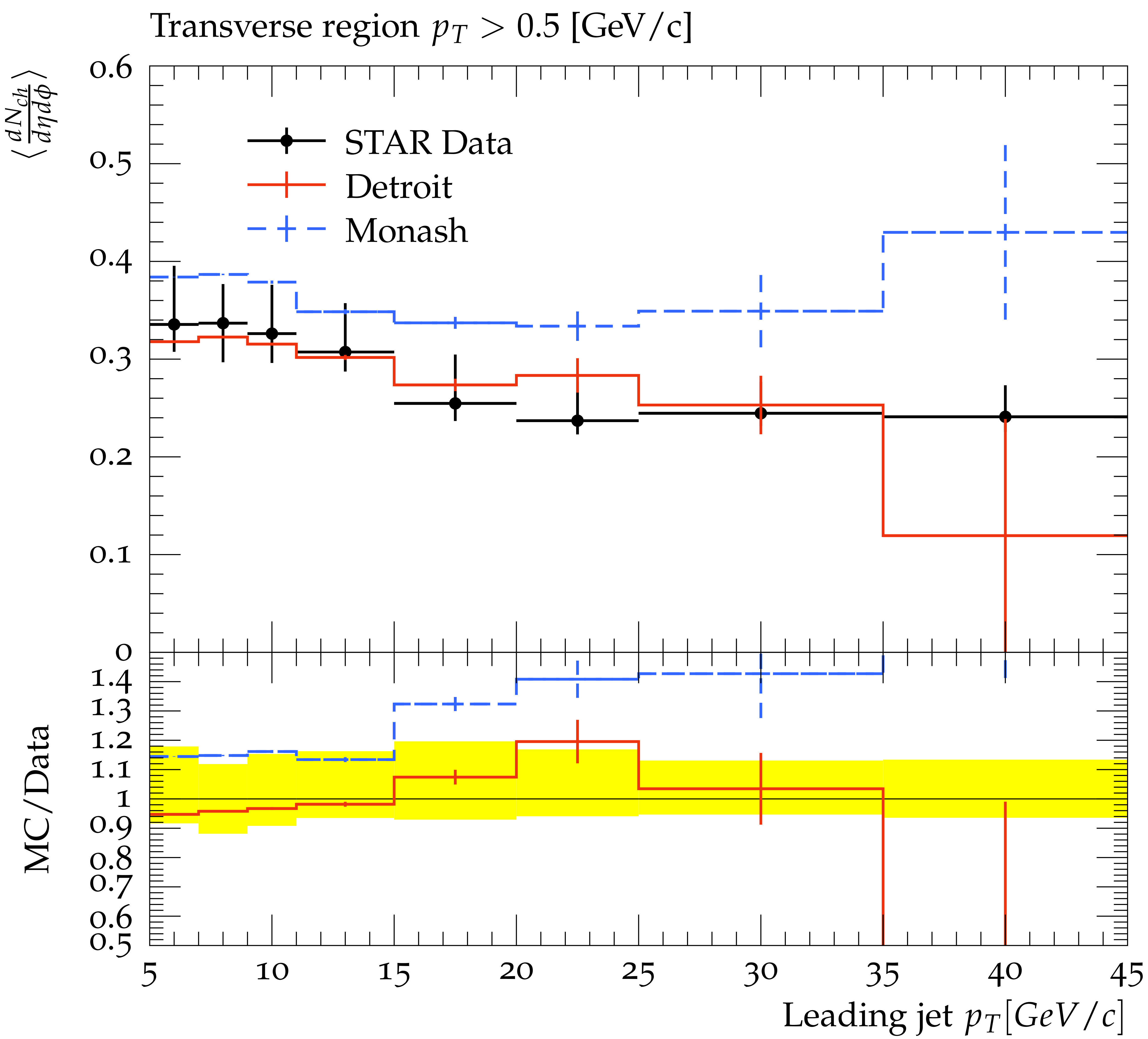
Big data II: Generator tuning

- With many available analyses comes possibilities.
- Systematized generator tuning is one! (https://professor.hepforge.org/)
- This is not a tuning talk, but...



- Future ALICE efforts possibly include compatibility of freezeout models.
- Full statistical framework for free! Large scale tests of QGP models? (like Contur for BSM)

One recent example - PYTHIA 8 Detroit Tune

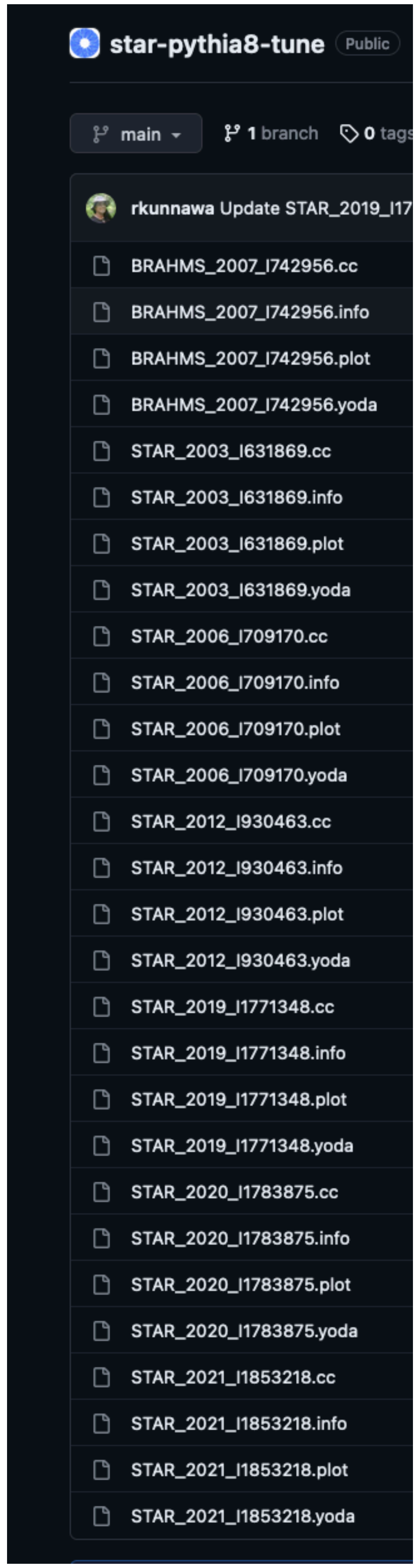


STAR Data

Detroit

Monash

Aguilar et. al, Phys. Rev. D 105 (2022) 1, 016011



star-pythia8-tune Public

main 1 branch 0 tags

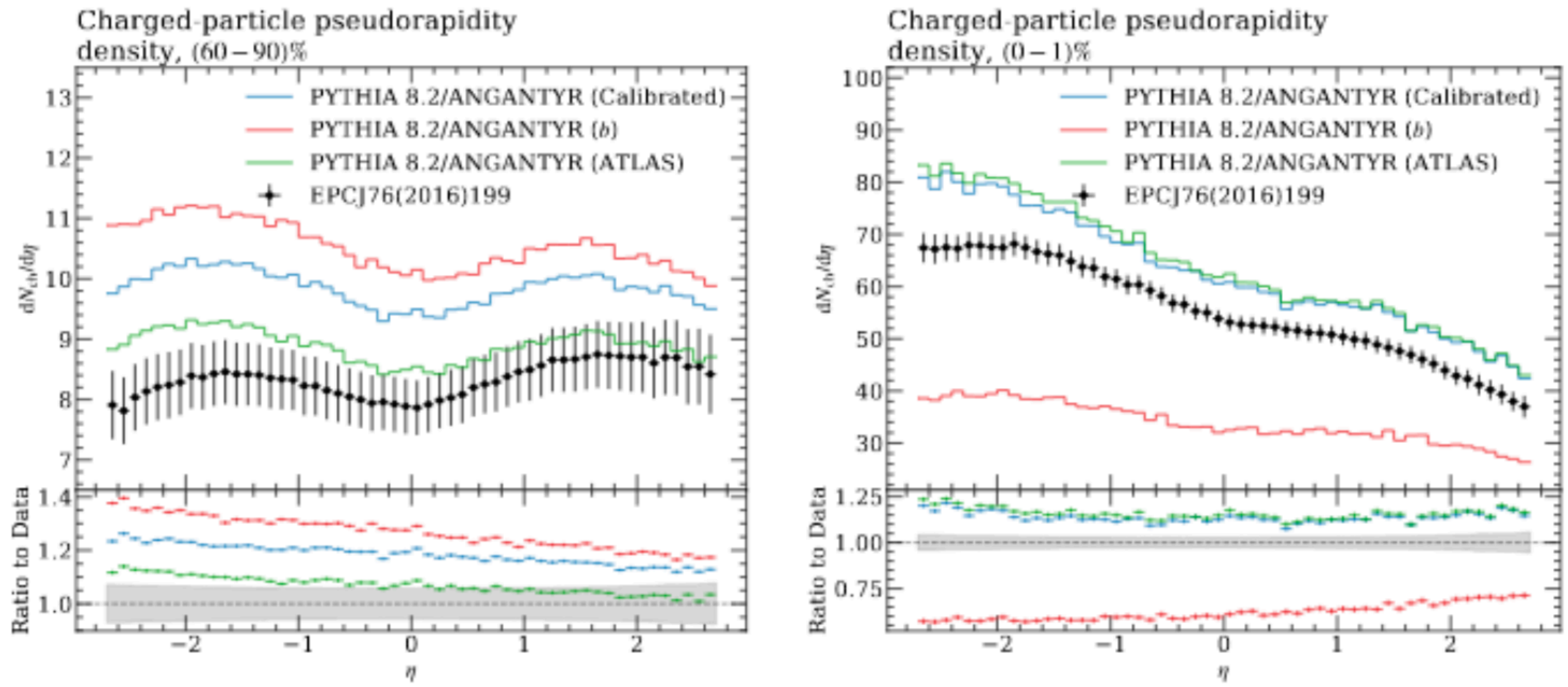
rkunnawa Update STAR_2019_I17

BRAHMS_2007_I742956.cc
BRAHMS_2007_I742956.info
BRAHMS_2007_I742956.plot
BRAHMS_2007_I742956.yoda
STAR_2003_I631869.cc
STAR_2003_I631869.info
STAR_2003_I631869.plot
STAR_2003_I631869.yoda
STAR_2006_I709170.cc
STAR_2006_I709170.info
STAR_2006_I709170.plot
STAR_2006_I709170.yoda
STAR_2012_I930463.cc
STAR_2012_I930463.info
STAR_2012_I930463.plot
STAR_2012_I930463.yoda
STAR_2019_I1771348.cc
STAR_2019_I1771348.info
STAR_2019_I1771348.plot
STAR_2019_I1771348.yoda
STAR_2020_I1783875.cc
STAR_2020_I1783875.info
STAR_2020_I1783875.plot
STAR_2020_I1783875.yoda
STAR_2021_I1853218.cc
STAR_2021_I1853218.info
STAR_2021_I1853218.plot
STAR_2021_I1853218.yoda

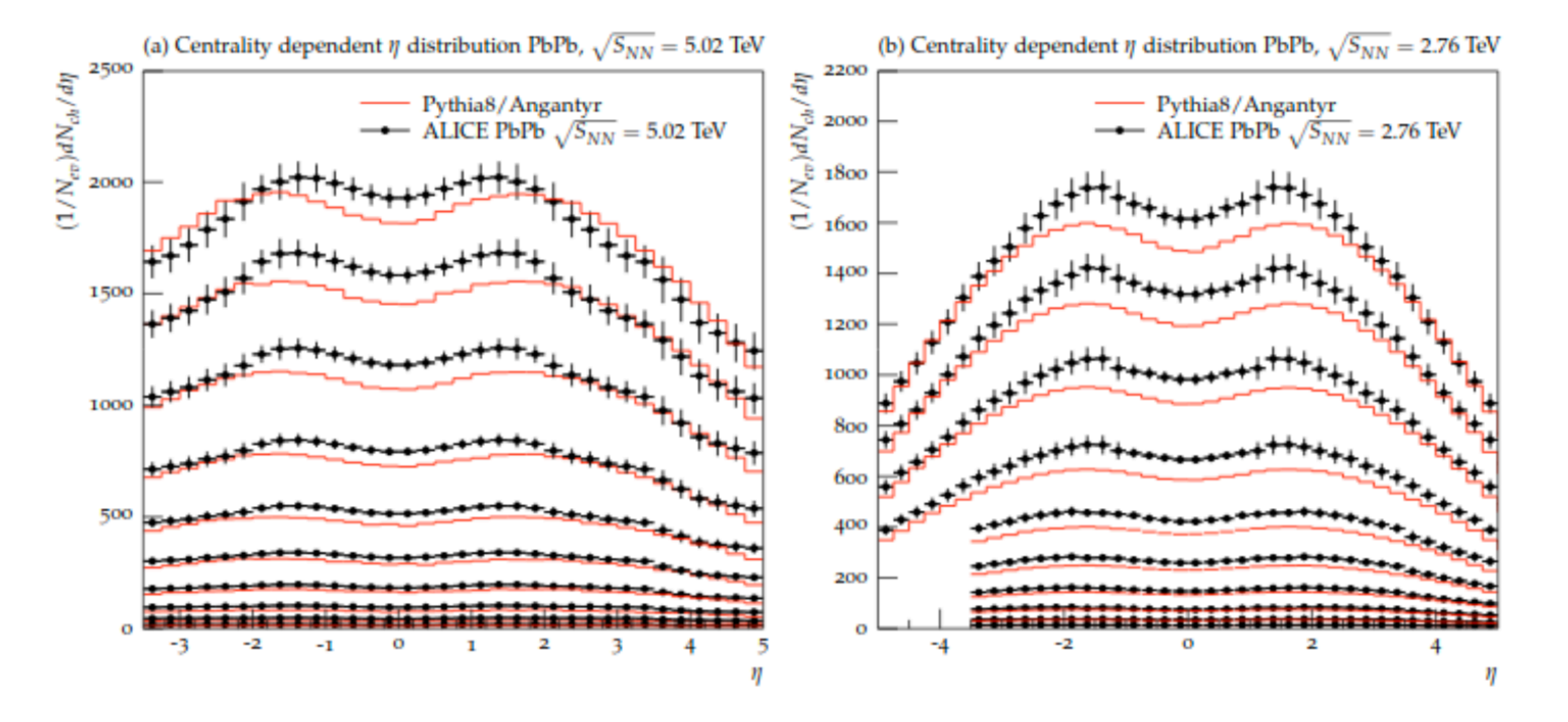
New features example: Rivet for Heavy ions

- Good example: Recent venture into heavy ion physics: Rivet for Heavy Ions (2001.10737 [hep-ph])
- Rivet for heavy ions is/was:
  - A dedicated crunch towards including HI functionality.
  - Included several people from both sides.
  - Documented in the paper above, and included in Rivet proper.
  - Not a done deal. Many potential improvements possible.
- Rivet for heavy ions is *not*:
  - Something separate from Rivet proper.
- Result: Features to allow comparison between heavy ion data and MC.

- Can’t do HI without centrality.
- Theory level definition not the same as experimental.
- Subtle biases quantified: especially in *p*A.



- Correctness is important. Another example (Angantyr: 1806.10820 [hep-ph]
- Both are 10% effects, same as MC accuracy.

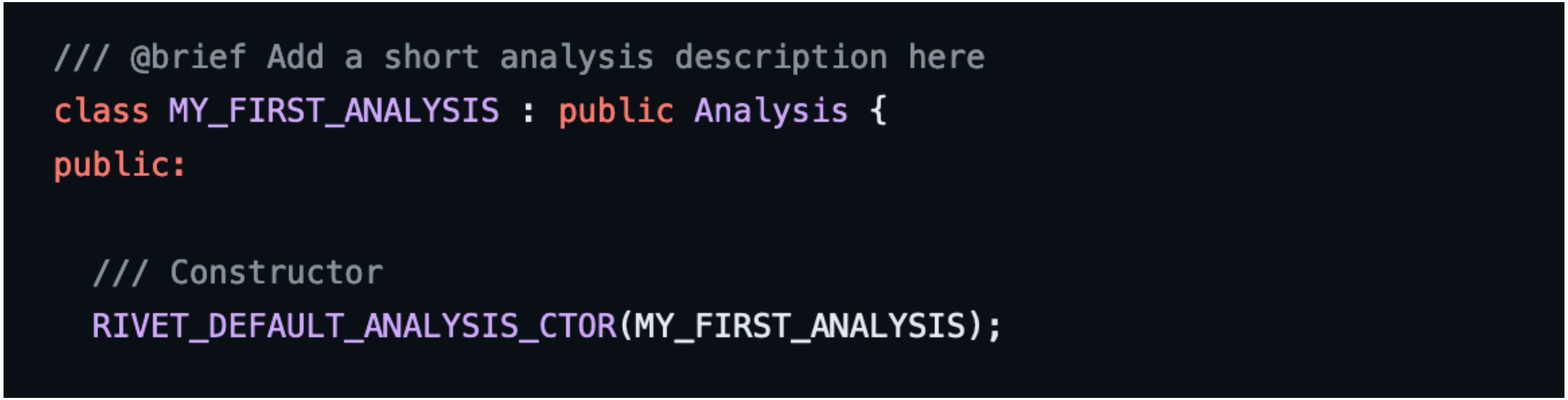


Rivet for HI

◇ Includes ALICE:: trigger projections.

◇ Includes ALICE:: primary particle projections.

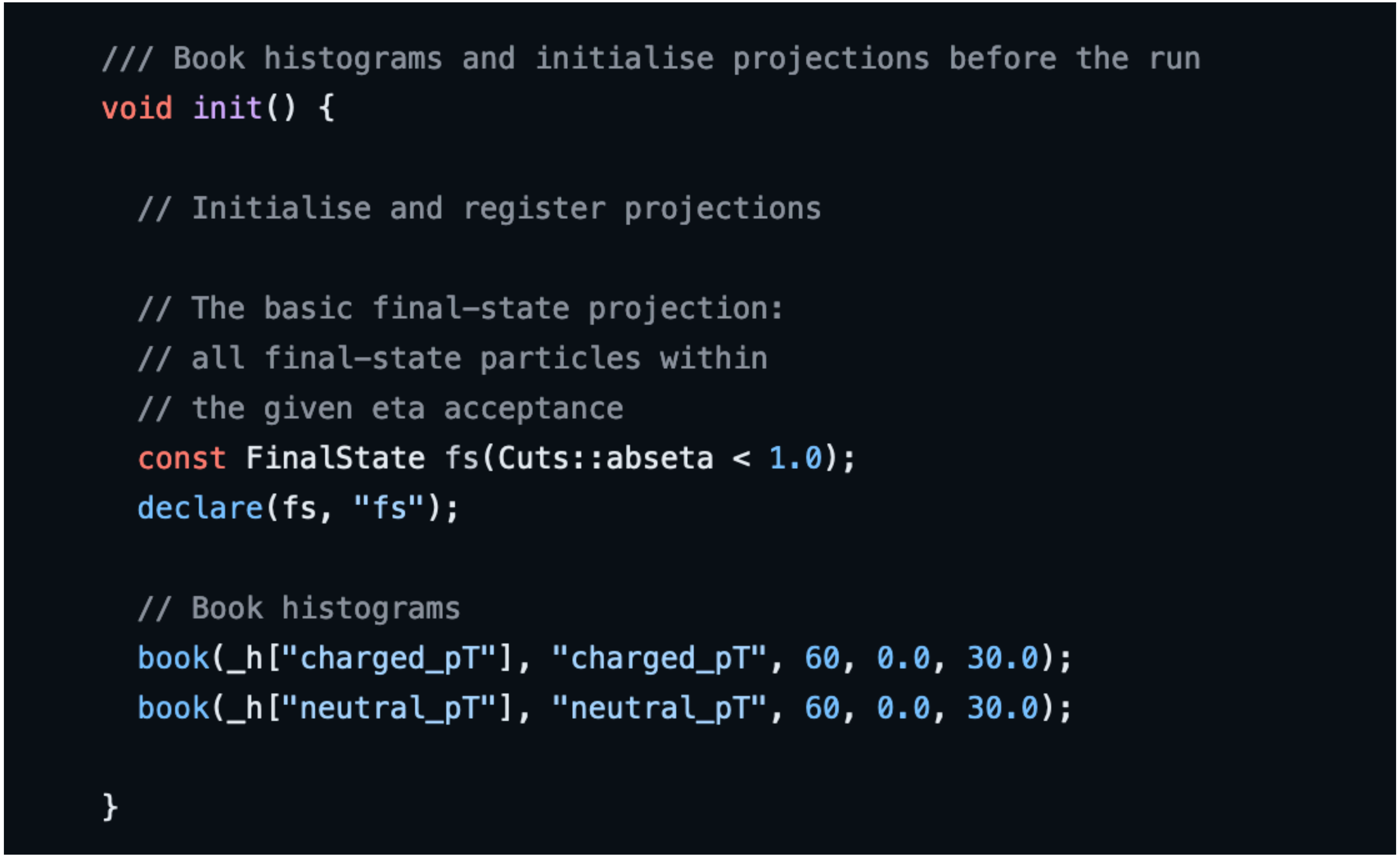
Writing your own analysis!



```
/// @brief Add a short analysis description here
class MY_FIRST_ANALYSIS : public Analysis {
public:

  /// Constructor
  RIVET_DEFAULT_ANALYSIS_CTOR(MY_FIRST_ANALYSIS);
```

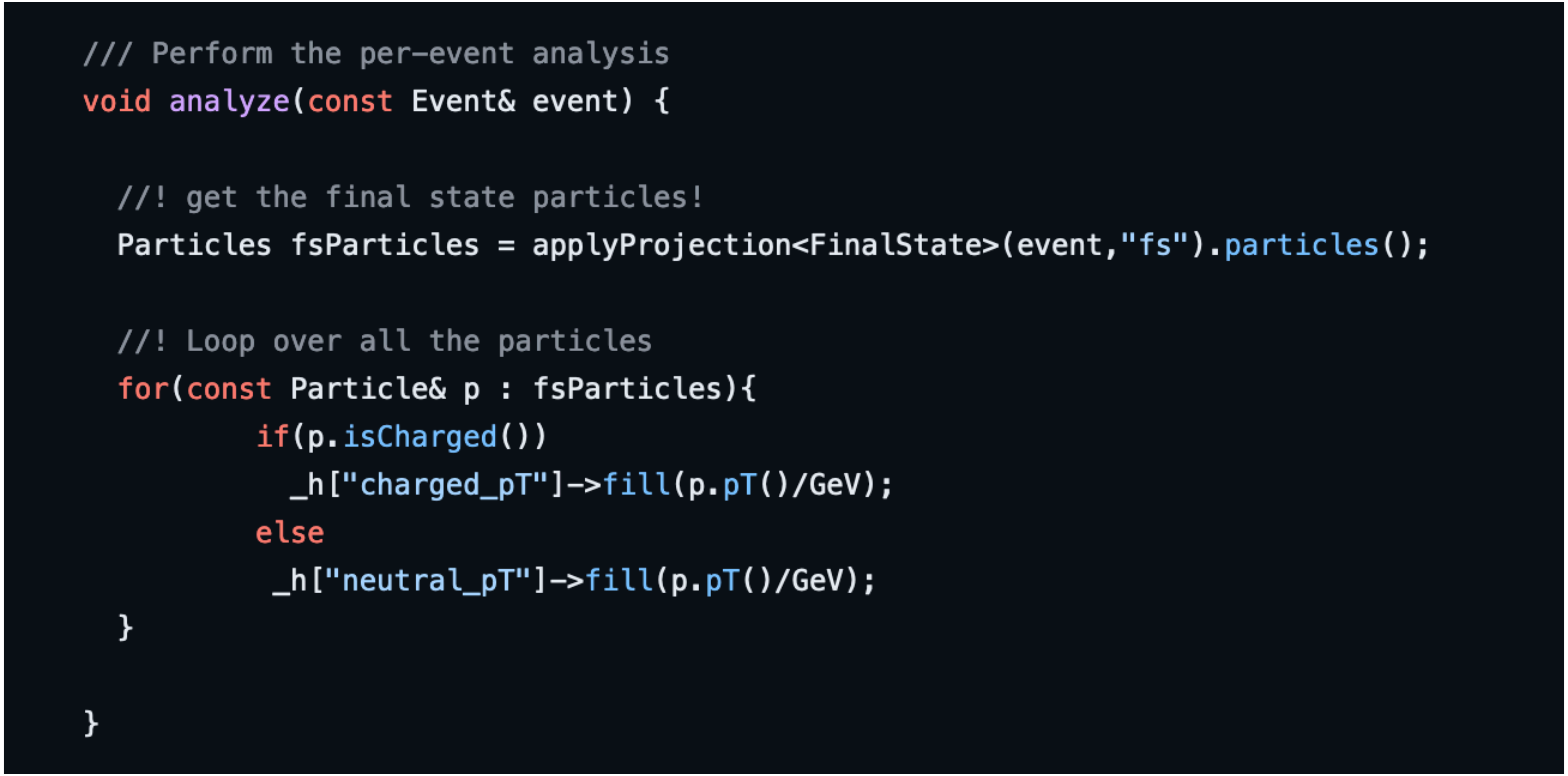
- All analysis in rivet derive from a base class! This has a lot of useful information within it!

Initialization method - called once at the start



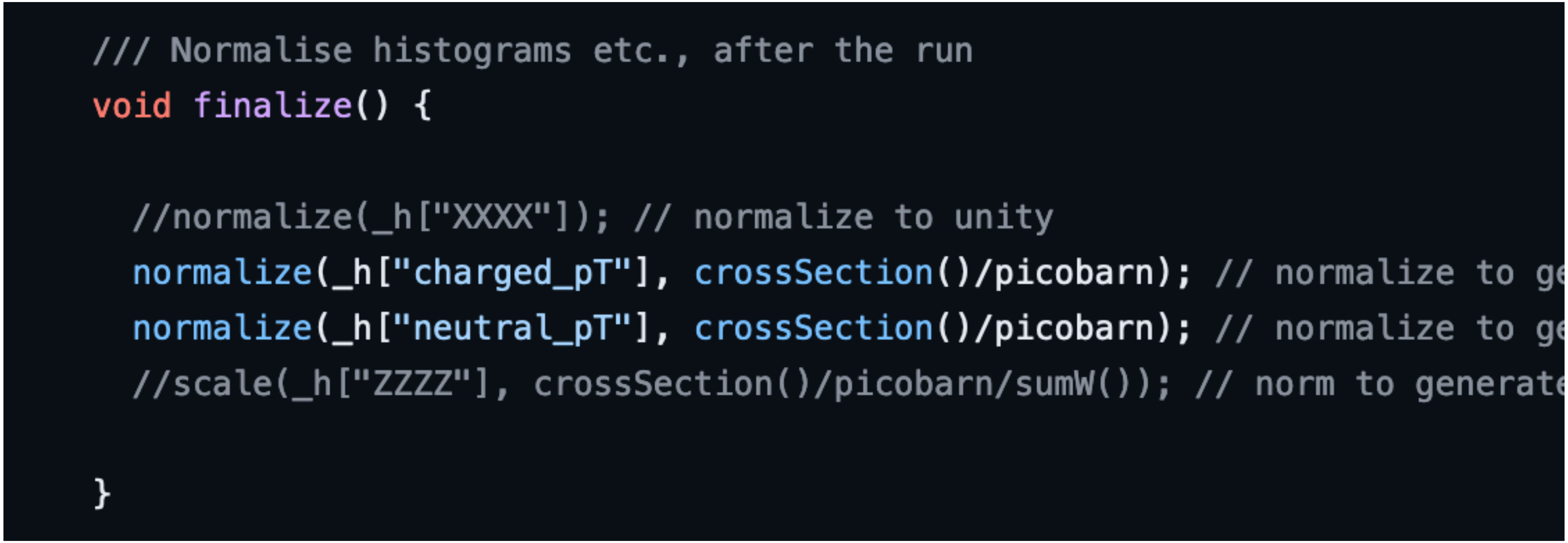
```
/// Book histograms and initialise projections before the run
void init() {

  // Initialise and register projections

  // The basic final-state projection:
  // all final-state particles within
  // the given eta acceptance
  const FinalState fs(Cuts::abseta < 1.0);
  declare(fs, "fs");

  // Book histograms
  book(_h["charged_pT"], "charged_pT", 60, 0.0, 30.0);
  book(_h["neutral_pT"], "neutral_pT", 60, 0.0, 30.0);

}
```

Analyze - runs for each event!



```
/// Perform the per-event analysis
void analyze(const Event& event) {

  //! get the final state particles!
  Particles fsParticles = applyProjection<FinalState>(event,"fs").particles();

  //! Loop over all the particles
  for(const Particle& p : fsParticles){
          if(p.isCharged())
            _h["charged_pT"]->fill(p.pT()/GeV);
          else
           _h["neutral_pT"]->fill(p.pT()/GeV);
  }

}
```

Finalize - called at the end!



```
/// Normalise histograms etc., after the run
void finalize() {

  //normalize(_h["XXXX"]); // normalize to unity
  normalize(_h["charged_pT"], crossSection()/picobarn); // normalize to ge
  normalize(_h["neutral_pT"], crossSection()/picobarn); // normalize to ge
  //scale(_h["ZZZZ"], crossSection()/picobarn/sumW()); // norm to generate

}
```

On towards the Analysis then!