Workflow

Slow-ish development:

C++/ROOT:

- +ROOT, RooFit
- +pre-existing expertise
- C++ reading library

JAVA/COATJAVA:

- +native COATJAVA synergy (HIPO)
- +cross-platform, multi-threading, memory management

Fast-ish development:

Python/PyROOT/etc:

- +near native access to ROOT
- +brevity/readability
- +WORLDWIDE USAGE
- -based on C++ reader library, i.e. lags from most recent development
- +easy multi-threading

Groovy/COATJAVA

- +brevity/readability
- +native access to COATJAVA
- worldwide usage
- +superior collections processing capabilities (sugar, IMHO)
- +easy multi-threading

Workflow

Stage 1: reducing cooked data (trains, DST) to custom skims

- Groovy: fast prototyping

Stage 2: reducing custom skims to plots, class

- Groovy to save into ROOT files, hists etc

Stage 3: aggregated data analysis

- Python with PyROOT modules and 3rd party ML libs

Summary

Reactions:

- $ep \to ep\gamma$
- $ep \to ep\pi^0$
- $ep \to ep\eta$
- $ep \to ep\phi$
- π^0, π^+, π^- SIDIS

Commmon particles:

- electron ID
- proton ID
- photon ID
- π^+, π^-, K^+, K^- IDs

- Develop common PID cuts (MC and DATA)
- Quality monitoring (custom timelines)
- Produce intermediate skims
  (HIPO4 event tagging, custom wagons)

- Event selection and further analysis:
  - exclusive cuts?
  - kinematic fitter?
  - machine learning algorithms?