General Purpose SIDIS Analysis Software

Github: https://github.com/c-dilks/largex-eic

- Dependencies: ROOT and Delphes
- Follow setup instructions in README.md
- Important: several scripts require environment variables setup with `source env.sh`

Example fast simulation ROOT file from Delphes:

https://duke.box.com/s/0x83y9uz56vafvm9hxige7efov9z8taw

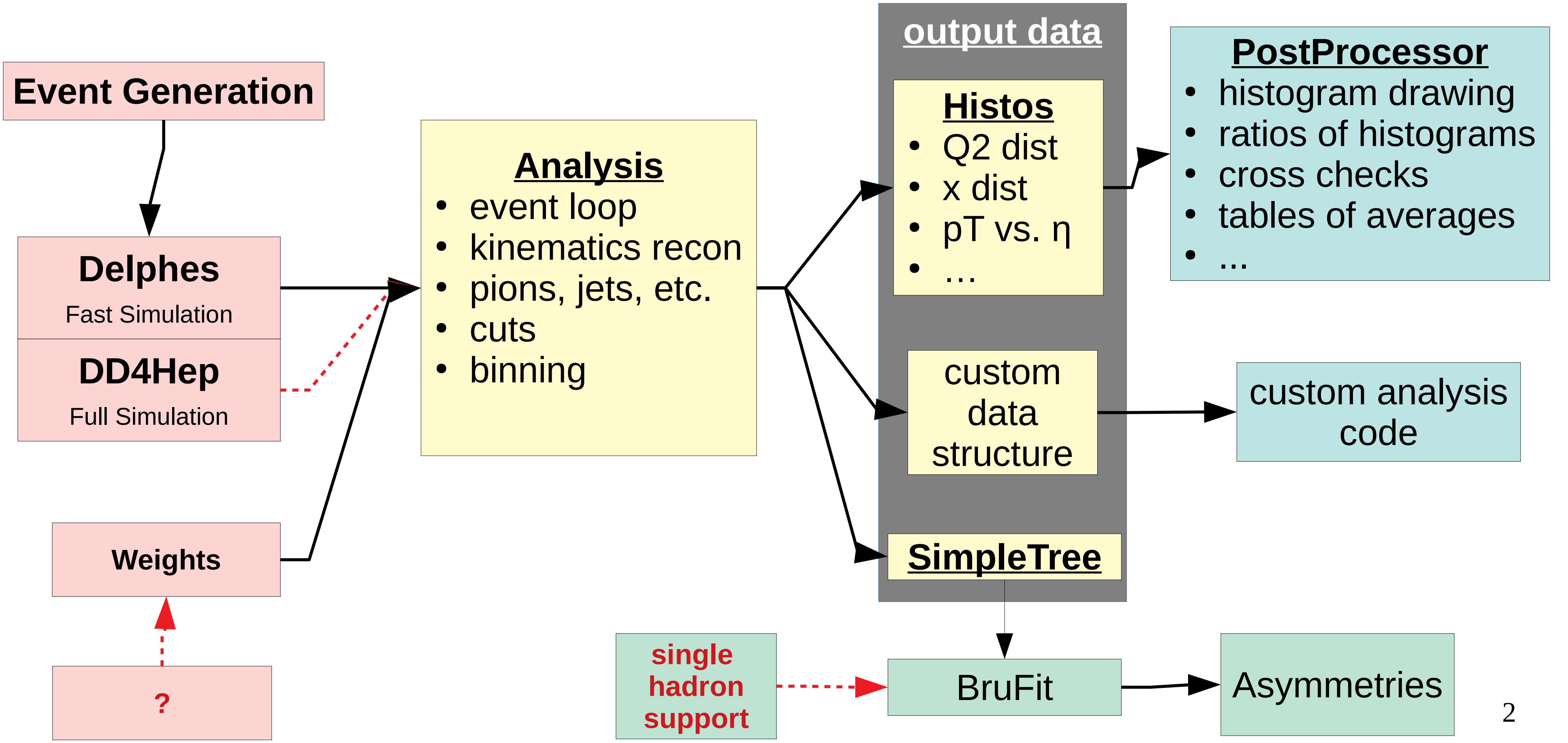
Download the ROOT file and store it in `largex-eic/datarec`

A hepmc file from Pythia is also provided, which you can run through Delphes (it is not the same data set as the example ROOT file)

General purpose SIDIS simulation analysis software

https://github.com/c-dilks/largex-eic

⇢ to do
→ done



Event Generation

Delphes
Fast Simulation

DD4Hep
Full Simulation

Weights

?

Analysis
- event loop
- kinematics recon
- pions, jets, etc.
- cuts
- binning

output data

Histos
- Q2 dist
- x dist
- pT vs. η
- …

custom data structure

SimpleTree

PostProcessor
- histogram drawing
- ratios of histograms
- cross checks
- tables of averages
- …

custom analysis code

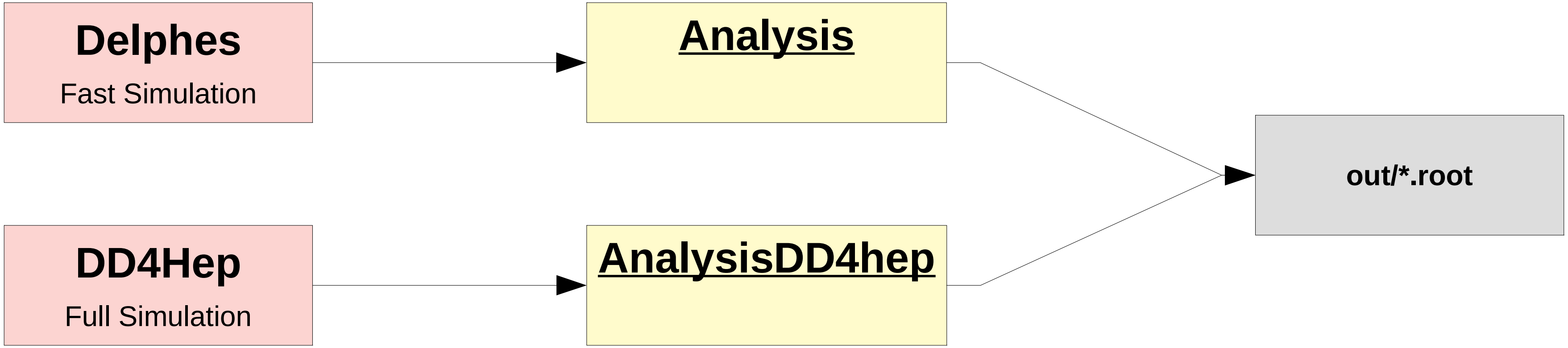
single hadron support

BruFit

Asymmetries

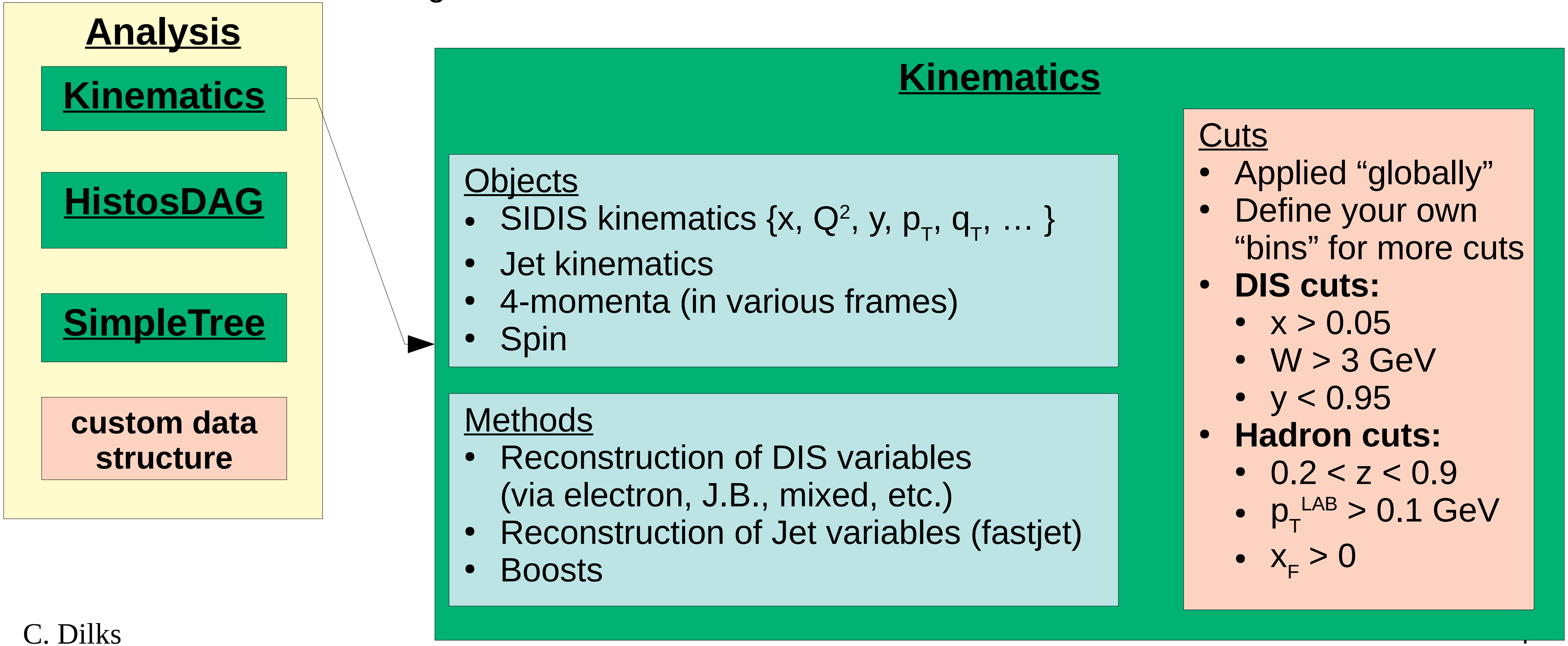
Analysis Classes

- Responsible for reading the fast/full simulation data, and producing a variety of output data structures written to ROOT files in out/
- Classes are steered by macros, which allow definitions of binning schemes and other settings
- Two classes currently exist, one for fast simulations and another for full simulations (we may eventually refactor the inheritance, e.g., make an Analysis base class)
- Fast simulation class is working, but full simulation class is still in progress (see *fullsim* branch)



Analysis class details: Kinematics

- A class that contains all of the kinematics reconstruction methods
- There are 2 instances: one for the reconstructed particle, and another for the true (generated) particle
- When reading each particle in the event loop Kinematics calculations will be performed and variables will be set with the resulting values



Analysis

- **Kinematics**
- **HistosDAG**
- **SimpleTree**
- **custom data structure**

Kinematics

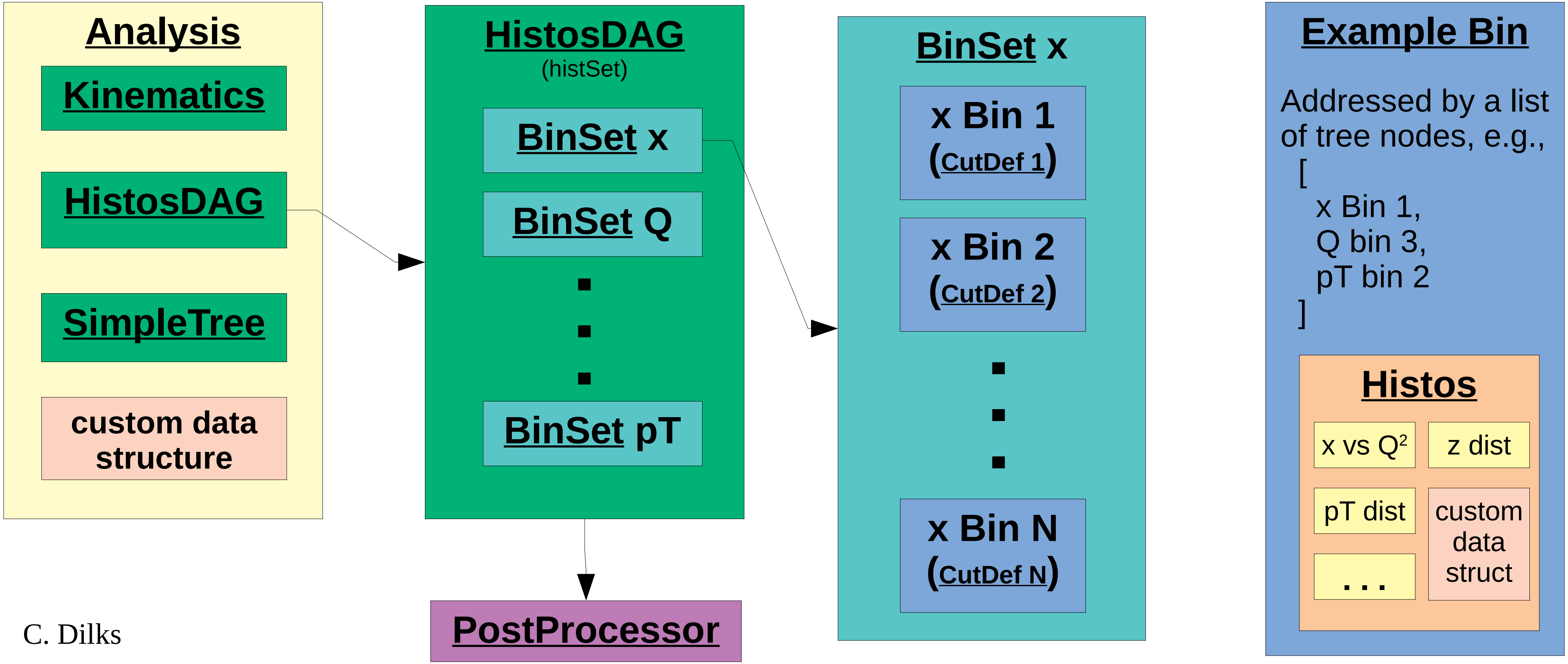
Objects
- SIDIS kinematics {x, Q^2, y, p_T, q_T, … }
- Jet kinematics
- 4-momenta (in various frames)
- Spin

Methods
- Reconstruction of DIS variables (via electron, J.B., mixed, etc.)
- Reconstruction of Jet variables (fastjet)
- Boosts

Cuts
- Applied “globally”
- Define your own “bins” for more cuts
- **DIS cuts:**
  - $x > 0.05$
  - $W > 3$ GeV
  - $y < 0.95$
- **Hadron cuts:**
  - $0.2 < z < 0.9$
  - $p_T^{LAB} > 0.1$ GeV
  - $x_F > 0$

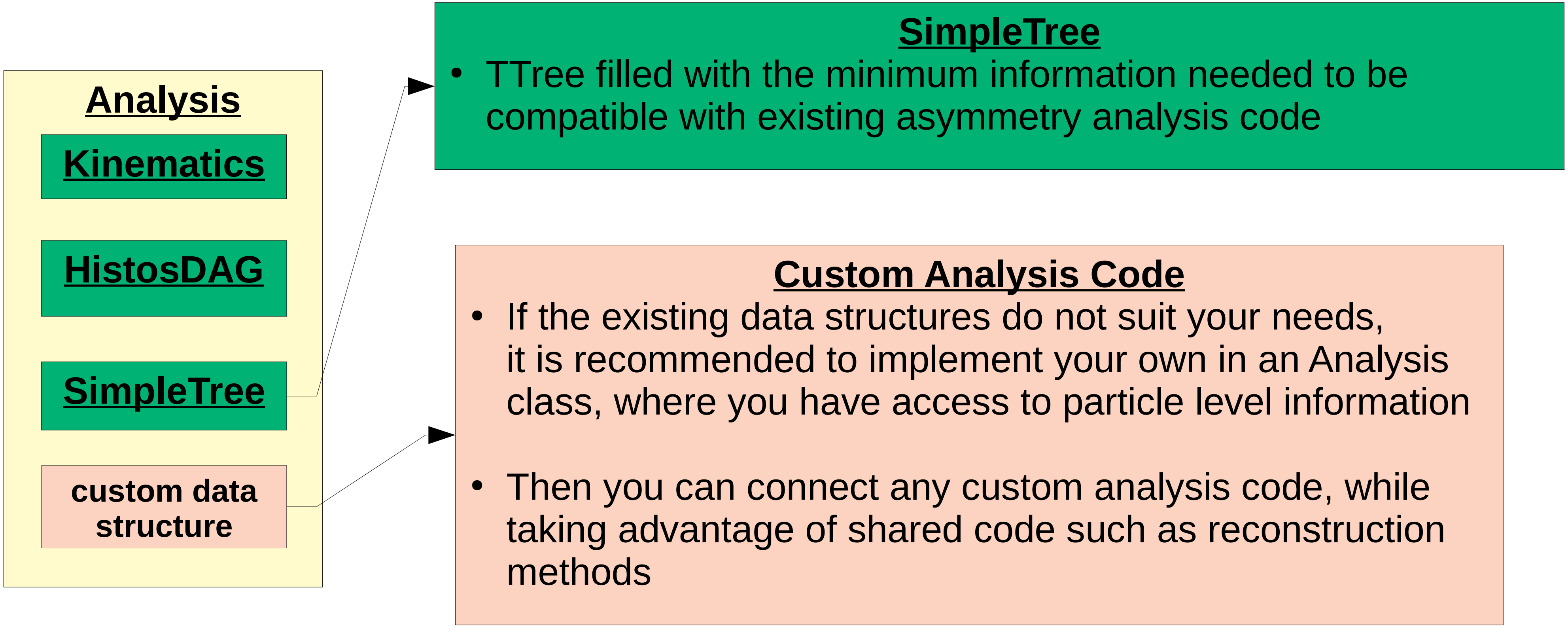
Analysis class details: Histos DAG (Directed Acyclic Graph)

- Tree of multidimensional bins, where each bin contains a set of histograms, a “Histos” object
- Implementation mostly done, but still needs more testing and documentation
- **Current version: Analysis::histSet – a multi-dimensional array of Histos objects**



C. Dilks

Analysis class details: Others



Analysis

Kinematics

HistosDAG

SimpleTree

custom data structure

SimpleTree

- TTree filled with the minimum information needed to be compatible with existing asymmetry analysis code

Custom Analysis Code

- If the existing data structures do not suit your needs, it is recommended to implement your own in an Analysis class, where you have access to particle level information
- Then you can connect any custom analysis code, while taking advantage of shared code such as reconstruction methods

General Procedure

underlined objects are classes (or macros)

- Choose your bins for each variable you are interested in; each bin of some variable x is specified by a CutDef, in a variety of ways:
  - Range: $a<x<b$
  - CenterDelta: $|x-a|<b$
  - Minimum: $x>a$
  - Maximum: $x<a$
  - No cut (full range of x)

- Bins of a particular variable x are collected into a BinSet (also called ‘bin scheme’), where you can either:
  - Manually define each bin
    - Example: [Bin1: $x<0.2$] [Bin2: $0.2<x<0.5$] [Bin3: $x>0.5$]
    - Example (note that overlapping bins are allowed!): [Bin1: full y] [Bin2: $y>0.03$] [Bin3: $y>0.05$]
  - Define an axis of bins: N bins between a and b
    - equal widths in linear scale
    - equal widths in logarithmic scale
    - any custom TAxis
    - Example: (x,$Q2$) bins with equal width in log scale

- **User specifies all Bins and BinSets in an analysis macro**

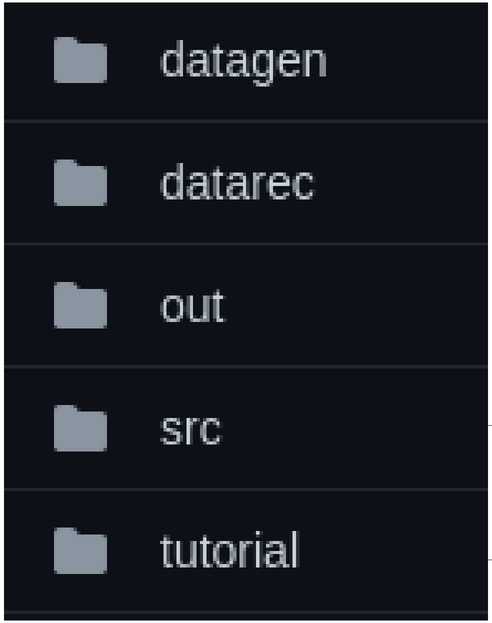
General Procedure

- Each multidimensional bin contains a Histos object
  - Set of user-defined histograms (1,2, or 3D)
  - Set of CutDefs associated with this bin
  - Settings for histograms (e.g., log scale drawing)
  - You are welcome to add your own data structures to the Histos class (or even inherit from it)

- No limit to number of BinSets, i.e. dimensions of your binning (*current histSet prototype has limits*)
  - You can only choose bins which are “available” in the Analysis class
  - Careful of the curse of dimensionality

- BinSet and Histos are streamable to ROOT files, which will happen automatically from an analysis macro
  - Analyze these with the PostProcessor class, which can do a variety of tasks:
    - Draw histograms in a specific format
    - Take ratios of histograms from two different bins
    - Dump averages of histograms for a set of bins and make a table
    - Add your own algorithms here
  - **PostProcessor is driven by a postprocessor macro, providing full bin-looping flexibility**

Code



datagen

datarec

out

src

tutorial

generated data, reconstructed data, analysis and postprocessing output

Class definitions

Tutorial Macros

Contributions

Git Workflow

- Write some code
- New branch?
  - git checkout -b <newBranch>
- git add <code>
- git commit -m “add feature xyz”
- Push:
  - New branch?
    - git push -u origin <newBranch>
    - open new draft pull request (PR), by following the URL that appears; mark as draft if you plan to make more commits; request should be from newBranch to main branch
  - Not a new branch?
    - git push
- Repeat, pushing more commits to this branch and PR until ready for merge
- Mark PR as ready (and notify others for review+merge)

use “Issues” for bug reporting, or feature ideas; you can also link a PR to an issue

keep up-to-date with main branch:

- git pull (if on main)
- git rebase or merge (to bring updates in main to your own branch)
- “Insights” tab → “Network” → view branch topology