Iguana

Implementation Guardian of Analysis Algorithms

https://github.com/JeffersonLab/iguana

Jefferson Lab

clas

Christopher Dilks

Purpose

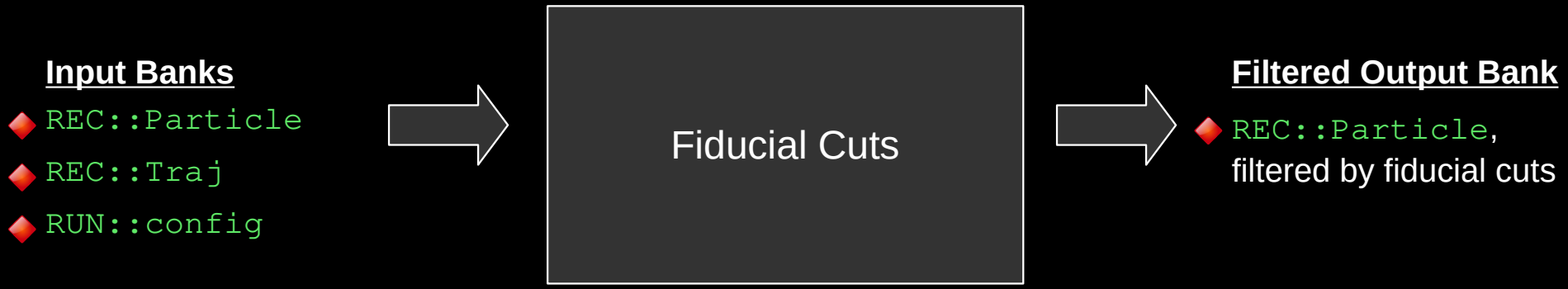
Encapsulate, centralize, and preserve common needs in **Iguana Algorithms**

- Methodology preservation (*cf*. data preservation efforts)
- Reproducibility
- Allow for focus on the important parts of an analysis
- Centralization increases the number of code reviewers
  - Lower probability of bugs
  - But if there are bugs, they impact *all* users
  - Validation is critical
- More details from the last CLAS Collaboration Meeting: https://indico.jlab.org/event/829/contributions/14072/attachments/10720/16241/iguana.pdf

What do we mean by Algorithm?

We define “Algorithm” as a function that maps a set of input banks to a set of output banks

Filter Algorithm: accepts/rejects rows of bank(s)



What do we mean by Algorithm?

Transformer Algorithm: modifies bank(s)

Input Banks
- `RUN::config`
- `REC::Particle`
- `REC::Particle::Sector` (created by SectorFinder algorithm)

→ Momentum Corrections →

Modified Output Bank
- `REC::Particle`, with corrected momenta

Creator Algorithm: creates new bank(s)

Input Banks
- `REC::Particle`

→ Inclusive Kinematics Reconstruction →

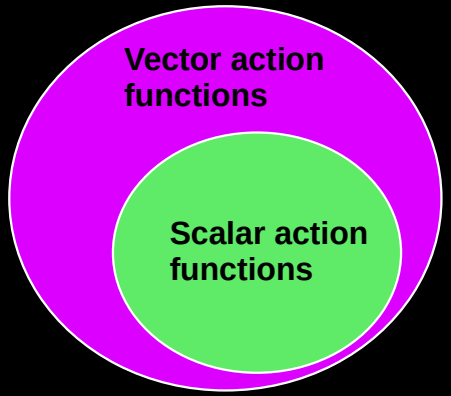
New Output Bank
- `physics::InclusiveKinematics`, with x, Q^2, W, etc.

Action Functions

- Algorithms use HIPO C++ `hipo::bank` objects as their input/output
  - https://github.com/gavalian/hipo
  - However, not all users use these objects → supported by “action functions”

- Action Functions
  - Definition: a function with *simple* parameters and return values, providing algorithm usage for *all* users
  - All Iguana algorithms should provide action functions
  - The main algorithm function `Run(std::vector<hipo::bank>&)` calls action functions

- Types of Action Functions
  - Scalar action functions: parameter types such as `int`, `double`, `string`
  - Vector action functions: parameter types are lists, e.g. `std::vector<int>`
    - May call the corresponding scalar action function iteratively
    - **TODO**: we haven’t yet written *any* vector action functions!
  - Properties:
    - Any scalar action function should have a corresponding vector action function
    - Not all vector action functions can have a corresponding scalar action function
      - Some algorithms can *only* have vector action functions



Iguana Usage Options

- **HIPO C++ API:** https://github.com/gavalian/hipo
  - The “primary” API, for the purpose of generalization, testing, infrastructure, etc.

- **Action functions:**
  - Available languages: algorithms are in C++, and bindings provide usage from other languages:
    - **C++**
      - “Native” (no bindings), since algorithms are in C++
    - **Python**
      - Via cppyy (the same as PyROOT; may be changed in the future!)
      - See HIPOPy example: https://github.com/JeffersonLab/iguana/blob/main/bind/python/iguana_ex_python_hipopy.py
    - **Fortran**
      - Using ISO_C_BINDING
      - TODO: missing several action functions (since we are still testing the design)
    - **Java** to be added soon
  - Action functions support **clas12root** usage
  - Action functions support **HIPO dataframes** usage
    - (room for improvement / user-friendliness)

Where can I find Iguana?

- On `ifarm`
  - `module avail iguana`
  - `module load iguana/0.7.0` (the current default)
- Build it yourself
  - Follow https://github.com/JeffersonLab/iguana/blob/main/doc/setup.md
  - All dependencies are available on `ifarm`
    - A bit more work if you want to build on your personal computer

Available Algorithms

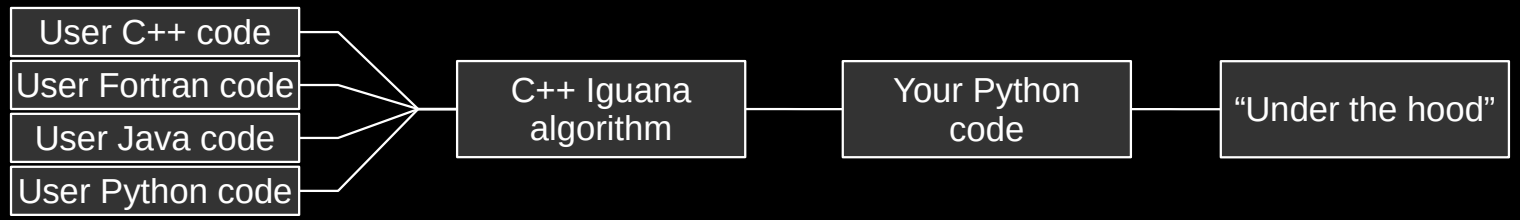
Algorithm	Maintainer	Status
clas12::FTEnergyCorrection	Asli Acar	Algorithm done; validation in progress
clas12::FiducialFilter	Gregory Matousek (for Stefan Diehl)	Done; in version **iguana/0.7.1** (not yet on ifarm)
clas12::LeptonIDFilter	Mariana Tenorio	Algorithm in progress
clas12::MomentumCorrection	Chris Dilks (for Richard Capobianco)	Done; validation in progress
clas12::PhotonGBTFilter	Gregory Matousek	Done
clas12::SectorFinder	Richard Tyson	Done, but needs a Validator
clas12::ZVertexFilter	Richard Tyson	Done, but needs better config and validator
physics::InclusiveKinematics	Chris Dilks	Done

Note: we have additional, example algorithms, such as clas12::EventBuilderFilter

Coming up next: presentations by the algorithm maintainers

Can I put an algorithm in Iguana?

- Yes, please!
- If you need a new dependency, ask and we'll try to add it
- Your algorithm *must* be in C++, so that it integrates well with other existing algorithms, tests, and language bindings
  - If your algorithm *cannot* be ported to C++, then we'll need a simple C++ "wrapper" algorithm that would call your code and handle its output
    - For example, if you *really* need your algorithm to be in Python:



Contributions are Welcome

- We follow the usual GitHub workflow
  - Issues: planned work, bugs, feature requests, …
  - Pull Requests: new code, fixed code, …
- You may also contact the CLAS Software Group
  - Via email
  - My email: `dilks AT jlab DOT org`
  - Post in the CLAS Discourse: https://clas12.discourse.group/
- New algorithms and ideas are welcome!

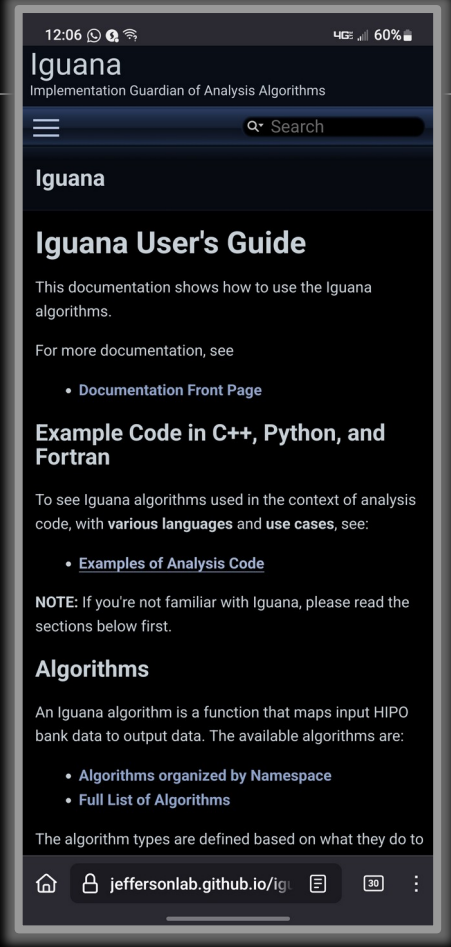
https://github.com/JeffersonLab/iguana

Iguana Walk-through

- Now let's go through the Iguana documentation, including:
  - Algorithms' documentation
  - C++ examples
    - Consumers: CMake, Makefile, meson
  - Python examples
  - Fortran examples

  https://github.com/JeffersonLab/iguana/blob/main/README.md
    - See in particular, the **Iguana User's Guide**

- clas12root usage in the **next slides**



NOTE: whereas these slides are "frozen" and will not be updated, the Iguana documentation is a "living document" and will receive frequent updates

Using Standard Iguana in Clas12Root

- Currently recommended use of iguana is directly via the Action functions
- Iguana algorithms with appropriate Action functions can be used directly

```
#include <iguana/algorithms/clas12/MomentumCorrection/Algorithm.h>
```

INCLUDE HEADER

```
clas12root::HipoChain chain;
chain.Add("file_*.hipo");
chain.SetReaderTags({0});  //create clas12reader with just tag 0 events
auto config_c12=chain.GetC12Reader();
config_c12->addExactPid(11,1);    //exactly 1 electron
config_c12->addExactPid(211,1);    //exactly 1 pi+
config_c12->addExactPid(-211,1);    //exactly 1 pi-
```

START CLAS12ROOT AS NORMAL

```
// create the algorithms
iguana::clas12::MomentumCorrection algo_momentum_correction;
// set log levels
algo_momentum_correction.SetOption("log", "debug");
// start the algorithms
algo_momentum_correction.Start();
```

CREATE IGUANA ALGORITHM

```
//loop over all events
while (chain.Next()){
  ...
  // get the corrected momentum using info from
  // clas12root region_particle(p) and run_config
  auto [px, py, pz] = algo_momentum_correction.Transform
    (p->getPx(),p->getPy(),p->getPz(),p->getSector(),
     p->getPid(),c12->runconfig()->getTorus());
```

CALL ALGORITHM IN EVENT LOOP

Jefferson Lab

Slide from Derek Glazier

Clas12Root - Iguana Interface

- Or more simply use the interface

```
// create the chosen algorithms
clas12root::Iguana ig;
ig.SetClas12(chain.C12ptr());//connect to clas12reader
ig.GetTransformers().Use("clas12::MomentumCorrection");
ig.GetTransformers().Use("clas12::FTEnergyCorrection");
ig.GetFilters().Use("clas12::ZVertexFilter");
ig.GetCreators().Use("physics::InclusiveKinematics");

ig.SetOptionAll("log", "debug");
ig.Start();
```

Choose iguana algorithms that have been added to interface

```
//declare 4-vector objects
FourVector p4beam(0,0,10.6,10.6);
FourVector p4target(0,0,0,MProton);
FourVector p4el(0,0,0,MElectron);
FourVector p4pip(0,0,0,MPion);
FourVector p4pim(0,0,0,MPion);
```

```
while ( chain.Next() ){
  //get some region particles I wish to analyse
  auto electron=c12->getByID(11)[0];
  auto pip=c12->getByID(211)[0];
  auto pim=c12->getByID(-211)[0];
  //apply all configured filters
  if( !(ig.GetFilters().doAllFilters({electron,pip,pim})) ) {
    continue;
  }
```

Filters applied on these particles

Corrected 4-vectors for given particles

```
//correct momentum and get 4-vectors
ig.GetTransformers().doAllCorrections({electron,pip,pim},{&p4el,&p4pip,&p4pim});

//calculate inclusive kinematics
auto kine = ig.GetCreators().doInclusiveKinematics(electron);
auto corrkine = ig.GetCreators().doInclusiveKinematics(p4el);
```

Inclusive kinematics for EB e-

Inclusive kinematrics for corrected e-

Note:

This is a Clas12root *wrapper* of the more *commonly used* Iguana algorithms

- Provides tighter integration with Clas12root's design → more optimized for Clas12root users
- Not maintained by Iguana developers, therefore may not be up-to-date
- If you want to use all of Iguana, use it directly (see previous slide)

Jefferson Lab

Slide from Derek Glazier

Next up: Algorithm Presentations

2:40 PM

Analysis tools

Speaker: Christopher Dilks (Jefferson Lab)

IGUANA status and usage examples

Speaker: Christopher Dilks (Jefferson Lab)

FT energy correction

Speaker: Asli Acar

Momentum corrections

Speaker: Christopher Dilks (Jefferson Lab)

Inclusive kinematics

Speaker: Christopher Dilks (Jefferson Lab)

3:30 PM → 4:00 PM **Coffee Break**

4:00 PM

Analysis Tool Cont'ed

Fiducial filter

Speaker: Gregory Matousek (Duke University)

Photon GBT filter

Speaker: Gregory Matousek (Duke University)

Lepton ID filter

Speaker: Mariana Tenorio Pita (Old Dominion University)

Sector finder

Speaker: Richard Tyson (Jefferson Lab)

Z-vertex filter

Speaker: Richard Tyson (Jefferson Lab)