Iguana

Implementation Guardian of Analysis Algorithms

image source

https://github.com/JeffersonLab/iguana

Christopher Dilks
CLAS Collaboration Meeting

12 March 2024

The Problem

- Diverse set of Fiducial Cuts implementations
  - Original versions In C++
  - Ports to Java, Python, and Fortran
  - Integrations in common frameworks, e.g., Chanser
  - Integrations in user analysis code

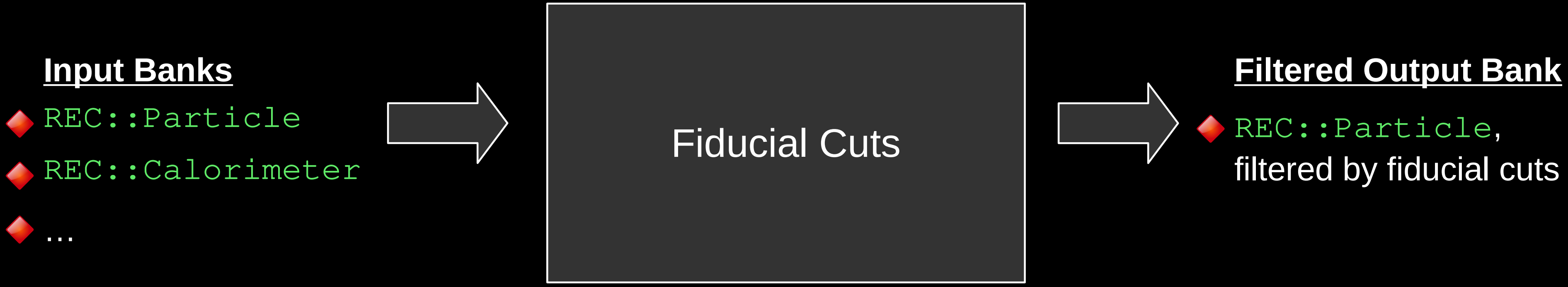
  → Highlights the importance of cross checking

- Fiducial cuts are among a set of common “things” → in general, “algorithms”
  - Many analyzers need to use them, with varying configurations (e.g., tight vs. loose cuts)
  - Other common algorithms include Momentum Corrections and PID Refinements
  - RG-A analyses refer to the common RG-A Analysis Note, and some common algorithms have been rewritten independently by *each* analyzer

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

For example, a **Filter**



Input Banks

- `REC::Particle`
- `REC::Calorimeter`
- …

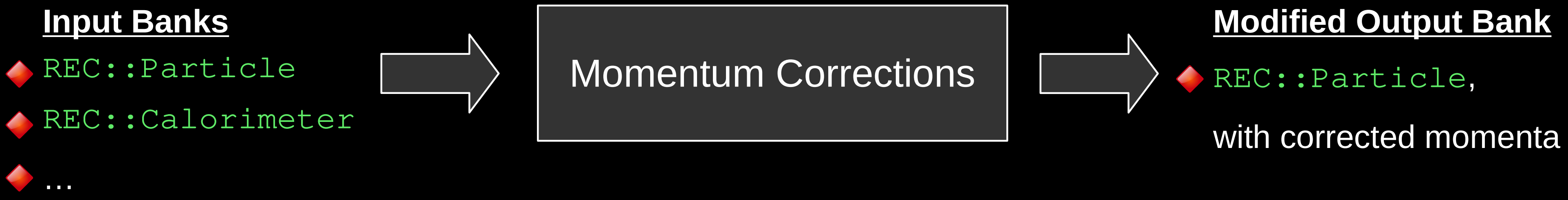
Fiducial Cuts

Filtered Output Bank

- `REC::Particle`, filtered by fiducial cuts

What do we mean by Algorithm?

Or a **Modifier**



Input Banks

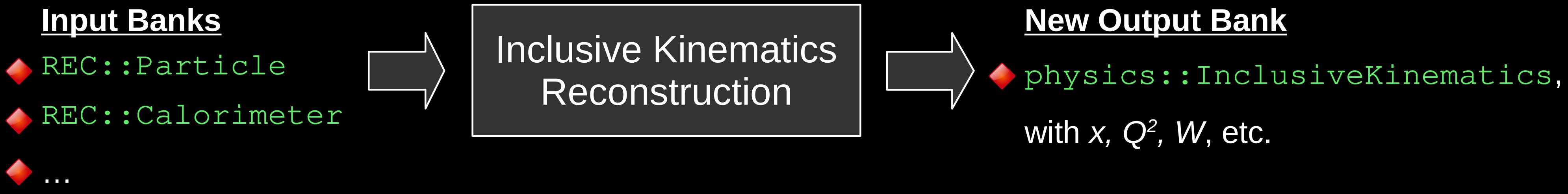
- `REC::Particle`
- `REC::Calorimeter`
- …

Momentum Corrections

Modified Output Bank

- `REC::Particle`, with corrected momenta

Or a **Creator**



Input Banks

- `REC::Particle`
- `REC::Calorimeter`
- …

Inclusive Kinematics Reconstruction

New Output Bank

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

A Solution: Centralizing the Algorithms

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

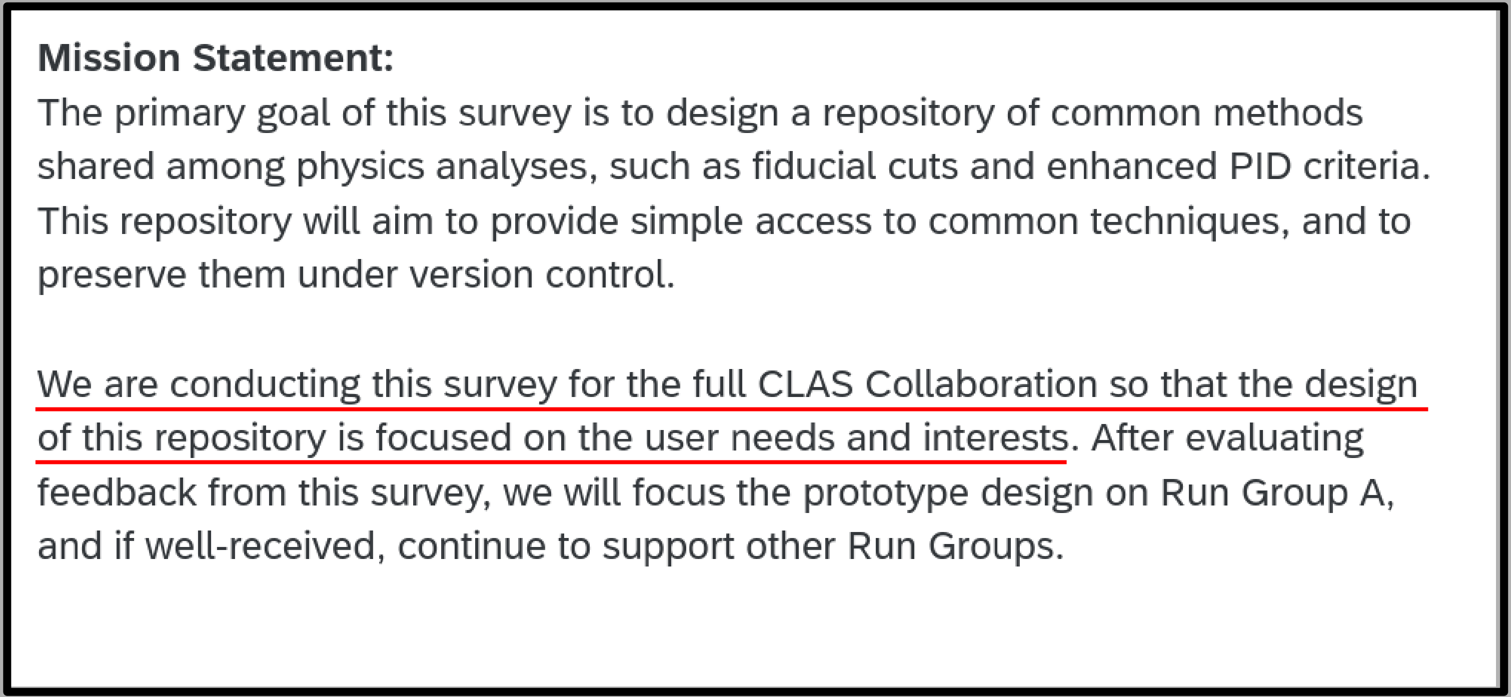
Why the name?

Iguana

Implementation Guardian of Analysis Algorithms

- Unique namespace
- Easier to reference, as opposed to a generic name such as `clas12-common-analysis-algorithms`
- Naming things is hard, so why not be creative?

User-centered design → Software Survey

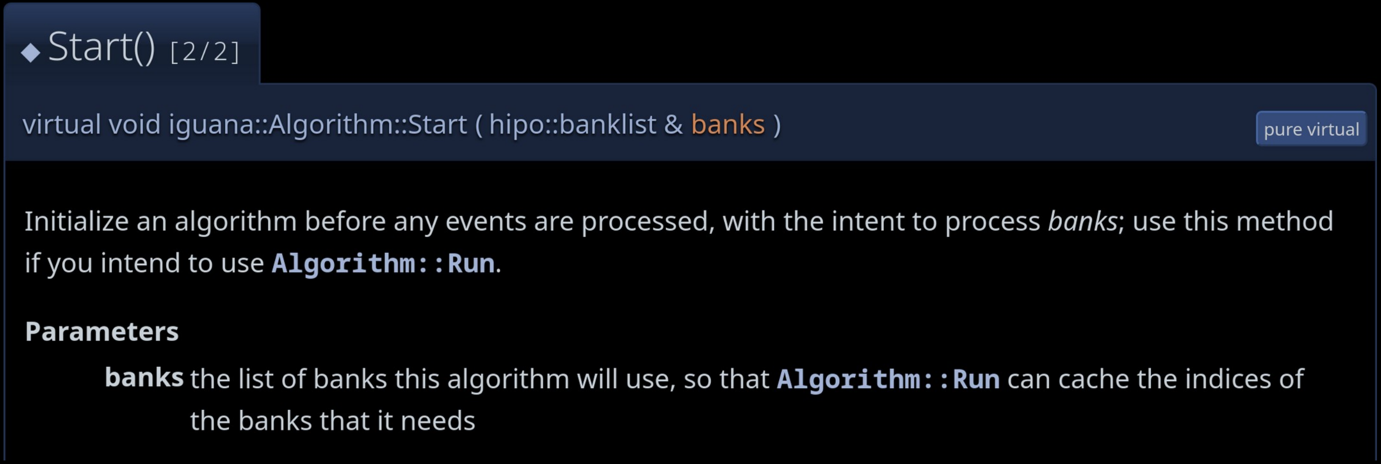


Mission Statement:
The primary goal of this survey is to design a repository of common methods shared among physics analyses, such as fiducial cuts and enhanced PID criteria. This repository will aim to provide simple access to common techniques, and to preserve them under version control.

We are conducting this survey for the full CLAS Collaboration so that the design of this repository is focused on the user needs and interests. After evaluating feedback from this survey, we will focus the prototype design on Run Group A, and if well-received, continue to support other Run Groups.

Algorithm Methods

All algorithms must implement the following 3 methods:



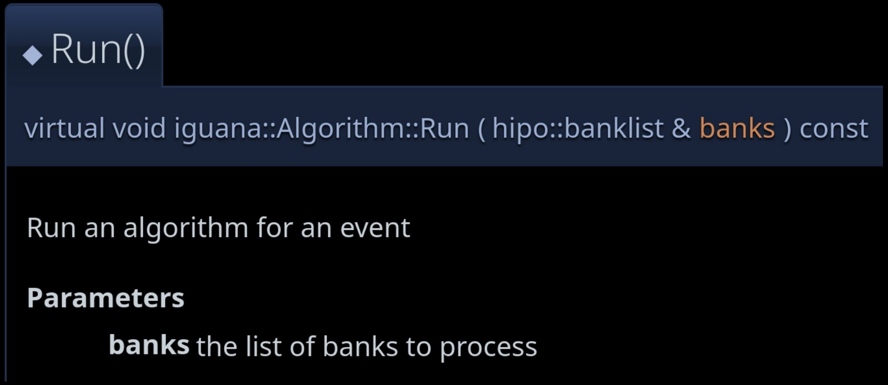
◆ Start() [2/2]

virtual void iguana::Algorithm::Start (hipo::banklist & banks) pure virtual

Initialize an algorithm before any events are processed, with the intent to process *banks*; use this method if you intend to use **`Algorithm::Run`**.

Parameters

banks the list of banks this algorithm will use, so that **`Algorithm::Run`** can cache the indices of the banks that it needs



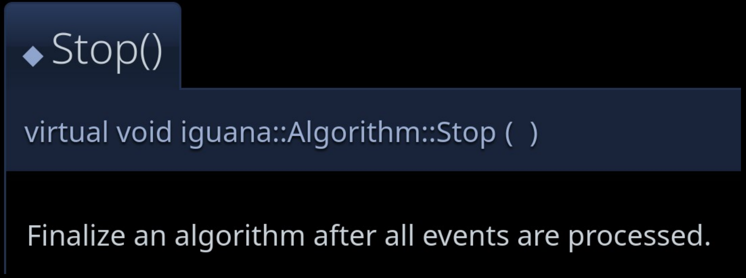
◆ Run()

virtual void iguana::Algorithm::Run (hipo::banklist & banks) const

Run an algorithm for an event

Parameters

banks the list of banks to process



◆ Stop()

virtual void iguana::Algorithm::Stop ( )

Finalize an algorithm after all events are processed.

Algorithm `Run` Method

Hold on… I don't use HIPO banks!

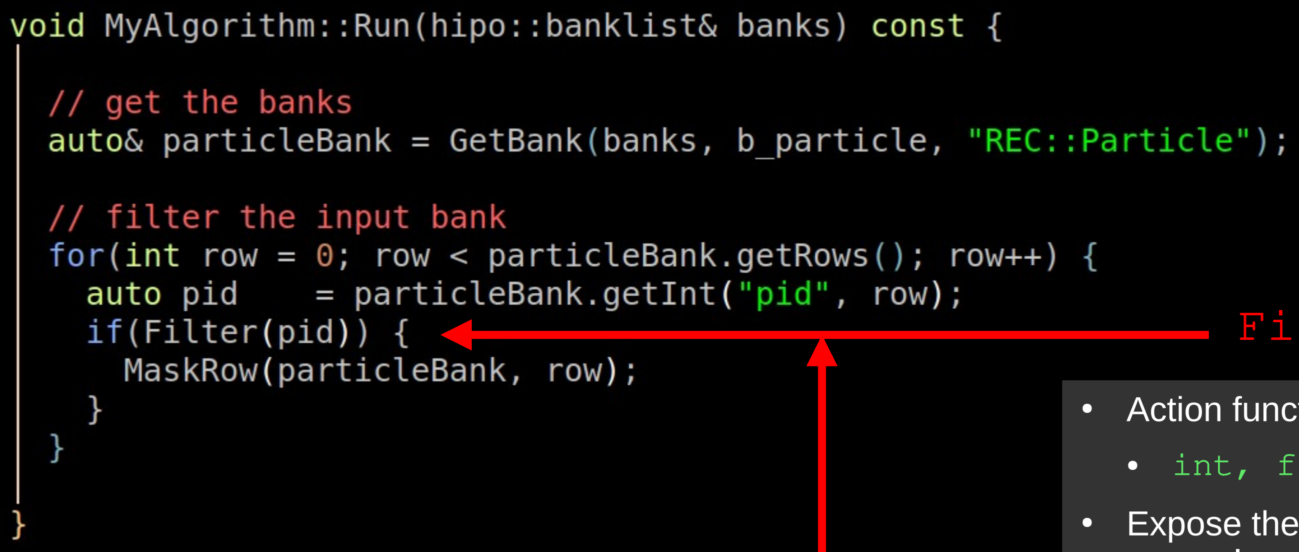
Or:

- I use HIPO banks, but not the C++ hipo banks from gavalian/hipo
- I use HIPO banks with a different library or language
- I use data frames from gavalian/hipo or elsewhere
- I have info from the banks (bank row elements)
- The relevant part of my code is not in C++

A goal of Iguana is to support *diverse* analyses, including all of the above use cases

- Action functions permit operation on bank rows
- Language bindings (will) permit usage from other programming languages

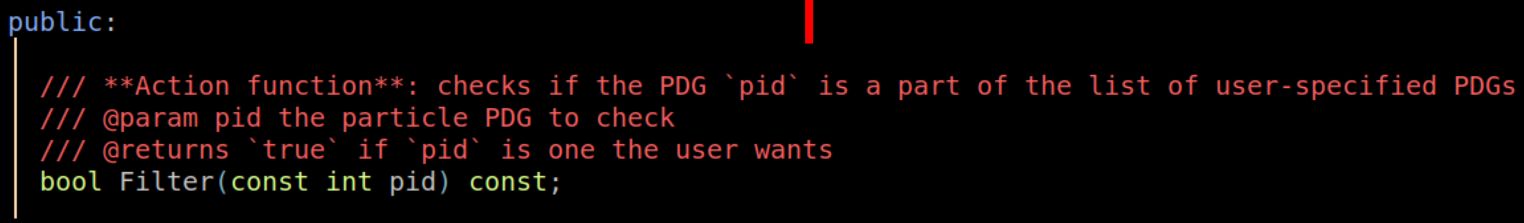
Action Functions



```
void MyAlgorithm::Run(hipo::banklist& banks) const {

  // get the banks
  auto& particleBank = GetBank(banks, b_particle, "REC::Particle");

  // filter the input bank
  for(int row = 0; row < particleBank.getRows(); row++) {
    auto pid    = particleBank.getInt("pid", row);
    if(Filter(pid)) {
      MaskRow(particleBank, row);
    }
  }

}
```

`Filter` is an Action Function

- Action functions operate on bank row *elements*
  - `int, float, double,` etc.
- Expose the *primary* algorithm functionality to the general user
- Unique to each algorithm; not standardized...

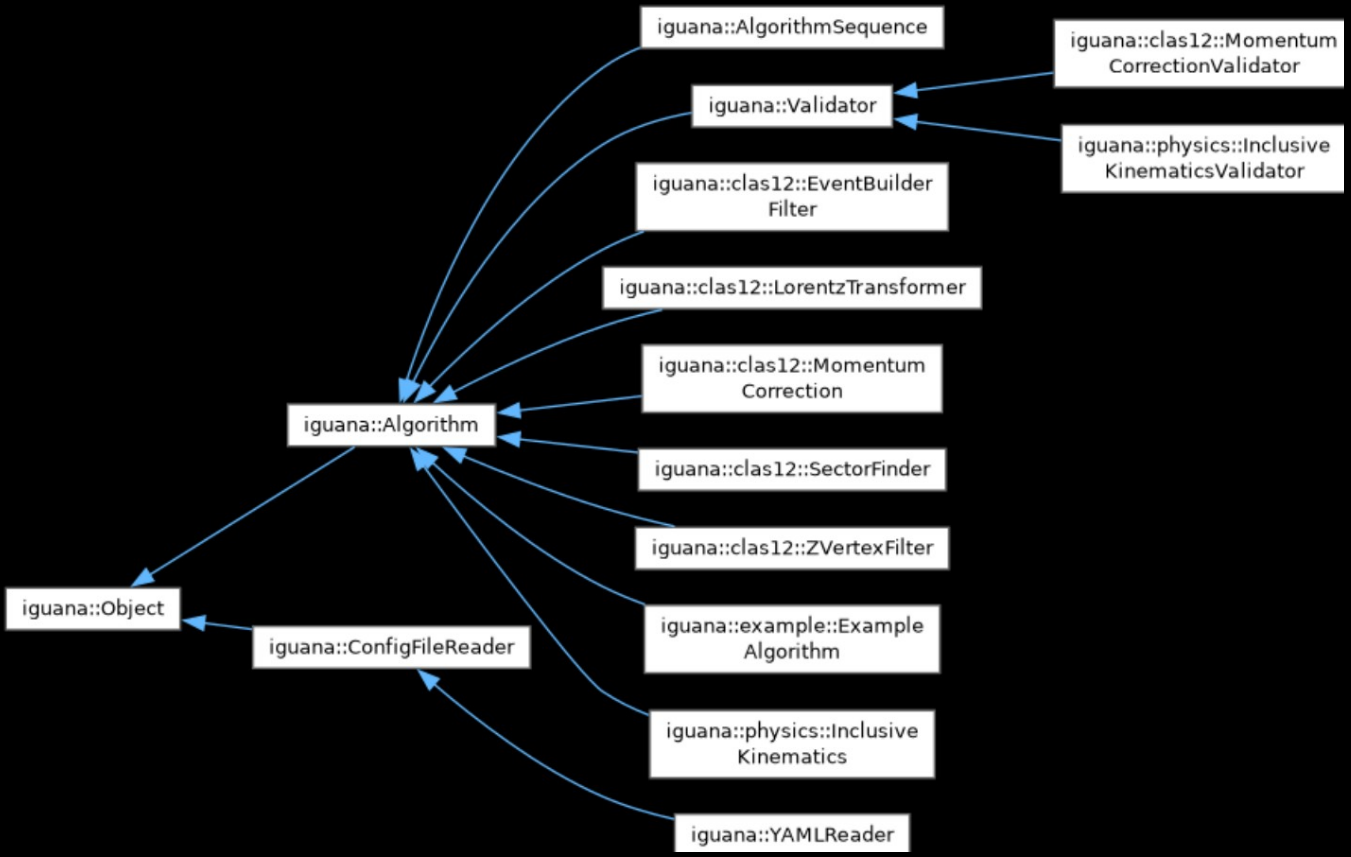


```
public:

  /// **Action function**: checks if the PDG `pid` is a part of the list of user-specified PDGs
  /// @param pid the particle PDG to check
  /// @returns `true` if `pid` is one the user wants
  bool Filter(const int pid) const;
```

Bindings

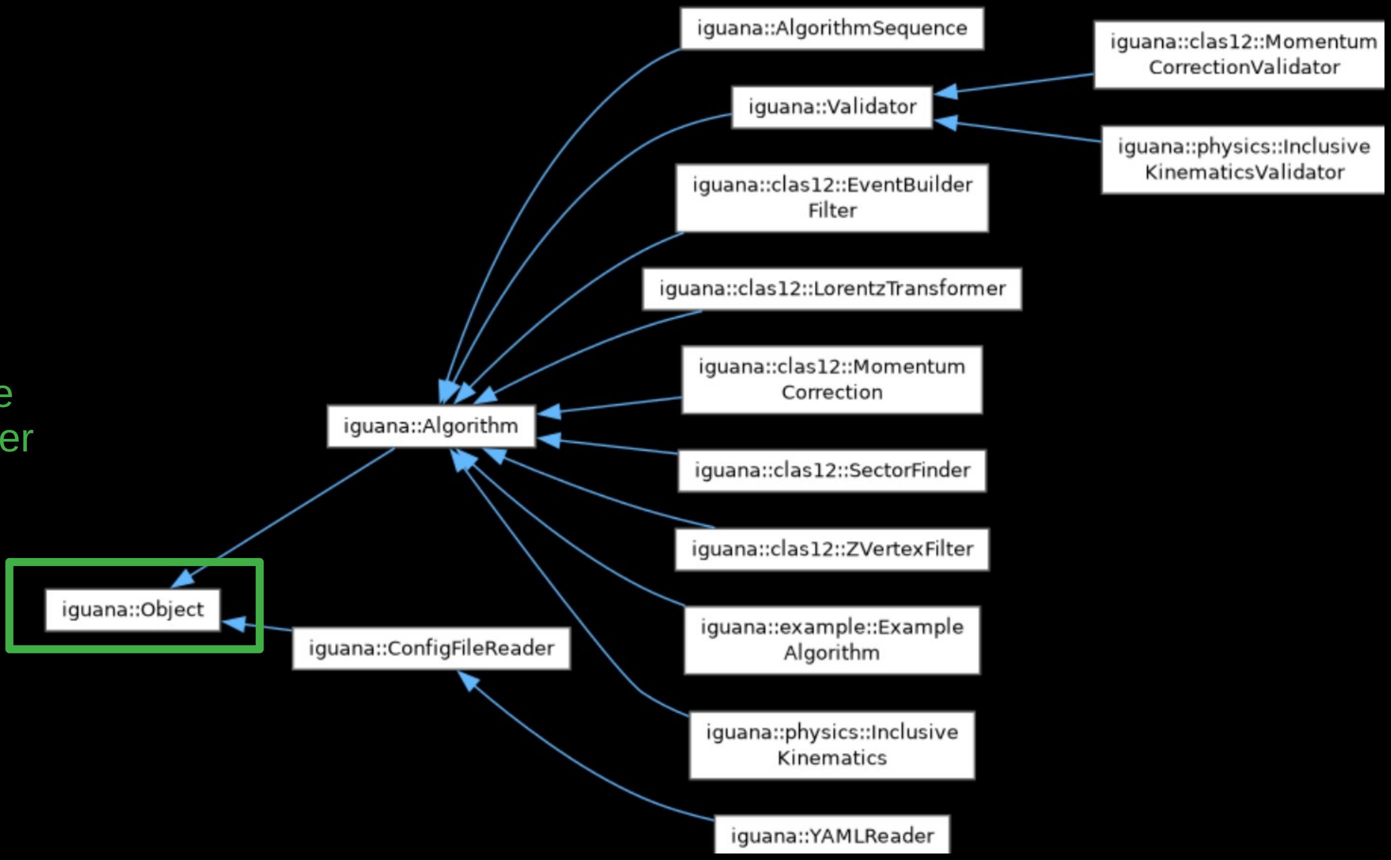
- Iguana algorithms are in C++
- We plan to support other languages via language bindings:
  - Python
  - Java
  - Fortran
- For Python, we currently use cppyy: https://cppyy.readthedocs.io/en/latest/
  - ROOT’s Python bindings, PyROOT, uses cppyy
  - Easy to implement, but our experience is that it is also easy to break
  - Planning to replace with something more robust
  - For now, this gives us some API design guidance: make it *simple*

Class Diagram


iguana::AlgorithmSequence
iguana::clas12::MomentumCorrectionValidator
iguana::Validator
iguana::physics::InclusiveKinematicsValidator
iguana::clas12::EventBuilderFilter
iguana::clas12::LorentzTransformer
iguana::clas12::MomentumCorrection
iguana::Algorithm
iguana::clas12::SectorFinder
iguana::clas12::ZVertexFilter
iguana::Object
iguana::ConfigFileReader
iguana::example::ExampleAlgorithm
iguana::physics::InclusiveKinematics
iguana::YAMLReader

Class Diagram

Something with a name and a Logger



iguana::AlgorithmSequence

iguana::Validator

iguana::clas12::MomentumCorrectionValidator

iguana::physics::InclusiveKinematicsValidator

iguana::clas12::EventBuilderFilter

iguana::clas12::LorentzTransformer

iguana::clas12::MomentumCorrection

iguana::Algorithm

iguana::clas12::SectorFinder

iguana::clas12::ZVertexFilter

iguana::example::ExampleAlgorithm

iguana::Object

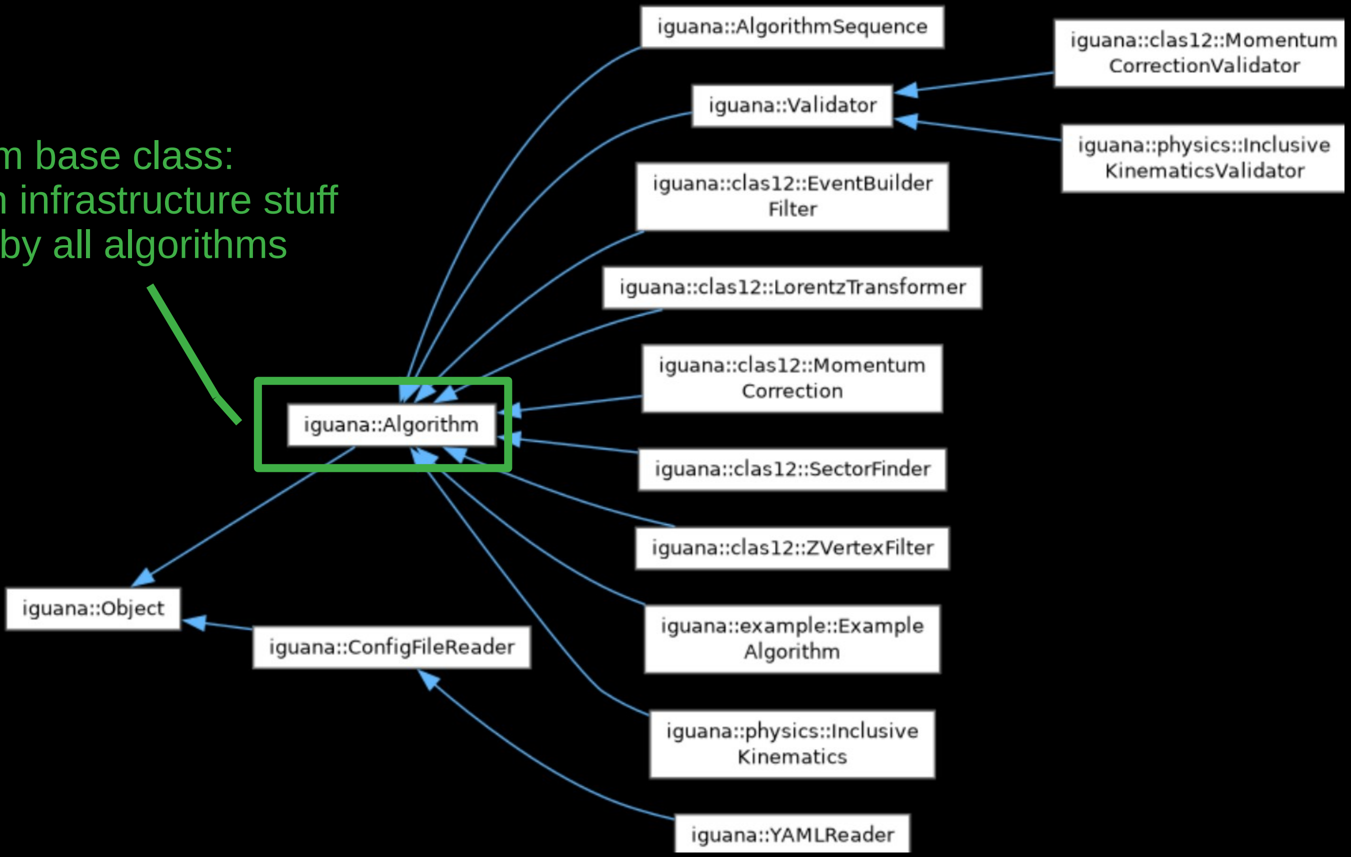
iguana::ConfigFileReader

iguana::physics::InclusiveKinematics

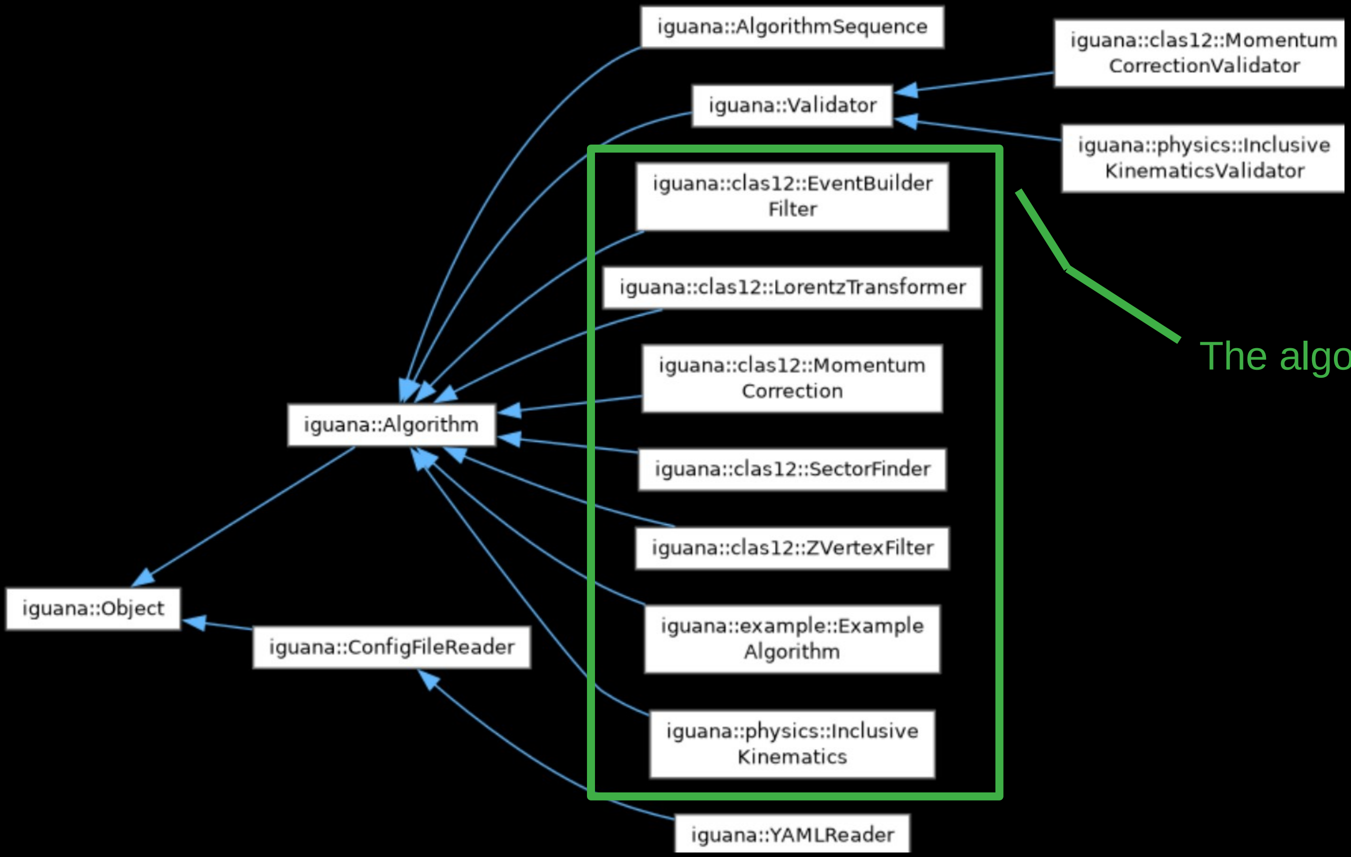
iguana::YAMLReader

Class Diagram

Algorithm base class: common infrastructure stuff needed by all algorithms

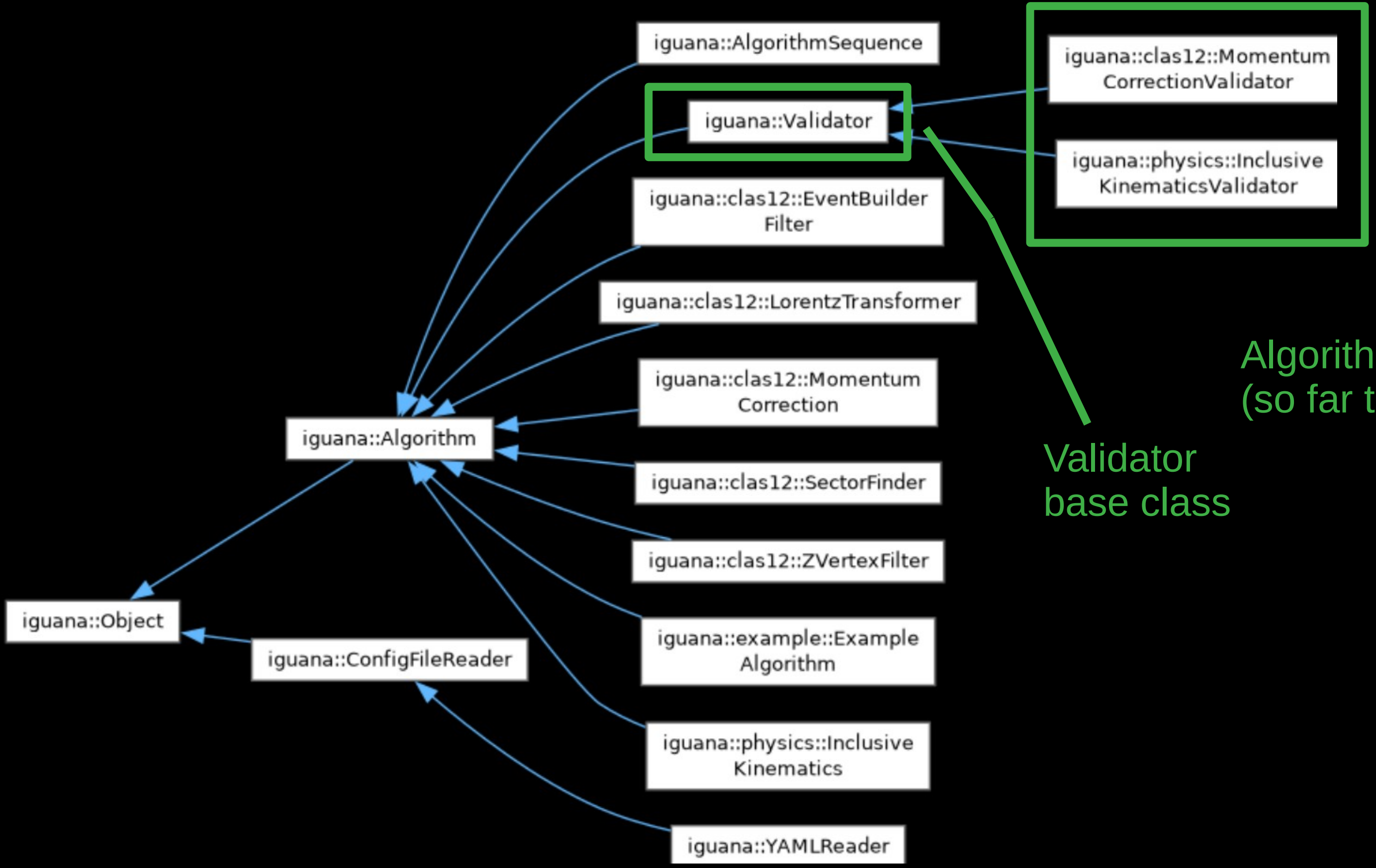

iguana::AlgorithmSequence
iguana::Validator
iguana::clas12::MomentumCorrectionValidator
iguana::physics::InclusiveKinematicsValidator
iguana::clas12::EventBuilderFilter
iguana::clas12::LorentzTransformer
iguana::clas12::MomentumCorrection
iguana::Algorithm
iguana::clas12::SectorFinder
iguana::clas12::ZVertexFilter
iguana::example::ExampleAlgorithm
iguana::physics::InclusiveKinematics
iguana::Object
iguana::ConfigFileReader
iguana::YAMLReader

Class Diagram


iguana::AlgorithmSequence
iguana::clas12::MomentumCorrectionValidator
iguana::Validator
iguana::physics::InclusiveKinematicsValidator
iguana::clas12::EventBuilderFilter
iguana::clas12::LorentzTransformer
iguana::clas12::MomentumCorrection
iguana::Algorithm
iguana::clas12::SectorFinder
iguana::clas12::ZVertexFilter
iguana::Object
iguana::ConfigFileReader
iguana::example::ExampleAlgorithm
iguana::physics::InclusiveKinematics
iguana::YAMLReader

The algorithms

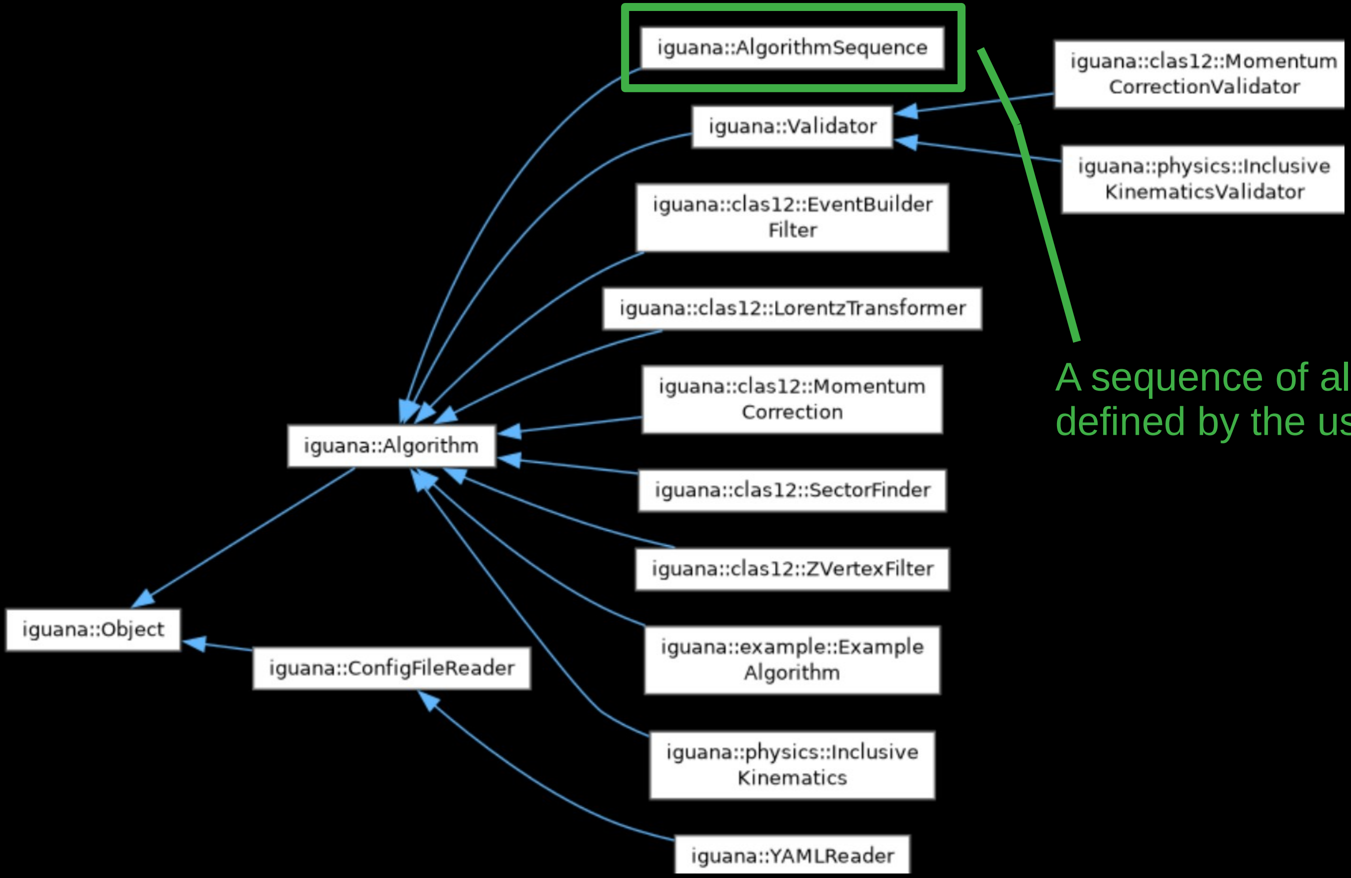
Class Diagram


iguana::AlgorithmSequence
iguana::Validator
iguana::clas12::EventBuilder Filter
iguana::clas12::LorentzTransformer
iguana::clas12::Momentum Correction
iguana::clas12::SectorFinder
iguana::clas12::ZVertexFilter
iguana::example::Example Algorithm
iguana::physics::Inclusive Kinematics
iguana::YAMLReader
iguana::Algorithm
iguana::Object
iguana::ConfigFileReader
iguana::clas12::Momentum CorrectionValidator
iguana::physics::Inclusive KinematicsValidator

Algorithm validators
(so far the only 2)

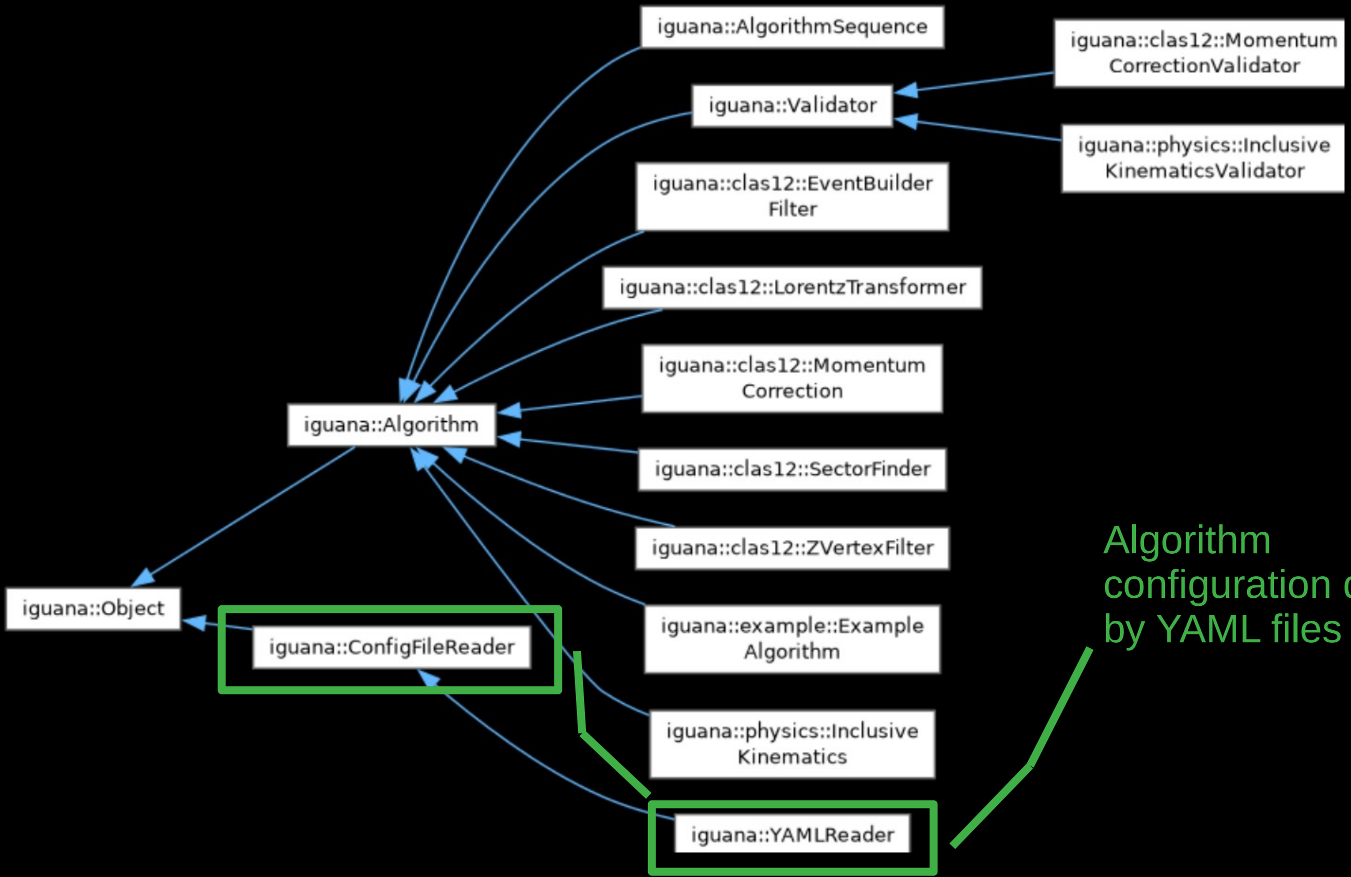
Validator
base class

Class Diagram


iguana::AlgorithmSequence
iguana::clas12::Momentum CorrectionValidator
iguana::Validator
iguana::physics::Inclusive KinematicsValidator
iguana::clas12::EventBuilder Filter
iguana::clas12::LorentzTransformer
iguana::clas12::Momentum Correction
iguana::Algorithm
iguana::clas12::SectorFinder
iguana::clas12::ZVertexFilter
iguana::Object
iguana::ConfigFileReader
iguana::example::Example Algorithm
iguana::physics::Inclusive Kinematics
iguana::YAMLReader

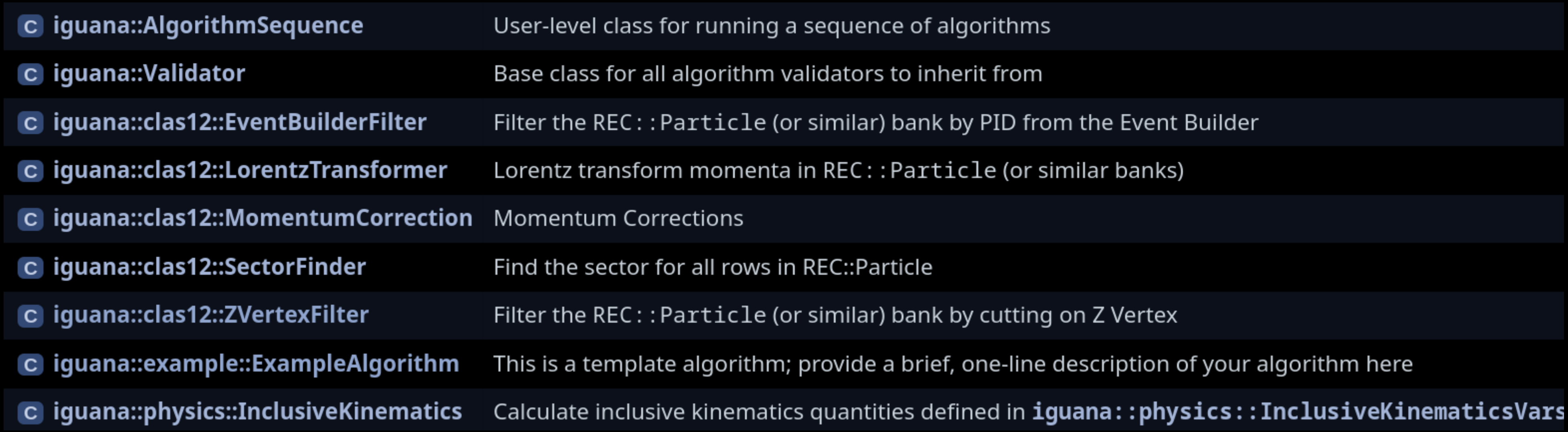
A sequence of algorithms, defined by the user

Class Diagram


iguana::AlgorithmSequence
iguana::clas12::Momentum CorrectionValidator
iguana::Validator
iguana::physics::Inclusive KinematicsValidator
iguana::clas12::EventBuilder Filter
iguana::clas12::LorentzTransformer
iguana::clas12::Momentum Correction
iguana::Algorithm
iguana::clas12::SectorFinder
iguana::clas12::ZVertexFilter
iguana::Object
iguana::ConfigFileReader
iguana::example::Example Algorithm
iguana::physics::Inclusive Kinematics
iguana::YAMLReader

Algorithm configuration defined by YAML files

Available Algorithms



C	**iguana::AlgorithmSequence**	User-level class for running a sequence of algorithms
C	**iguana::Validator**	Base class for all algorithm validators to inherit from
C	**iguana::clas12::EventBuilderFilter**	Filter the `REC::Particle` (or similar) bank by PID from the Event Builder
C	**iguana::clas12::LorentzTransformer**	Lorentz transform momenta in `REC::Particle` (or similar banks)
C	**iguana::clas12::MomentumCorrection**	Momentum Corrections
C	**iguana::clas12::SectorFinder**	Find the sector for all rows in REC::Particle
C	**iguana::clas12::ZVertexFilter**	Filter the `REC::Particle` (or similar) bank by cutting on Z Vertex
C	**iguana::example::ExampleAlgorithm**	This is a template algorithm; provide a brief, one-line description of your algorithm here
C	**iguana::physics::InclusiveKinematics**	Calculate inclusive kinematics quantities defined in **`iguana::physics::InclusiveKinematicsVars`**

Planned Algorithms

- Fiducial cuts (… any volunteers…?)
  - Just need to “copy and paste” the “right” version
  - Good opportunity to learn how to implement an algorithm, in case you are thinking of implementing your own algorithm
- FT Energy Corrections (Asli)
  - https://github.com/JeffersonLab/iguana/pull/85
- Lepton ID (Mariana & Pierre)
  - ROOT TMVA
- Semi-inclusive single-hadron kinematics (Chris)
  - Preservation of kinematics reconstruction

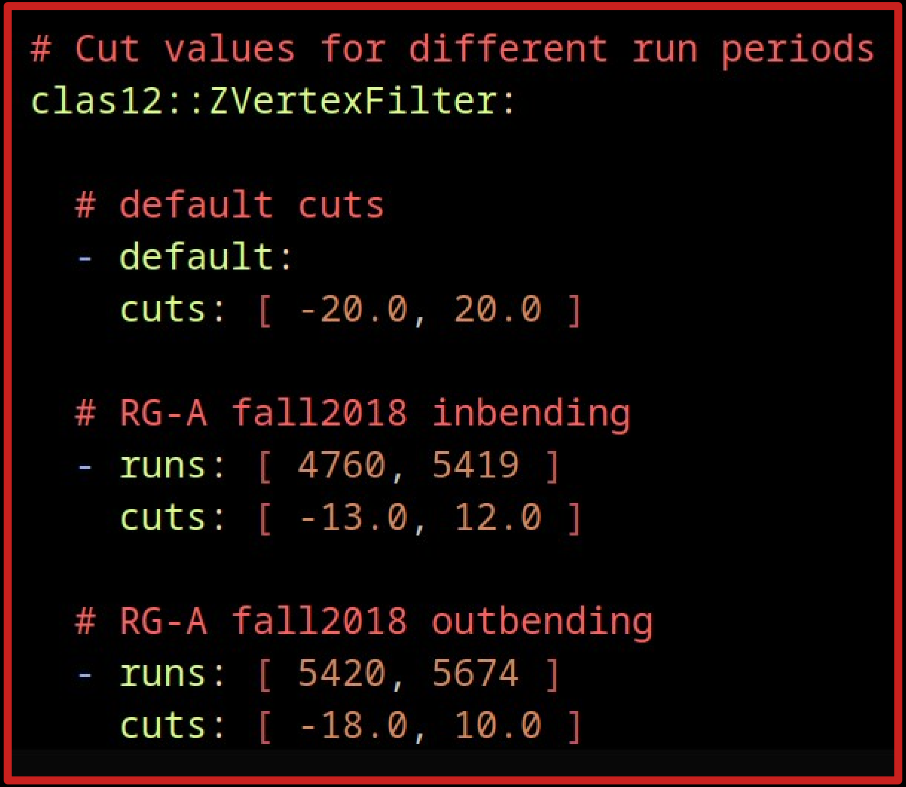
Planned Infrastructure Changes

- HIPO bank iterators
  - Currently: filtering a bank *just* sets PID to -1
  - Obviously not a good idea; this is just temporary
  - Upgrading to “HIPO Iterators” will permit filtering
    - Loop over bank rows → use an iterator instead
    - Only rows which pass the filter will be looped over
- Change Run
  - Some configurations are run dependent
  - Plan to add a general “Change Run” function to take care of this in a thread-safe way
- Other plans and issues: https://github.com/JeffersonLab/iguana/issues

Algorithm Configuration

Algorithm configuration is defined in YAML files

- Modifiable without rebuilding Iguana
- May be overridden by user’s configuration files, or on-the-fly in the code
- Standard format → programmatically modifiable
- Generalized tree structure:
  - Nested configurations
  - Dependent configurations, e.g., on run range or on PDG



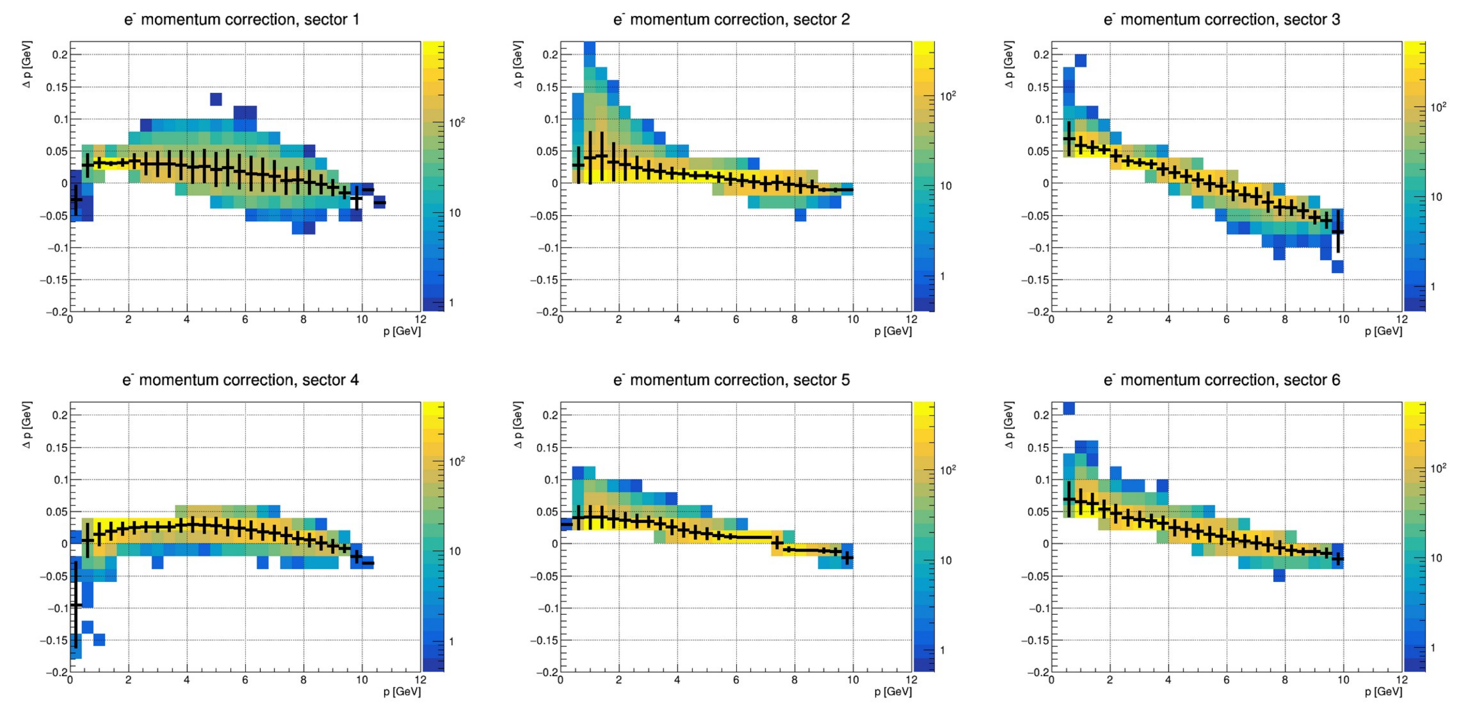
```yaml
# Cut values for different run periods
clas12::ZVertexFilter:

  # default cuts
  - default:
    cuts: [ -20.0, 20.0 ]

  # RG-A fall2018 inbending
  - runs: [ 4760, 5419 ]
    cuts: [ -13.0, 12.0 ]

  # RG-A fall2018 outbending
  - runs: [ 5420, 5674 ]
    cuts: [ -18.0, 10.0 ]
```

Automated Testing

Continuous Integration (CI) tests via GitHub Actions

- Build and test Iguana on Linux and macOS
- Tests examples and all algorithms
  - At least each algorithm ‘Run’ call is tested
  - Some algorithms have additional “Validators”
- Additional tests and automation:
  - Coverage: reports how much of the code is tested, and what lacks tests
  - Sanitizers: detects memory leaks, uninitialized reads, overflows, data races, etc.
  - Documentation generation: all C++ code is required to be documented

Validator: Momentum Corrections



The momentum correction validator just makes plots, so its up to a human to check them

Reproducibility checking may be better done with *unit tests*

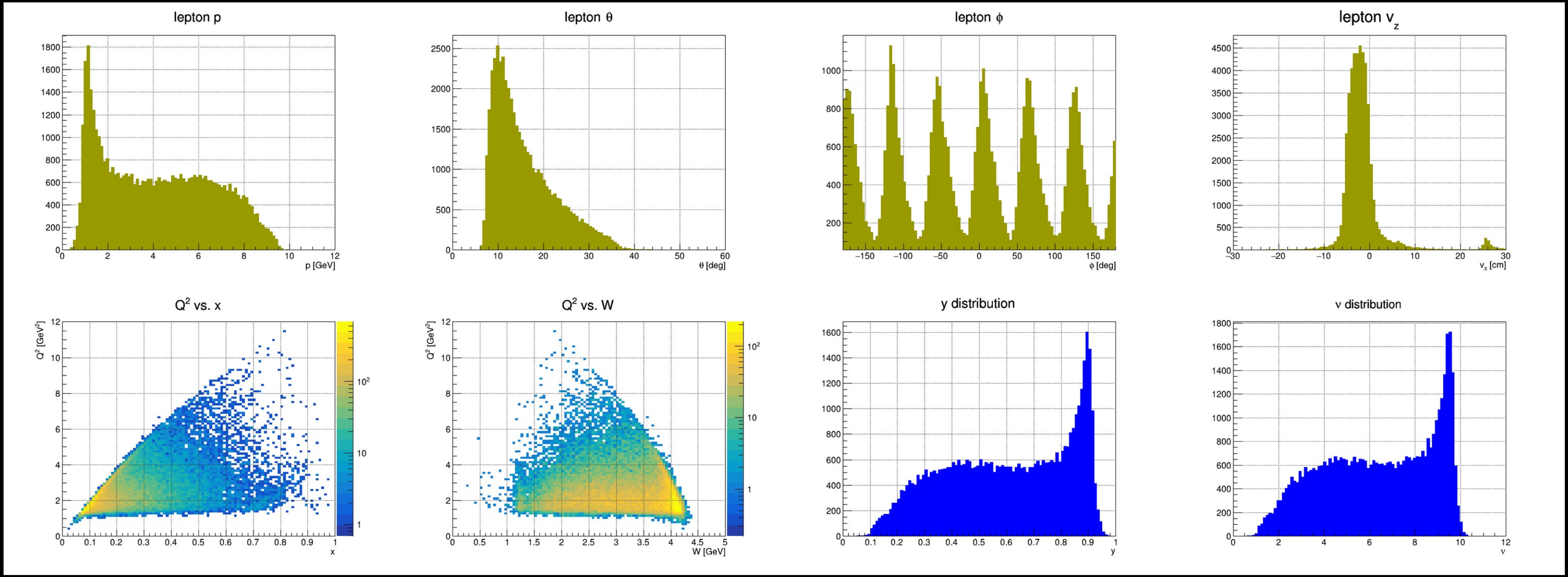
Planning on automatically deploying all such plots on a webpage

cf. plots from

https://clasweb.jlab.org/wiki/index.php/CLAS12_Momentum_Corrections#tab=Electron_Corrections__28Plots_29_-_Inbending

Validator: Inclusive Kinematic Calculations

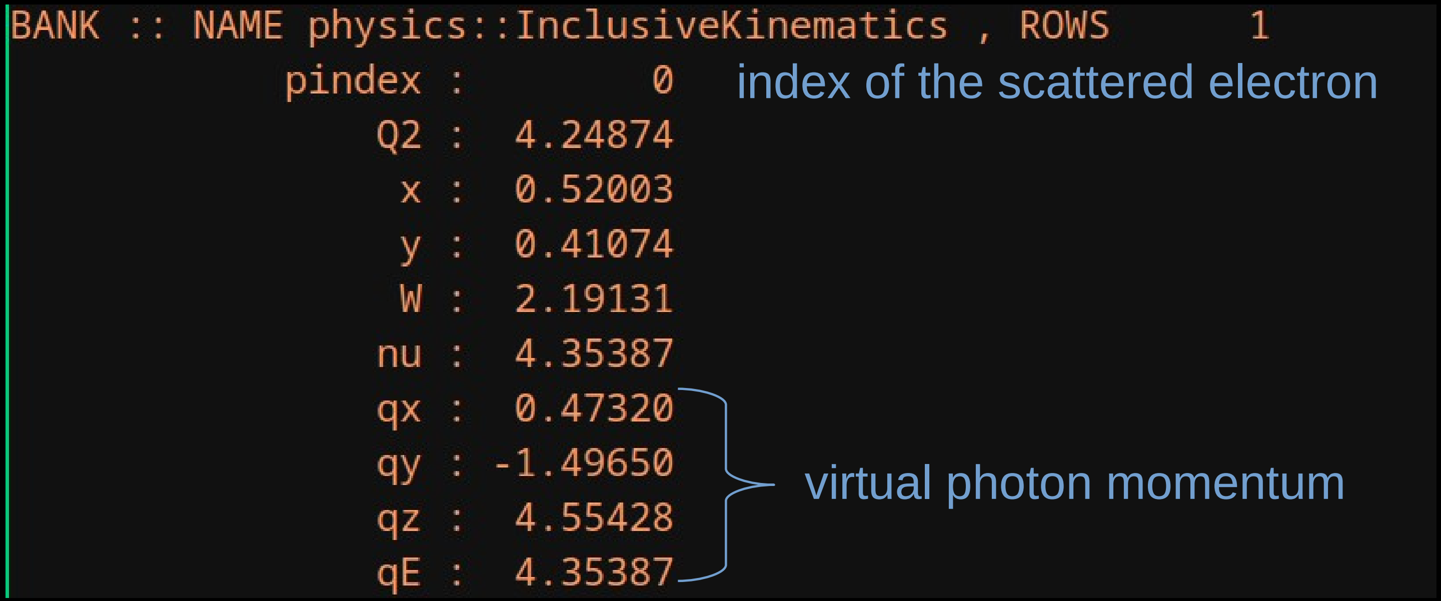
Inclusive kinematics distributions; no cuts applied (yet)



By the way… this is a “Creator” Algorithm

iguana::physics::InclusiveKinematics **creates** and fills a **new** bank (schema)

Example for 1 event:



```
BANK :: NAME physics::InclusiveKinematics , ROWS      1
            pindex :         0
                Q2 :   4.24874
                 x :   0.52003
                 y :   0.41074
                 W :   2.19131
                nu :   4.35387
                qx :   0.47320
                qy :  -1.49650
                qz :   4.55428
                qE :   4.35387
```

index of the scattered electron

virtual photon momentum

Validation: **You!**

- We’re a collaboration, and Iguana is Open Source
- Iguana is not a black box
- You are encouraged to be skeptical of algorithms and to check them
- Please report and/or fix any issues

Want to try Iguana?

https://github.com/JeffersonLab/iguana

- Available from the new Alma9 interactive node
  - Procedure:

    ```
    ssh ifarm9.jlab.org
    source /group/clas12/packages/setup.csh
    module load iguana/0.4.0
    ```

  - Warning: some things are missing from this build…
- Source code available on GitHub
  - https://github.com/JeffersonLab/iguana
  - Build it yourself

If you build it yourself...

Dependencies

Guidance:
https://github.com/JeffersonLab/iguana/blob/main/doc/setup.md#dependencies

- **meson**: build system generator
- **hipo**: C++ HIPO API
  - You may need the latest version on the `master` branch, since the latest release lacks some features now required by Iguana
- **fmt**: for printout messages
- **yaml-cpp**: for YAML configuration files
- **ROOT (optional)**
  - If you don’t have it, algorithms which need it won’t be built

Contributions are Welcome

image source

- 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