AGC workshop systematics + correctionlib

Alexander Held (University of Wisconsin–Madison)

Andrew Wightman (University Nebraska–Lincoln)

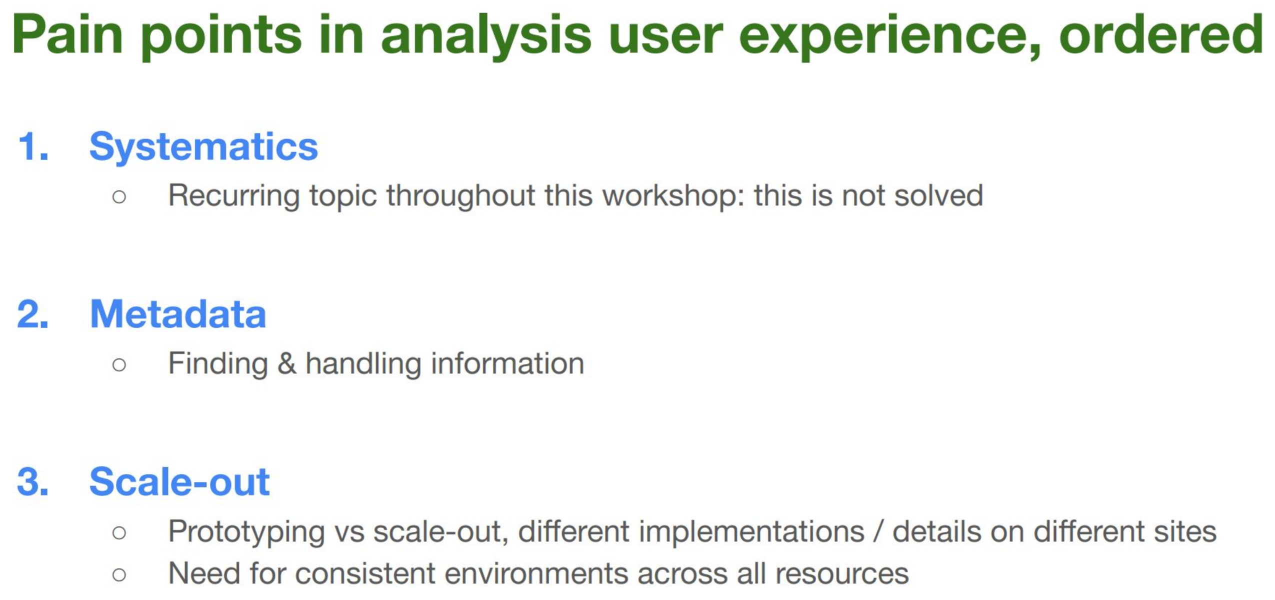
May 3, 2023

IRIS-HEP AGC workshop 2023
https://indico.cern.ch/e/agc-workshop-2023

This work was supported by the U.S. National Science Foundation (NSF) Cooperative Agreement OAC-1836650 (IRIS-HEP).

Context

Analysis Ecosystem Workshop II



Pain points in analysis user experience, ordered

1. **Systematics**
   - Recurring topic throughout this workshop: this is not solved
2. **Metadata**
   - Finding & handling information
3. **Scale-out**
   - Prototyping vs scale-out, different implementations / details on different sites
   - Need for consistent environments across all resources

Types of systematic uncertainties

- The default "*nominal*" scenario takes all events (+ properties of objects within each event) from some "nominal" simulation
- Compared to the nominal case, a systematic variation can
  - **Change the weight** of each event,
  - **Change properties of objects** within events,
  - **Replace all events** completely.

Effects of the types of uncertainties

- **Changing weights**
    - Apply *prescription* to get new per-event weight
    - Typically computationally cheap
- **Changing event properties**
    - Apply *prescription* to get new object kinematics for each event
    - Typically more expensive (may affect subsequent calculations!)
- **Replace all events**
    - This essentially behaves like the “nominal” case
    - Bookkeeping exercise mostly

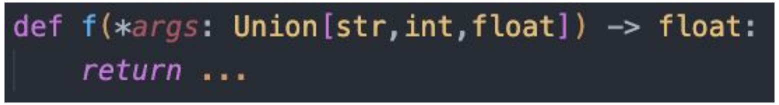
Challenges in practice

- There is a lot of **bookkeeping** involved
    - Different systematic uncertainties act on different samples via different methods

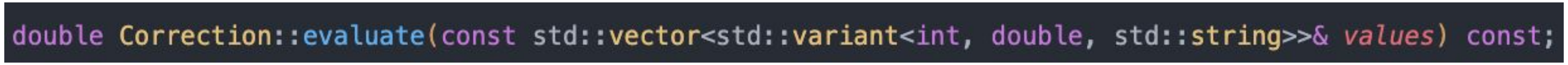
- Users need to be able to **access and implement** *prescription*
    - ATLAS: centrally provided tools, typically run on the grid
    - CMS: on-the-fly evaluation possible with NanoAOD

Correctionlib

- `correctionlib` provides JSON data format + tool for applying *prescription* / corrections
  - C++ and Python interface



```python
def f(*args: Union[str,int,float]) -> float:
    return ...
```



```cpp
double Correction::evaluate(const std::vector<std::variant<int, double, std::string>>& values) const;
```

- See Nick Smith's PyHEP 2022 talk for more information
- AGC just switched to starting to employ `correctionlib`
  - Opportunity for streamlining implementations
  - Lower barrier to entry: can centrally provide JSONs to use
  - Opportunity to explore & improve user experience in implementations

The ideal (?) user experience

correctionlib goes into here
not clear that this level of abstraction is possible (or a good idea!)

```
def process(self, events):

    for variation in all_the_variations:

        # get event + weight for variation
        varied_event, weight = magic_interface(events, variation)

        # perform event selection (>= 4 jets)
        filter = ak.count(varied_event.jet.pt, axis=1) >= 4

        # extract observable (jet pT sum)
        observable = ak.sum(varied_event[filter].jet.pt, axis=-1)

        # fill histogram
        histogram.fill(observable, weight=weight)

    return histogram
```

The AGC example in practice

- **Andrew** will **showcase** how things look like for the **AGC setup**

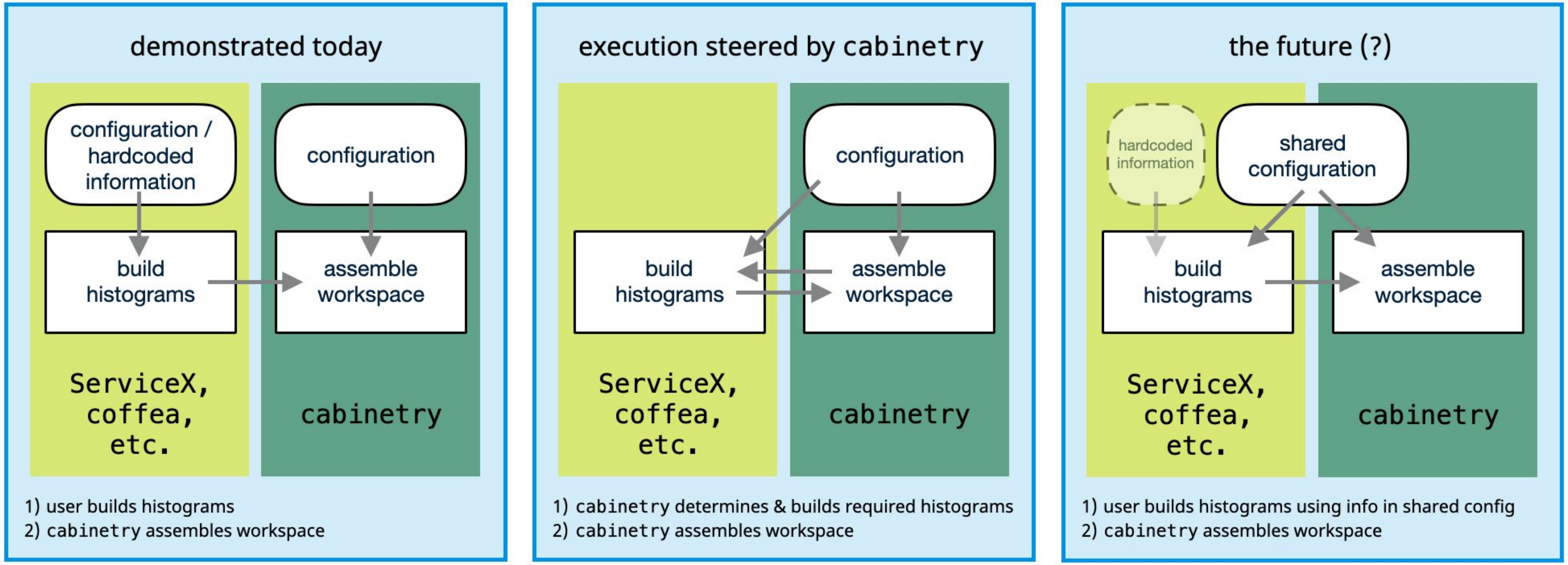
- Planning to **increase use of** `correctionlib` for AGC
  - More systematic uncertainties coming for AGC v2
  - analysis-grand-challenge#101

- **Feedback** regarding user experience / interfaces would be great!
  - Hoping to provide a nice example for how to set things up via AGC

Backup

AGC ATLAS Open Data H>ZZ* notebook

- See notebook for more context



1) user builds histograms
2) cabinetry assembles workspace

1) cabinetry determines & builds required histograms
2) cabinetry assembles workspace

1) user builds histograms using info in shared config
2) cabinetry assembles workspace