cabinetry

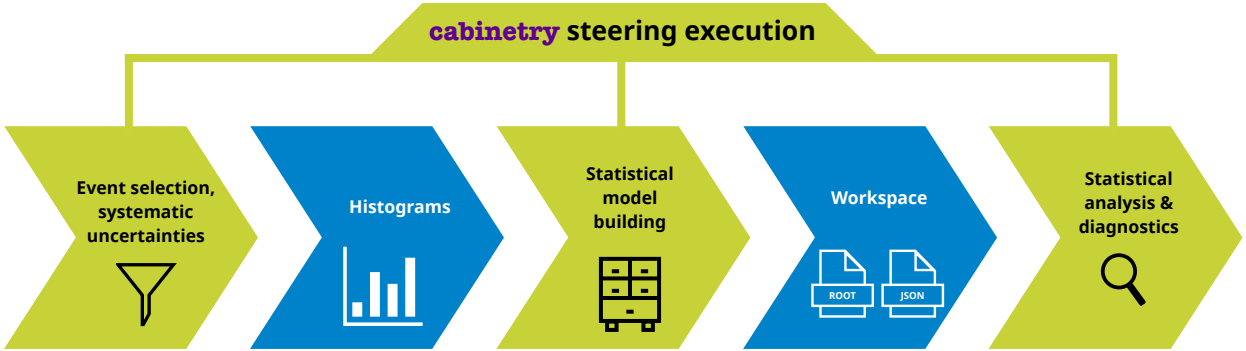
Kyle Cranmer[1], Alexander Held[1]

[1] New York University

IRIS-HEP retreat https://indico.cern.ch/event/896167
May 27, 2020

What is cabinetry?

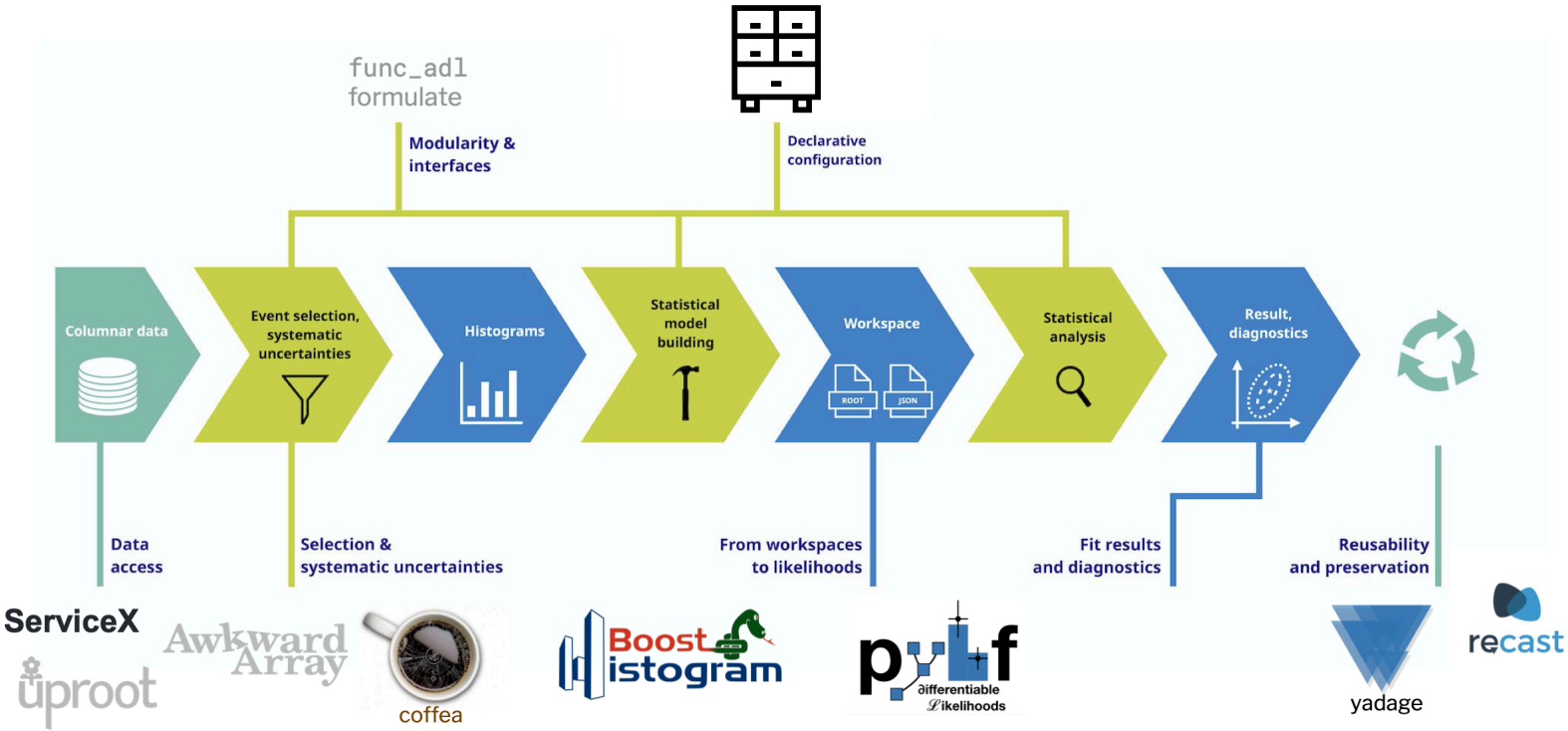
- analyzers use cabinetry to *design template fit models* via a *declarative configuration* file
  - analyzers specify selections for signal/control regions, (Monte Carlo) samples, systematic uncertainties
  - cabinetry steers the histogram creation (region ⊗ sample ⊗ systematic)
  - using the histograms, cabinetry produces a workspace (serialized fit model)
  - cabinetry steers statistical inference and provides diagnostics and visualization tools



- github: https://github.com/alexander-held/cabinetry/
- IRIS-HEP project page: https://iris-hep.org/projects/cabinetry.html

cabinetry within IRIS-HEP

- a declarative configuration steers cabinetry and its interactions with other tools
- data is delivered to cabinetry via e.g. ServiceX after coffea processing, or straight from uproot as an awkward array
- pyhf is used for inference



Hello world (1)

- simple fit model defined via configuration file in YAML format
  - required histograms are implicitly specified as region ⊗ sample ⊗ systematic
- cabinetry creates histograms, visualize data, creates a workspace and runs a fit
- try out the example in the repository

```python
import cabinetry

cabinetry_config = cabinetry.config.read("config_example.yml")

# create template histograms
histo_folder = "histograms/"
cabinetry.template_builder.create_histograms(
    cabinetry_config, histo_folder, only_nominal=True, method="uproot"
)

# perform histogram post-processing
cabinetry.template_postprocessor.run(cabinetry_config, histo_folder)

# visualize templates and data
cabinetry.visualize.data_MC(cabinetry_config, histo_folder, "figures/", prefit=True,
method="matplotlib")

# build a workspace
ws = cabinetry.workspace.build(cabinetry_config, histo_folder)

# run a fit
cabinetry.fit.fit(ws)
```

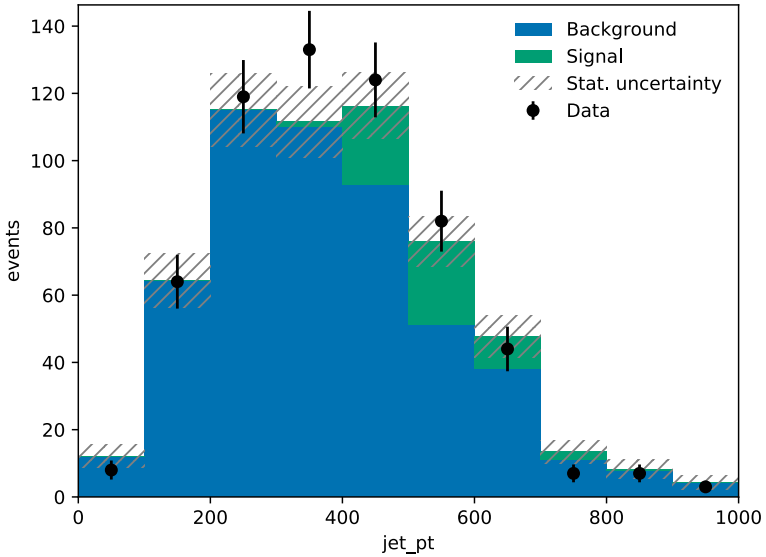
```yaml
General:
  Measurement: "My fit"
  POI: "Signal strength"

Samples:
  - Name: "Data"
    Tree: "pseudodata"
    Path: "ntuples/data.root"
    Data: True

  - Name: "Background"
    Tree: "background"
    Path: "ntuples/prediction.root"
    Weight: "weight"

  - Name: "Signal"
    Tree: "signal"
    Path: "ntuples/prediction.root"
    Weight: "weight"

Regions:
  - Name: "Signal Region"
    Variable: "jet_pt"
    Filter: "lep_charge > 0"
    Binning: [0, 100, 200, 300, 400, 500]

Systematics:
  - Name: "Luminosity"
    OverallDown: -0.05
    OverallUp: 0.05
    Samples: ["Signal", "Background"]
    Type: OVERALL

NormFactors:
  - Name: "Signal strength"
    Nominal: 1
    Min: 0
    Max: 5
    Samples: "Signal"
```

Hello world (2)

(partial) output

```
INFO - cabinetry.fit - staterror_Signal-Region[bin_0]: 0.811171 +/- 0.200473
INFO - cabinetry.fit - staterror_Signal-Region[bin_1]: 0.989652 +/- 0.090191
INFO - cabinetry.fit - staterror_Signal-Region[bin_2]: 1.010105 +/- 0.068933
INFO - cabinetry.fit - staterror_Signal-Region[bin_3]: 1.083871 +/- 0.069312
INFO - cabinetry.fit - staterror_Signal-Region[bin_4]: 1.016181 +/- 0.069627
INFO - cabinetry.fit - staterror_Signal-Region[bin_5]: 1.015840 +/- 0.087416
INFO - cabinetry.fit - staterror_Signal-Region[bin_6]: 0.950477 +/- 0.102159
INFO - cabinetry.fit - staterror_Signal-Region[bin_7]: 0.732744 +/- 0.188015
INFO - cabinetry.fit - staterror_Signal-Region[bin_8]: 0.914896 +/- 0.242512
INFO - cabinetry.fit - staterror_Signal-Region[bin_9]: 0.824391 +/- 0.345761
INFO - cabinetry.fit - Luminosity                    : 0.278680 +/- 0.843117
INFO - cabinetry.fit - Signal strength               : 1.078683 +/- 0.381934
```

visualization example (pre-fit)



Status and goals

- **cabinetry** started a month ago
  - feature set is being actively expanded

- **goals:**
  - become a convenient tool for LHC-style binned template fit definition and steering
  - leverage other existing IRIS-HEP tools for faster time-to-insight
    - factorize as much as possible, and support multiple backends where sensible
  - develop and promote API to help factorize tasks into independent modules
  - declarative approach
    - but allow analyzers to supply their own functions for key tasks that interact with the declarative configuration

- **other existing tools:**
  - range of frameworks with similar scope exits: `CMS combine`, many within ATLAS (`HistFitter`, `TRExFitter`, `WSMaker`, ...)
  - want to provide a python / non-ROOT alternative, based on experience from existing tools

Getting data to cabinetry

- **cabinetry** makes requests to build histograms, which should be fulfilled by external backends
  - need a common API
  - simple uproot backend currently used as reference

- **Challenges:**
  - defining the data location in a file type agnostic way
    - how to deal with substructure in the file, what if the file is not a file but a location in memory?
  - language for defining observables, cuts, weights
    - using `data["MET"] > 100` or `MET > 100` or ...
    - implicit assumptions of how strings map to objects within the file
    - may not need to support complex operations - this can be done in upstream tools

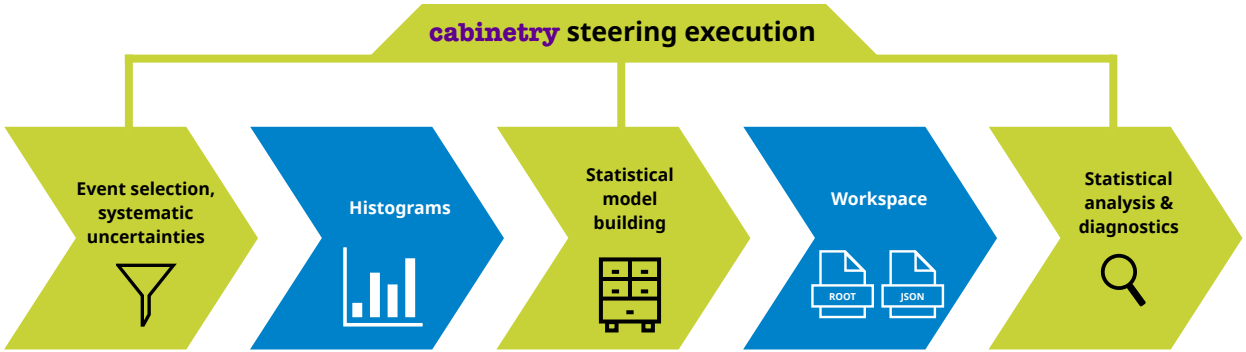
Years 3, 4, 5

- **year 3:**
  - obtain community feedback, promote community involvement in development
  - converge on core design decisions
  - expand feature set to cover most common use cases
  - gather experience with fully featured LHC-style analysis

- **years 4+5:**
  - move towards a stable product
  - Integrate in IRIS-HEP analysis systems grand challenge

Summary

- **cabinetry is**
  - a new effort aiming to interface many existing IRIS-HEP tools
  - a modular, python-based approach to building workspaces for statistical inference with template fits
  - both a library (e.g. workspace creation from histograms) and a framework (steering other tools)
  - welcoming contributions and thoughts!



Backup

Fully declarative approach?

- design of a declarative configuration that points to the right path for the data needed for any histogram?
  - analyzers might store their data in many different structures

```
nominal/
  signal.root
    region_1
    region_2
  [...]
systematic_variation/
  signal.root
    region_1
     [...]
```

```
region_1/
  signal_nominal.root
  signal_systematic_variation.root
  background.root
region_2/
  signal_nominal.root
  signal_systematic_variation.root
  background.root
```

  - difficult to support any possible structure via pre-defined options
  - more powerful approach: analyzers define their own option and provide a function to parse them

- see https://github.com/alexander-held/cabinetry/issues/16 for more