

# UW Work In AS

M. Proffitt, E. Torro, G. Watts  
(UW/Seattle)

# People

- Gordon
  - Worked extensively in Data Preservation and Analysis Languages
  - Member of the ATLAS and MATHUSLA experiments, as well as DZERO.
  - (and AMY, but that is loooooooong ago!)
- Emma Torro
  - Post-doc at UW, ATLAS and MATHUSLA
  - Analysis and Trigger expert, has helped design several analyses, and has helped manage old-school C++ analysis framework
  - Officially joins IRIS-HEP effort in a month
- Mason Proffitt
  - Graduate Student at UW, ATLAS and MATHUSLA
  - Worked on b-tagging in ATLAS (uses python analysis ecosystem)
  - Also expert at analysis in C++
  - Joined IRIS-HEP effort on January 1.

# Some Documents

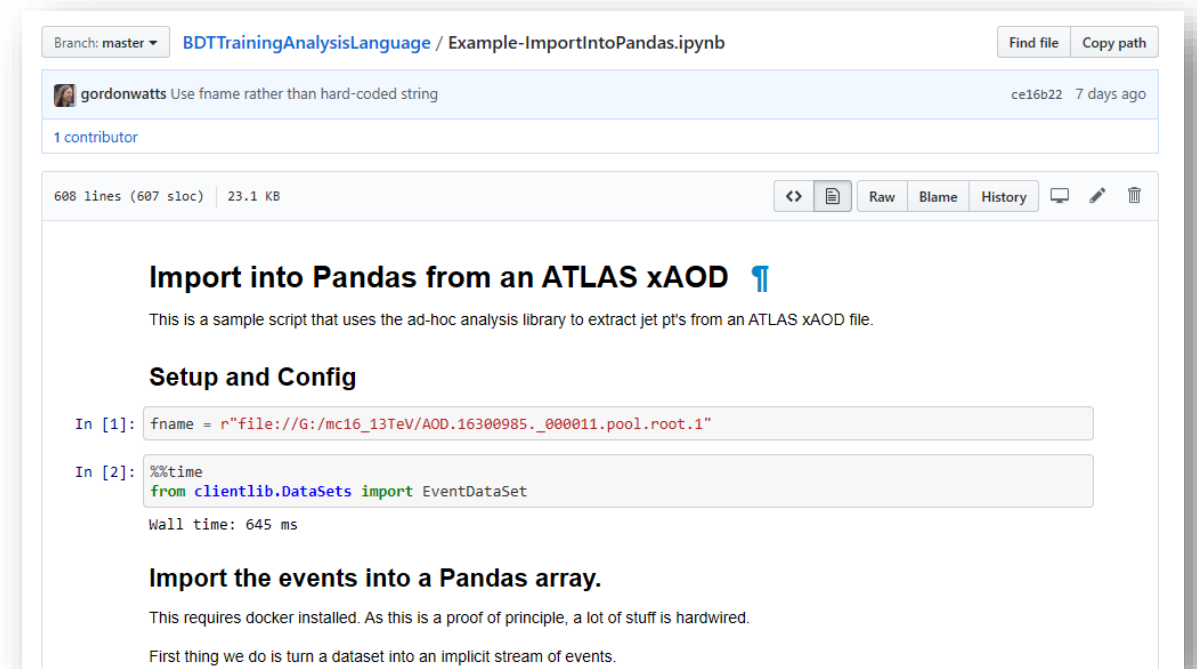
- An attempt to develop an ‘intellectual framework’ to think about analysis languages and systems.
  - Document: [Analysis Language Use Cases](#)
    - What processing steps are required in a LHC/HEP experiment?
    - Example queries that a query system needs to support
      - These are use cases that we would want our analysis language to be able to handle cleanly.
    - We need to add more of these! Help welcome!
  - Slides we’ve started
    - [Analysis Systems Global View](#) – slightly updated slide from the kick off meeting that tries to put AS in the grand context
    - [Analysis Language Hierarchy](#) – attempt to classify the various types of analysis languages out there (along just one axis so far).
    - [Analysis System Context](#) – Crude attempt to place roles for analysis languages, Physicist Interface, and the analysis system.
- All documents are rough, and all welcome contributions
- To help us (UW??) organize our thinking when relating to the already existing large body of work.

# A LINQ Based Analysis Language

- Prior work was based on C#'s linq library
  - A set of operators of sequences of events, jets, tracks, etc.
  - Plotting, producing output TTree's
  - Backend ran only on flat ROOT TTree's
  - Used in several published analyses.
- New: Prototype in Python to answer several questions:
  - Can we do this in python in a reasonable way?
  - Can the same frontend language handle flat ROOT TTree's, columnar awkward arrays, and xAOD's as input?
  - Can an Abstract Syntax Tree (AST) contain the complete information for a query
    - E.g. is it expressive enough to be the over-the-wire language?
  - Emma and Mason also working on this effort.
  - Hope to have initial framework to answer some of these questions by HOW.
  - Prior experience tells says almost certainly yes for all but the columnar analysis... that will take some showing to prove.

# Current State

- [Repro on github in gwatts' private area](#)
  - Goal: be able to write out the training I need for a full Run 2 analysis I'm working on with Emma and Mason.
- Current status:
  - Runs on ATLAS xAOD's
  - Uses docker to extract data
  - Writes it to a ROOT file, and then returns that as a pandas Dataframe.
  - Can run in a Jupyter notebook (!!)
    - Note: running xAOD loop in a notebook like this is just... weird in a very good way.
  - But... this is basically all it can do.



The screenshot shows a Jupyter Notebook interface for a file named 'Example-ImportIntoPandas.ipynb' in the 'BDTTrainingAnalysisLanguage' repository. The notebook content includes a title 'Import into Pandas from an ATLAS xAOD', a description of the script's purpose, and two input cells. The first cell sets a filename, and the second cell imports the 'EventDataSet' class from 'clientlib.DataSets'. The notebook also shows execution time and instructions for running the script.

```
Branch: master | BDTTrainingAnalysisLanguage / Example-ImportIntoPandas.ipynb | Find file | Copy path
```

gordonwatts Use fname rather than hard-coded string | ce16b22 | 7 days ago

1 contributor

608 lines (607 sloc) | 23.1 KB | Raw | Blame | History

### Import into Pandas from an ATLAS xAOD

This is a sample script that uses the ad-hoc analysis library to extract jet pt's from an ATLAS xAOD file.

#### Setup and Config

```
In [1]: fname = r"file:///G:/mc16_13TeV/AOD.16300985._000011.pool.root.1"
```

```
In [2]: %%time
from clientlib.DataSets import EventDataSet

Wall time: 645 ms
```

#### Import the events into a Pandas array.

This requires docker installed. As this is a proof of principle, a lot of stuff is hardwired.

First thing we do is turn a dataset into an implicit stream of events.

- Next:
  - Flesh out more primitives (like filtering, nested loops, etc.)
  - Back end to run on a DataFrame and flat TTree's
  - Then look at a more pythonic way of expressing the queries
    - Currently is playing to the strengths of C#

# Apologies

- G. Watts and M. Proffitt are attending a FAIR workshop
  - Findable, Accessible, Interoperable, and Reusable
  - In short: data preservation
  - If people are interested, happy to report at a future discussion.