Analysis Metadata - ATLAS

HSF DAWG

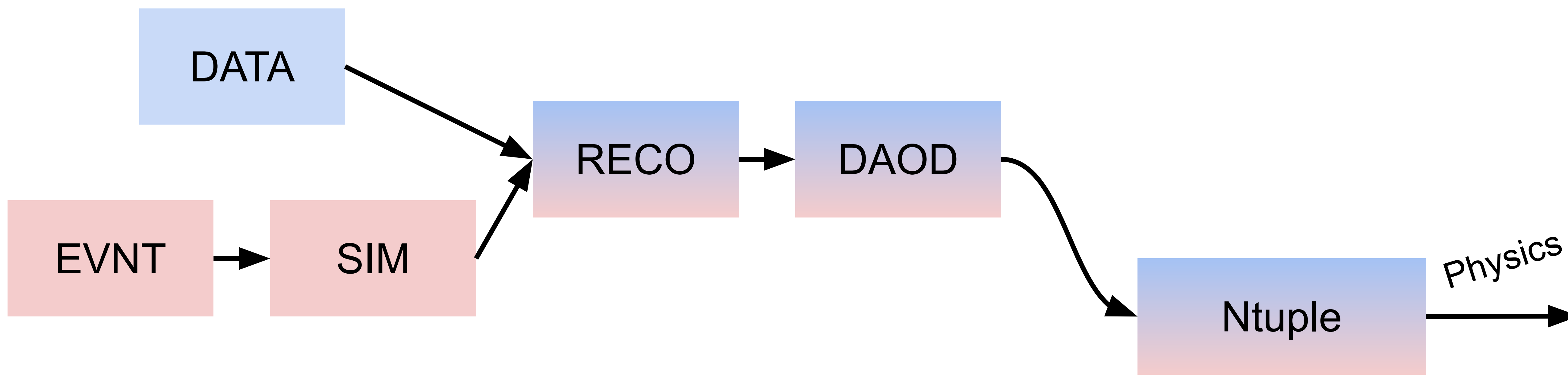
S. Alderweireldt (Edinburgh), J. Anders (Bern)

February 16th, 2020

Introduction

- Brief overview of Metadata usage from an analyser's perspective
- Originally designing these slides I didn't think there was *too much* usage of metadata when dealing with the end-user analysis
    - I was quite wrong…
- To note before starting:
    - The use of metadata from the analysis-side mostly works well!
    - Accessing the required metadata isn't too difficult/complex (once an analyser knows where everything is)
    - Using the metadata to set up an analysis is *mostly* trivial
    - But there are some cases though where the incorrect propagation/accessing of metadata can cause significant issues.

ATLAS dataflow

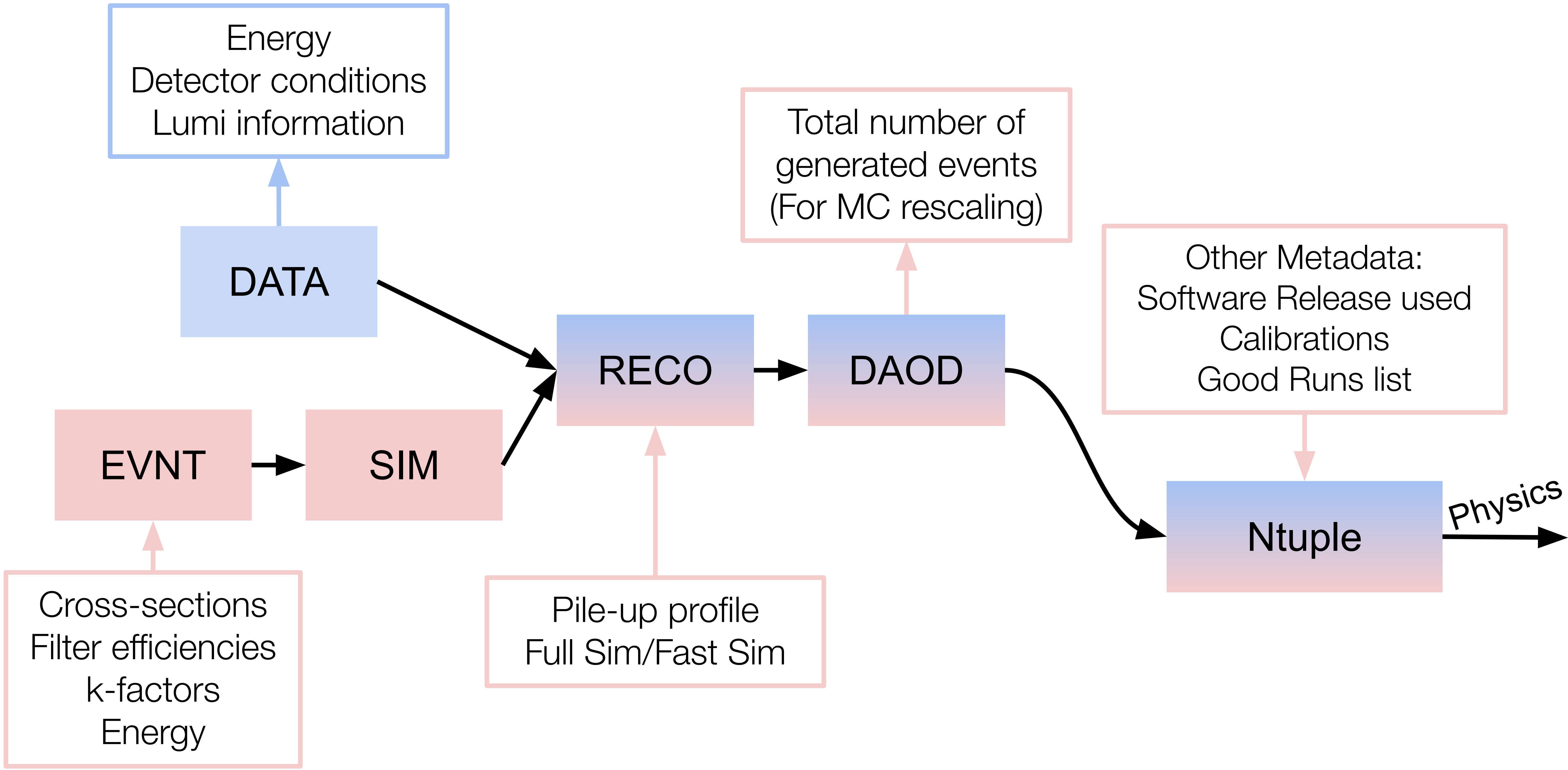


Analysis software
submitted to the grid (via rucio)
and monitored (with panda)
Tadej's talk from last time

• EVNT:	HEP MC event generated in free space
• SIM:	Addition of detector simulation to the EVNT
• DATA:	… Data
• RECO:	Reconstruction of potential physics objects (e, μ, τ, γ…) applied to both SIM & DATA
• DAOD:	Reduced file containing only events/objects individual analyses are interested in
• NTuple:	General analysis workflow only considers running the user's analysis code on the DAOD to produce a ntuple

ATLAS dataflow - Analysis Metadata

- General analysis workflow only considers running the user's analysis code on the DAOD to produce a ntuple (which is then used for plotting/statistics/etc)
  - But there is a large amount of metadata that is needed at the "Physics" stage, from the previous steps



Accessing Analysis Metadata

- There are multiple ways that the metadata is currently accessed, some easier to deal with from an analysis perspective than others:
  - Via the input file/container name
    (Energy, pile-up profile, Full/Fast sim, release used)
    - Simple string manipulation of the input container name to provide the energy and the "tag" to identify which PU profile to use/which software release was used (which both require searching the correct database)
  - Via a look-up table/database
    (detector conditions, cross-sections, filter efficiencies, k-factors)
    - Detector conditions for a given run are collected using the Run Number (contained within the input file) as input to an online database
    - Cross-sections/etc are done in a similar manner (grabbing the data-set ID and cross-referencing with an online database (however this database is available within an analysis framework as a .txt file)
  - Within the file itself
    (Lumi info, total number of generated events)
    - Usually pretty easy to use
  - Via a twiki (GRL, calibrations)
    - General information (which GRL to provide to which year/conditions, which calibration for a given release) tends to be stored in a selection of twikis

Accessing Analysis Metadata

- There are multiple ways that the metadata is currently accessed, some easier to deal with from an analysis perspective than others:
  - Via the input file/container name
    (Energy, pile-up profile, Full/Fast sim, release used)
    - Simple string manipulation of the input container name to provide the energy and the "tag" to identify which PU profile to use/which software release was used (which both require searching the correct database)
  - Via a look-up table/database
    (detector conditions, cross-sections, filter efficiencies, k-facto
    - Detector conditions for a given ru
      (contained wit
  -
    provide to which year/conditions, which
    a given release) tends to be stored in a selection of twikis

These different access methods aren't necessarily better/worse than each other when considered in a more general (experiment-wide) manner but specifically when considering the analyses' perspective some are much easier to deal with than others

There is a considerable amount of working with metadata, in analysis, that does require simply "knowing" where different information is kept/documented, which may be non-obvious to all users

Potential difficulties

- As introduced earlier:
  - Accessing the required metadata isn't too difficult/complex (once an analyser knows where everything is)
    - Though this does require you to know where everything is…
  - Using the metadata in an analysis is *mostly* trivial
    - Though if you don't realise you need metadata for a certain calculation/tool/function then you may need to rerun…
  - There are some cases though where the incorrect propagation/accessing of metadata can cause significant issues.
    - Using incorrect values (due to metadata being accessed/used incorrectly) will lead to incorrect physics

- A few examples:

  1 - Filenames
  2 - Look-up tables
  3 - Stored file information
  4 - Twikis

1 - Metadata from filenames

- A lot of information is provided in our filenames:
  - Most of this information can be considered "obvious" (Energy, generator etc), but it can all be double checked using the AMI database/is contained in the file metadata
  - Less obvious is the information contained in the "tags"
    - e/s/r-tag corresponds to settings used for EVNT/SIM/RECO
      - The correct r-tag is required to apply the correct pile-up profile
    - p-tag corresponds to the settings used to produce the DAOD
      - Important to know, to apply the correct calibrations

```
mc16_13TeV.364100.Sherpa_221_NNPDF30NNLO
_Zmumu_MAXHTPTV0_70_CVetoBVeto.deriv.
.DAOD_SUSY1.e5271_s3126_r10201_p3990
```

- Pros:
  - Quick, easy, for obvious/trivial pieces of information it's probably fine
- Cons:
  - Requires look-up of what the corresponding tags mean
  - Can be difficult to automate: not obvious to catch all tag/calibration cases for special conditions etc when setting up tools.
    (but again, most of this information should be available as in-file metadata)
  - If the tag information is incorrect it can be very difficult to work out what the file contains

2 - Metadata from look-up tables/databases

- Example: The cross-sections/k-factors/filter efficiencies used for reweighting MC yields (to the relevant lumi) is provided by AMI

    - The information from a centrally produced MC sample will appear in AMI, with all of the relevant cross-section information (from the generation step onwards)
        - An independent tool queries the database, putting all information into a .txt file (this is usually then used by the analysers)
        - If this cross-section information is to be changed (k-factors added, cross-sections changed to NNLO etc), then this is provided by the user

- Pros:
    - Usually pretty robust,
    - Simple for analyses to grab the values on-the-fly (but they can be easily accessed/added after running)
- Cons:
    - AMI can be down
    - If something is changed in AMI this can propagate secretly to the analysis level (leading to vastly incorrect MC scaling)
    - Input information (to change cross-sections to NNLO etc) can be added incorrectly (human error, unavoidable)

3 - Metadata from stored file information

- Example: The total number of processed events (for a MC sample) is stored in the file meta-data for the DAODs
  - Derivation-step removes events which are not relevant for the analysis, as such the "total number of generated events" is needed for luminosity scaling

- Pros:
  - Usually as simple as finding the correct saved histogram and grabbing the value
- Cons:
  - In some cases (certain systematic variations) a different histogram is needed than the nominal, if this isn't known beforehand (and accessed) the user will need to rerun the jobs to find the correct values.

4 - Metadata from Twikis

- Examples:
  - The correct Good-Runs-List to apply to certain data periods are documented on a twiki (with links to download the GRL for the analysis).
    The Lumiblock information (provided as content in data files) is then used to check if the run passes the GRL or not
  - The relevant calibrations (for physics objects) can be software-release dependent, this usually requires a check of the tags from the file, to see which release should be used for which calibration (cross-referenced in a twiki)

- Pros:
  - The information provided is usually very explanatory (if the twiki is maintained well), giving analysers a clear overview of what/how to use this information
- Cons:
  - Can require constant checking/updates (during data-taking new GRLs are produced relatively often, calibrations that are release dependent essentially require a database to see what should apply to which release)

Summary

- There are many places in the downstream analysis where accessing the metadata is vital
  - If any of the previous examples (boxes on slide 4) are not applied/applied incorrectly this will negatively affect the physics output
  - A lot of this information will be non-obvious to a beginner, it may be useful to introduce the importance of metadata clearly in analysis tutorials etc

- Each method its own pros & cons:
  - String manipulation of the file name may be considered "dirty" but it is usually simple enough (for an analyser) to take precedence over accessing metadata in full
  - Look-up tables allowing for on-the-fly (and post-running use) are great, but obviously care still must be taken when using such automated services
  - In-file metadata is fantastic, if you know that it's there/that you need it otherwise rerunning a full analysis to acquire the correct metadata on a deadline is stressful
  - Twikis can be great when well documented, and give very useful information, though needing to constantly refer to them for each release/calibration/run can be tiresome

Summary

- Each method its own pros & cons:

  - String manipulation of the file name may be considered "dirty" but it is usually simple enough (for an analyser) to take precedence over accessing metadata in full
    - …Software shouldn't depend upon this if it can be helped though

  - Look-up tables allowing for on-the-fly (and post-running use) are great, but obviously care still must be taken when using such automated services
    - Any issues here must be caught by the analysers

  - In-file metadata is fantastic, if you know that it's there/that you need it, otherwise rerunning a full analysis to acquire the correct metadata on a deadline is stressful
    - Documentation of the in-file metadata required should be provided (and usually is)

  - Twikis can be great when well documented, and give very useful information, though needing to constantly refer to them for each release/calibration/run can be tiresome
    - For fixed information (end of run GRLs etc) this is fine, for ongoing efforts, perhaps something more flexible could be beneficial? (though may be complex to set-up)

Conclusion

- Each method its own pros & cons:

  - String manipulation of the file name may be considered "dirty" but it is usually simple enough (for an analyser) to take precedence over accessing metadata in full
    - …Software shouldn't depend upon this if it can be helped though

  - Look-up tables allowing for on-the-fly (and post
    obviously care still

Generally speaking accessing/using Metadata works well.

Once an analyser knows what they need and where to find it, it tends to be something accessed/applied/implemented once and forgotten about

There may be some aspects which could be improved/made easier and clearer, however metadata usage is usually not a cause for concern

  - well documented, and give very useful information, though needing to constantly refer to them for each release/calibration/run can be tiresome
    - For fixed information (end of run GRLs etc) this is fine, for ongoing efforts, perhaps something more flexible could be beneficial? (though may be complex to set-up)