# Analysis workflow with Easyjet

CAF - ATLAS France users meeting
28/11/2024

Georges Aad (CPPM)


# Introduction

- AthAnalysis based framework aiming to process DAODs and dump ntuples
- Mainly used by different di-Higgs analyses
  - Some analysis use EasyJet as submodule
- HHFramework practical information:
  - Conveners: G. Aad (CPPM), Christophe Roland (LPNHE)
  - Previous conveners (still active): Louis D'Eramo (LPCA), Thomas Strebler (CPPM)
  - Tuesdays bi-weekly 5pm CERN time https://indico.cern.ch/category/17096/
  - atlas-phys-hdbs-dihiggs-hhframework@cern.ch
  - Easyjet Git repository
  - Mattermost

Core software

Analysis

| | |
|---|---|
| EasyjetHub | Object SelectorAlg clean up |
| EasyjetPlus | Enable cppcheck in CI pipelines |
| EasyjetTests | Use bypass config for bbll tests |
| VBSVV4qAnalysis | [VBSVV4q] update name of DAOD_PHYS samples |
| XbbCalib | Object SelectorAlg clean up |
| ZCharmAnalysis | Remove ConstAccessor for HadronConeExclTruthLabelID |
| bbVVAnalysis | bbVV - nullptr check and grid-submit macro fix |
| bbbbAnalysis | Use SysReadDecorHandle for b-tag selection in 4b Sel... |
| bbllAnalysis | Remove ConstAccessor for HadronConeExclTruthLabelID |
| bbttAnalysis | Removing alltop systematics samples list from prod dire... |
| bbyyAnalysis | Use SysReadDecorHandle in BoostedVarsbbyyAlg |
| llttAnalysis | Update PHYS dataset |
| multileptonAnalysis | Object SelectorAlg clean up |
| ssWWAnalysis | Remove ConstAccessor for HadronConeExclTruthLabelID |
| ttHHAnalysis | Object SelectorAlg clean up |
| vbshiggsAnalysis | Object SelectorAlg clean up |

# What is EasyJet

- Common framework implementing CP recommendations through CP algorithms
- Based on AthAnalysis
  - Using Athena CP algorithm block configuration for object calibration, ID, systematics, …
  - Using component accumulator to schedule EasyJet and analysis specific algorithms
  - Input can be DAOD_PHYS or DAOD_PHYSLITE
- Very modular approach
  - Easy to add analysis specific functionalities
- Flexible python interface to configure and steer different tools and algorithms
- Yaml files to set/modify user configuration
  - Converted to Flags available to all python blocks
  - Can be overwritten with command line option
- Ntuple dumper from CP::TreeMakerAlg
  - All systematics available in the same tree
- Easy access to newcomers and for fast studies
  - Configurable mutli-working point support for object selection
  - Easy to plugin computation of new variables
  - Support for several MVA tools through Athena MVAUtils (TMVA, ONNX, …)
  - Configurable slimming/skimming/thinning capabilities

# Configuration and Steering


- Two level configuration based on Python and Yaml
  - All configuration accumulated in AthConfigFlags
- Yaml used to define analysis parameters
  - CP configurations, containers, cuts, output branches …
  - Recursive file merging with possibility to “include” Yaml files
    - Default files containing default blocks and flags
    - Analysis specific Yaml can extend blocks and overwrite existing flags
- Steering of algorithm configuration and sequence done in python
  - Allows dynamic configuration building depending on analysis flags
  - Allows algorithmic checking/merging/extension of flags
- More flexible than ConfigText in athena
  - Dynamic (python) interface to ConfigFactory (CP Algs) and ComponentAccumulator (sequence) rather than static description (Yaml)
  - No need to duplicate properties shared by different blocks

More details in Dan’s et al presentation

# Configuration and Steering


Base file

```yaml
Photon:
  ID: ""
  Iso: ""
  amount: 0
  variables: ["pt", "eta", "phi", "E", "effSF"]
  variables_int: []
  forceFullSimConfig: false
  extra_wps: []
```

Analysis specific file

```yaml
include: EasyjetHub/base-config.yaml
```

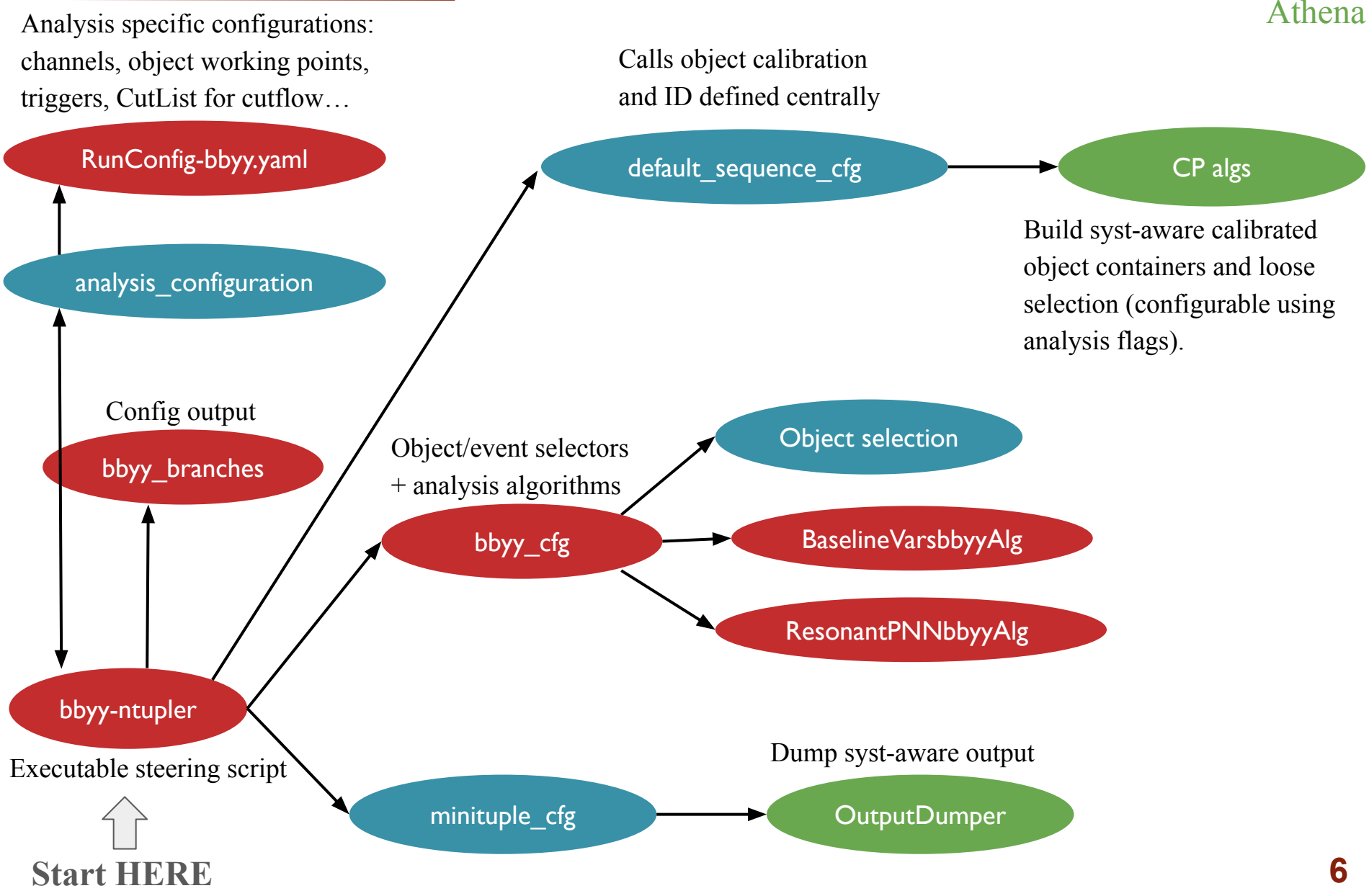
```yaml
# photon ID and isolation requirements
Photon:
  ID: "Loose" # Loose, Tight
  Iso: "NonIso"
  amount: 2
  variables: ["ptOvermyy"]
  variables_int: ["isEMTight"]
  # The first extra wp is used for the final photon selections
  extra_wps:
    - ["Tight", "FixedCutLoose"]
    - ["Tight", "NonIso"]
    - ["Loose", "FixedCutLoose"]
```

Config algs in python using flags filled from yaml

```python
def photon_sequence(flags, configAcc):

    wps = [(flags.Analysis.Photon.ID, flags.Analysis.Photon.Iso)]
    if 'extra_wps' in flags.Analysis.Photon:
        for wp in flags.Analysis.Photon.extra_wps:
            wps.append((wp[0], wp[1]))

    configSeq = ConfigSequence()
    config = ConfigFactory()
    makeConfig = config.makeConfig

    # Temporary hack, we should do this in a more systematic way
    # The config sequence will deal with the systematics suffix
    output_name = drop_sys(flags.Analysis.container_names.output.photons)
    configSeq += makeConfig('Photons', containerName=output_name)
    configSeq.setOptionValue('.recomputeIsEM', False)
    configSeq.setOptionValue('.crackVeto', True)
    configSeq.setOptionValue('.forceFullSimConfig',
                             flags.Analysis.Photon.forceFullSimConfig
                             and flags.Analysis.DataType is DataType.FastSim)

    # PID configuration
    for id, iso in wps:
        configSeq += makeConfig('Photons.WorkingPoint', containerName=output_name,
                                selectionName=id + '_' + iso)
        configSeq.setOptionValue('.qualityWP', id)
        configSeq.setOptionValue('.isolationWP', iso)
        configSeq.setOptionValue('.forceFullSimConfig',
                                 flags.Analysis.Photon.forceFullSimConfig
                                 and flags.Analysis.DataType is DataType.FastSim)
```

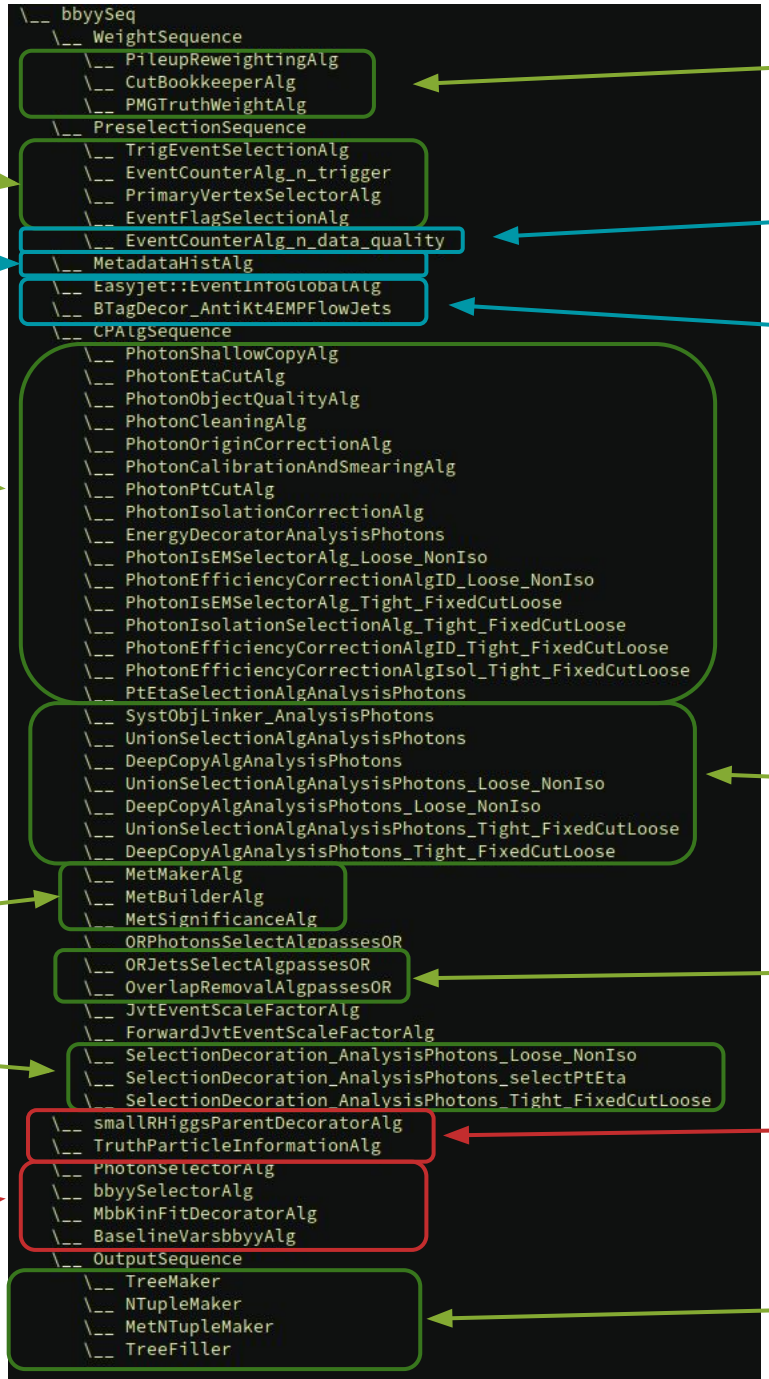
# Example structure (bbyy)

Analysis package
EasyJet Central package
Athena



# Sequence flow

Typical Alg sequence




Add generic event weights

Trigger, PV and quality selection

Add cutflow at any step

Add MetaData

EasyJet adds some handy info

Objects (photons) calibrations and selection from CP algs

Handle (thinned) containers

Build MET after calibration

Perform overlap removal

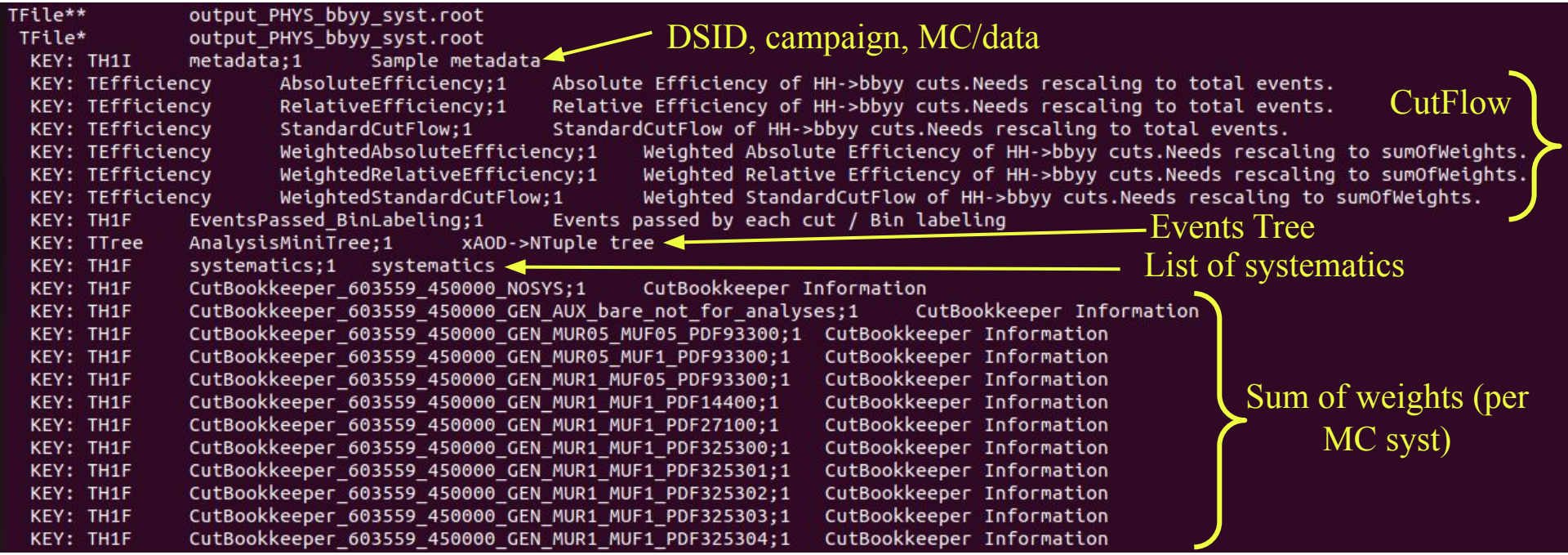
Decorate objects with (multiple) selections

Add analysis specific truth information

Add analysis specific variables and tight/event selection

Dump output

# Output Structure




- Typical structure with all what is needed in typical analysis
  - Metadata, cutflows and sum-of-weights including MC systematics
- Tree containing:
  - Generic events variables
  - Trigger information
  - Vector of object variables after calibrations (configurable variable list and selection)
  - Analysis specific variables (decorated to EventInfo)
  - Truth information
  - Event weights and scale factors

# Systematics

- Systematics handled centrally using Athena
  - Code aware of the list of systematics for each object type
- Systematic selection configurable from Yaml
- All systematics in one tree
  - Optimized duplication of branches that are not affected by specific systematics
  - All objects passing an OR of all systematics are added to the vectors
  - Add “boolean” branches to select which object passes cuts for which systematics

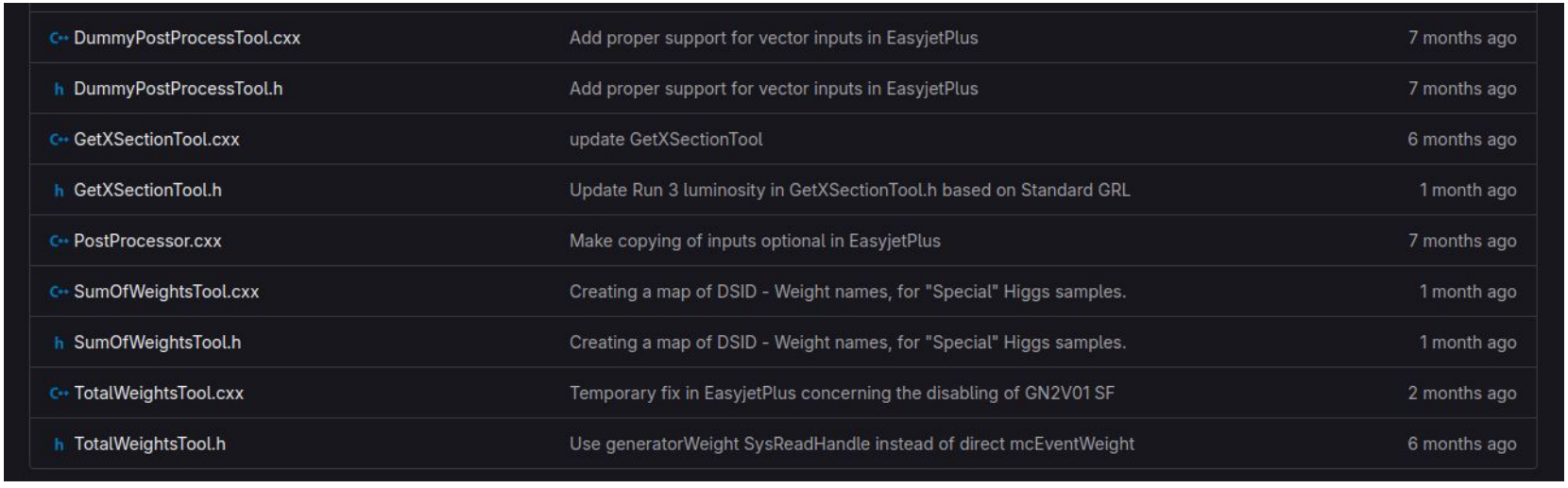
## Traditional


More details in [Minori’s presentation]

## Optimized

# Post-Processing Software

- Post processing is needed before statistical analysis
  - Computation of sum-of-weights and cross-section normalisation factors
  - Potential formatting and skimming/slimming for the fitting tools
  - Developpement of MVAs and fast analysis optimisations
  - Fast checks and data/MC plots
- Several options available and used by different analyses
- Easyjet provides a simple post-processing software with basic functionalities (EasyJetPlus)
  - Also based on AthAnalysis
  - Providing a starting point but underdeveloped for now
    - Getting cross-sections from PMG
    - Computation of sum-of-weights and sample normalisation
    - Skimming and slimming functionalities can be added if there is interest from the users



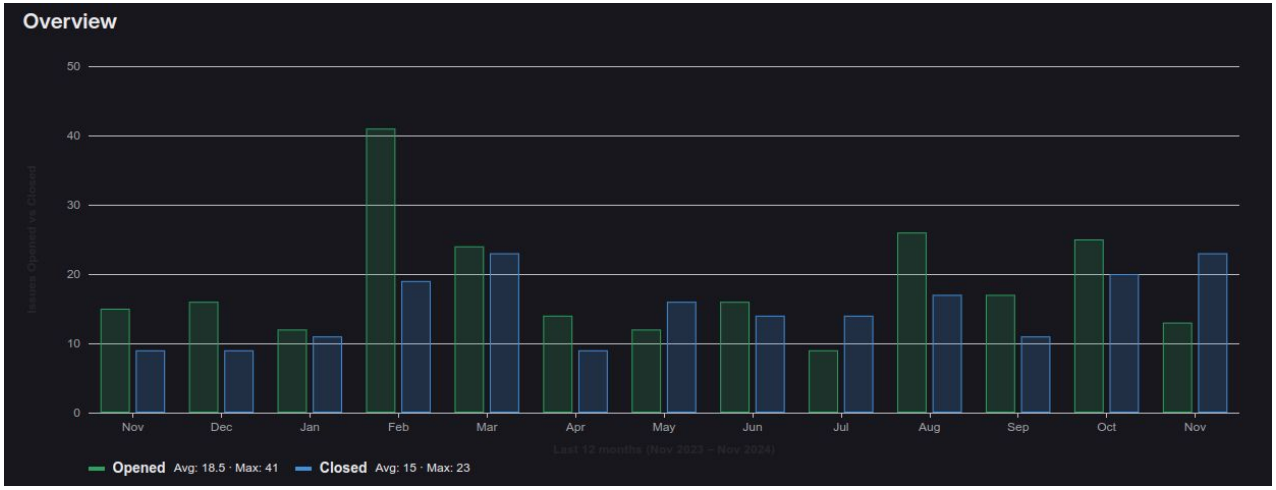
| | | |
|---|---|---|
| DummyPostProcessTool.cxx | Add proper support for vector inputs in EasyjetPlus | 7 months ago |
| DummyPostProcessTool.h | Add proper support for vector inputs in EasyjetPlus | 7 months ago |
| GetXSectionTool.cxx | update GetXSectionTool | 6 months ago |
| GetXSectionTool.h | Update Run 3 luminosity in GetXSectionTool.h based on Standard GRL | 1 month ago |
| PostProcessor.cxx | Make copying of inputs optional in EasyjetPlus | 7 months ago |
| SumOfWeightsTool.cxx | Creating a map of DSID - Weight names, for "Special" Higgs samples. | 1 month ago |
| SumOfWeightsTool.h | Creating a map of DSID - Weight names, for "Special" Higgs samples. | 1 month ago |
| TotalWeightsTool.cxx | Temporary fix in EasyjetPlus concerning the disabling of GN2V01 SF | 2 months ago |
| TotalWeightsTool.h | Use generatorWeight SysReadHandle instead of direct mcEventWeight | 6 months ago |

# Management and Support

- Project managed in git
  - Git issues and merge requests
- CI for basic automatic checks
  - Compiles and runs
  - Plans to add more checks on the output
- Quick update following athena releases
- Human supports on mattermost and bi-weekly Zoom

~50 merge request per months


Active community



# Grid processing

More details in Georges' presentation

- Started to gain experience with the grid
  - More understanding and optimisations still needed (progressing)
- Collecting information from the users
  - No complaining about running nominal MC
  - MC with systematics taking long time (more than a week)
    - Needs tuning and babysitting of prun commands
    - Using scouts to define the job parameters not efficient (timewise for us)
  - Running on data is also long (more than a week)
    - Also tuning needed
- Running on PHYSLITE is especially constraining
  - More than 100k events per file (smallest possible jobs)
    - Cannot split file with nEventPerJob and SkipEvents due to metadata
    - No way now to ensure correct sum-of-weight for normalisation if we split files in several jobs
- Trying to optimise the job submission and the code
  - Avoid needing to split jobs
  - Last resort is to split systematics into 2 (several) blocks

# Grid example: ttHH ntupler on ttbar sample

- Checking as an example one task running ttHH analysis on a ttbar sample
  - Task link
- Running with systematics
  - Not sure about the analysis strategy (running MVAs, CPU/memory intensive tasks, …)
  - Not sure about the exact number of systematics but should be comparable for most analyses

```
prun --inDS mc20_13TeV.410472.PhPy8EG_A14_ttbar_hdamp258p75_dil.deriv.DAOD_PHYSLITE.e6348_s3681_r13167_p6266 --outDS user.awierda.051_sys_GN2v01.2024_10_03_T184904.410472.e6348_s3681_r13167_p6266 --notExpandInDS --exec "ttHH-ntupler %IN -l --run-config config.yaml --out-file output-tree.root" --mergeOutput --outputs TREE:output-tree.root --writeInputToTxt IN:in.txt --useAthenaPackages --nGBPerJob 2 --athenaTag AthAnalysis,25.2.25 --inTarBall code.tar.gz
```

Good for small output but can take significant extra time

Large number of events per file O(100k)

Leads to one file per job for the dataset
448 jobs for 448 files

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 41524513 | analy | panda-client-1.5.78-jedi-run | Renske Wierda | — | finished | 448<br>(99%) 447<br>(0%) 1 | 54,681,000<br>(99.6%) 54,481,000 | 929.6M<br>370.7M<br>369.8M<br>881.6k | 2024-10-03 17:03:54<br>2024-10-16 06:38:31 | 1 | 1001<br>1000 | 0 | 18kg<br>18kg<br>43g |

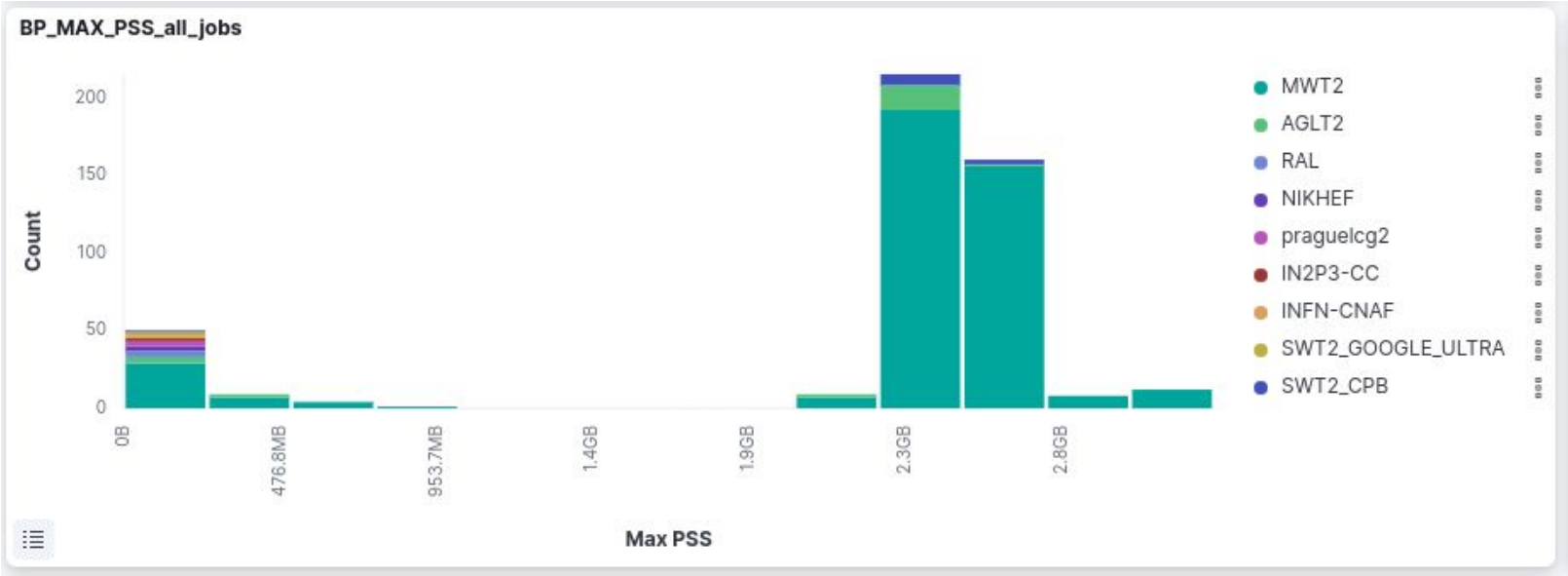
1 job failed after many attempts

More than 12 days from start to finish

# ttHH on ttbar job: Memory

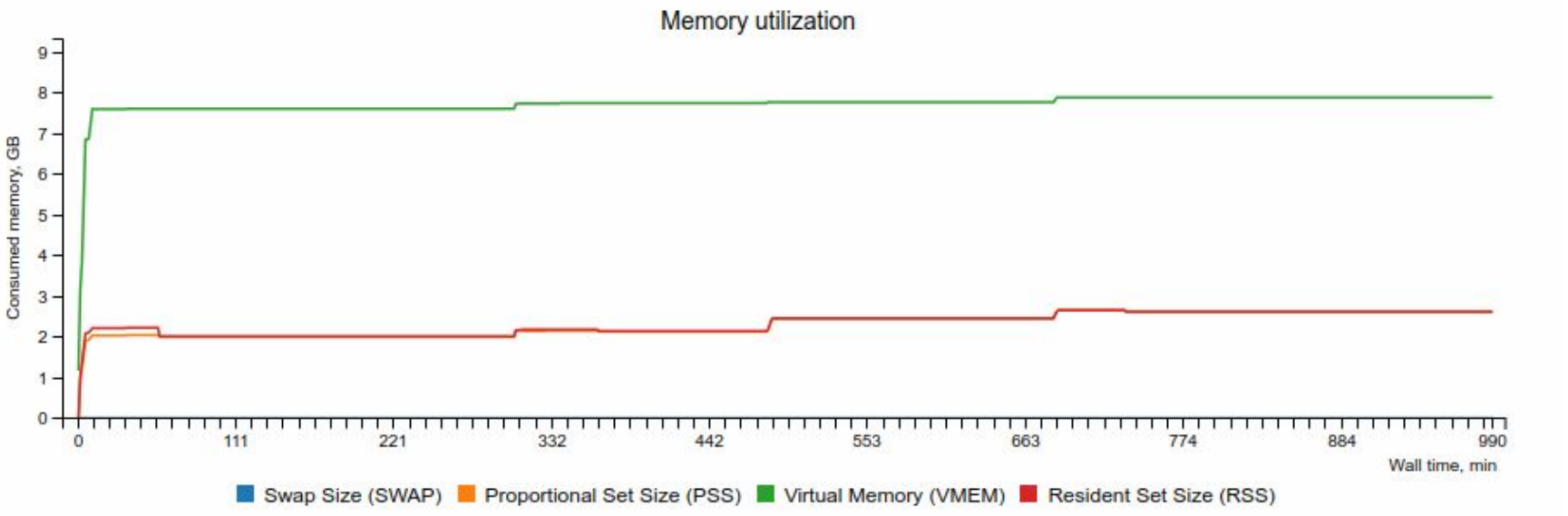
- Memory around 2.5 GB per job
  - Max around 3 GB which might create some problems on some sites (some have 2.6 GB limit)
  - But it does not seem to create any major issues

All plots are taken from the panda website
Many more are available per task to help with diagnostics



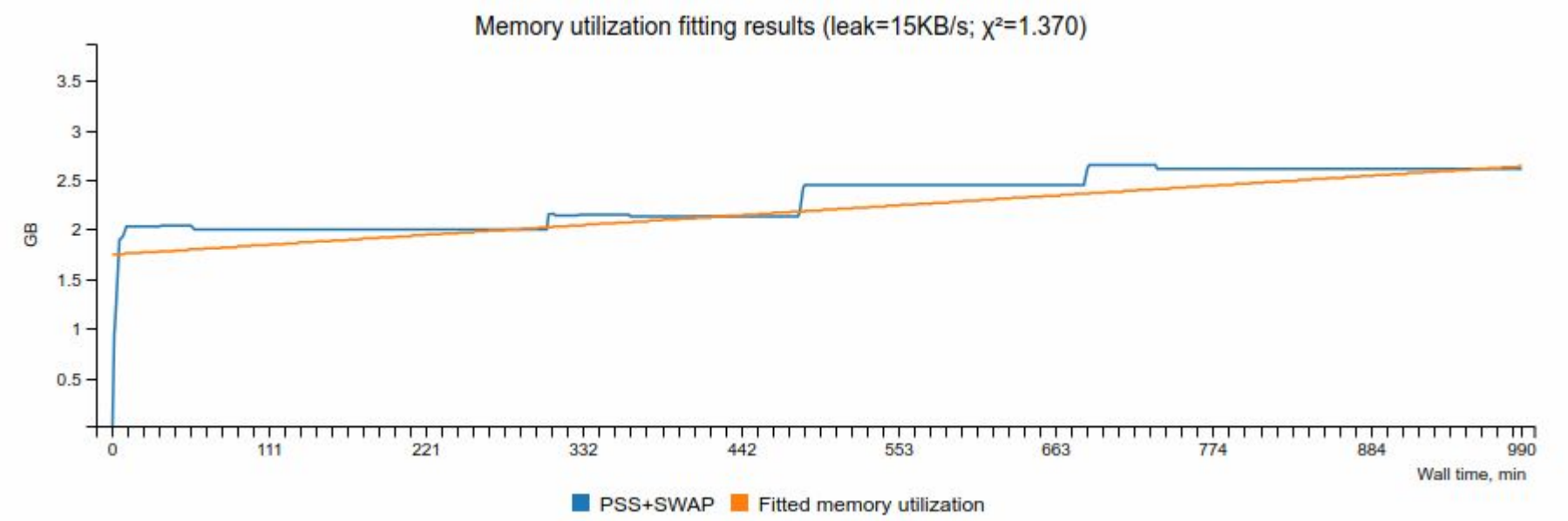
# ttHH on ttbar job: Memory

- Look at PSS and/or RSS (don't care about virtual)
  - Swap should be 0
- Memory increases at the beginning and stays relatively stable until the end of the jobs
  - Reasonable long jobs should not be killed do to memory issues
- No swap memory is used
  - Otherwise the job can take a huge amount of time and never finishes



# ttHH on ttbar job: Memory

- Estimated memory leak plot available
  - Done by iterative linear fit removing first events with a sharp increase
  - Not perfect but can give an idea
- 15 KB/s leak found in this job
  - Can’t exclude a small leak but it is most probably less than the given value
  - This needs a more thorough check (not our priority for now)



# ttHH on ttbar job: Wall time

- Most jobs finish in less than one day
  - Maximum less than 3 days
- Why the task took 12 days then?
  - 448 jobs is not a huge number and should mostly run in parallel

# ttHH on ttbar job: History

- Looking more carefully at the task profile in time
  - Up to 15 attempts before finishing

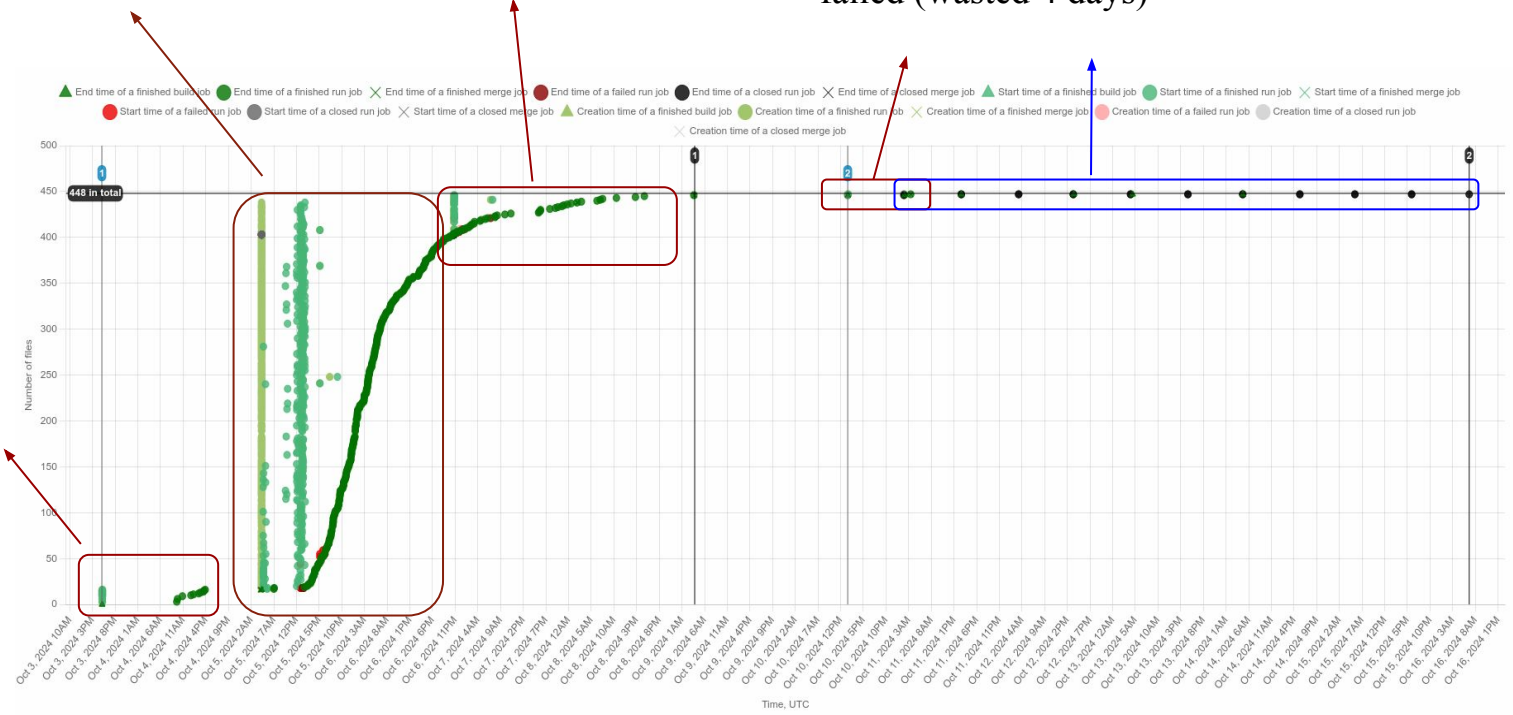
| attemptnr (16) | 0 (9) 1 (417) 2 (25) 3 (4) 4 (2) 5 (1) 6 (1) 7 (1) 8 (1) 9 (1) 10 (1) 11 (1) 12 (1) 13 (1) 14 (1) 15 (1) |
|---|---|

Bulk submission
93% done after 4 days
Jobs take ~1 day of CPU

Second and third attempt of failed jobs
98.5% done after 6 days

Last job finished with attempt 4 after 8 days
Followed by 11 attempts for the last job that failed (wasted 4 days)

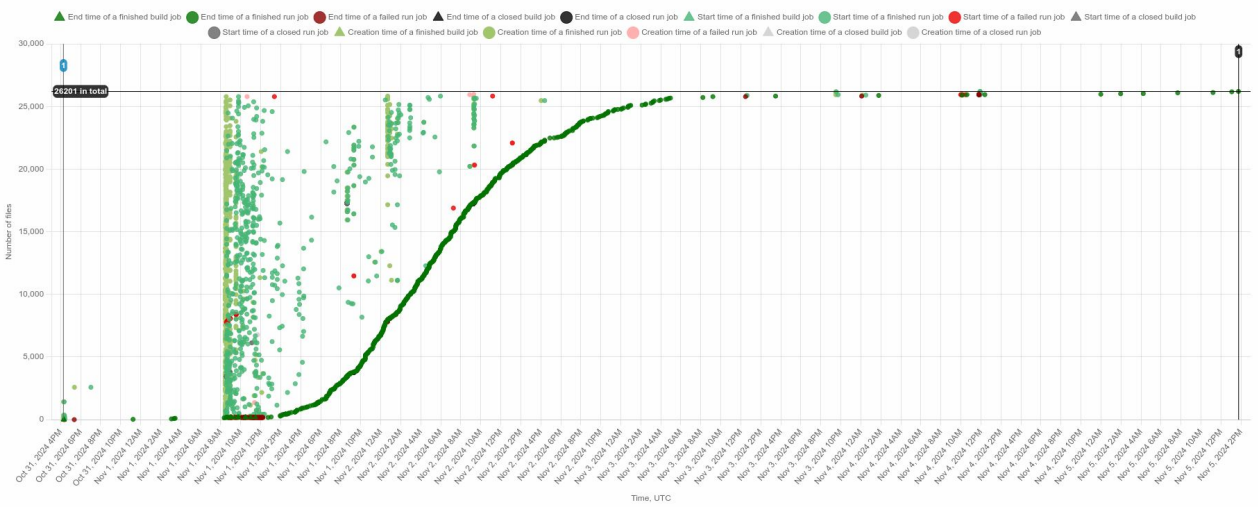
Scouts and sites availability



# Running on data (example)

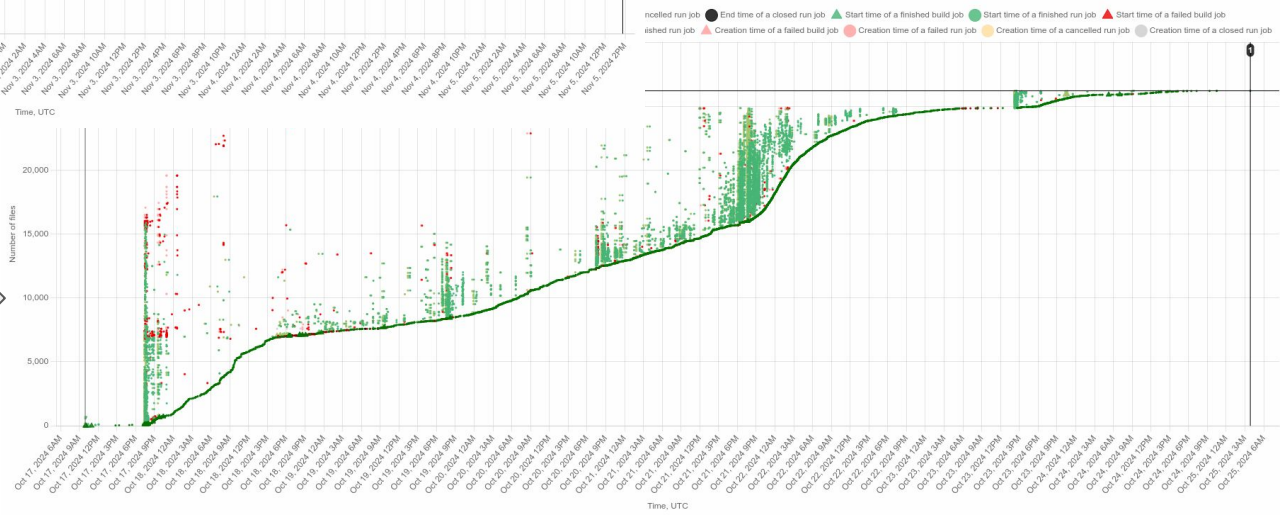
- Work in progress, detailed look not yet done
- Few preliminary conclusions
  - Large number of files distributed on the grid
  - Harder to tune than MC (big spread in Wall time between jobs)
  - But could finish in few days using large nGBPerJob

The two tasks are running similar ntuple dumper but not exactly the same config




⇦ 100GB per job: 6 days

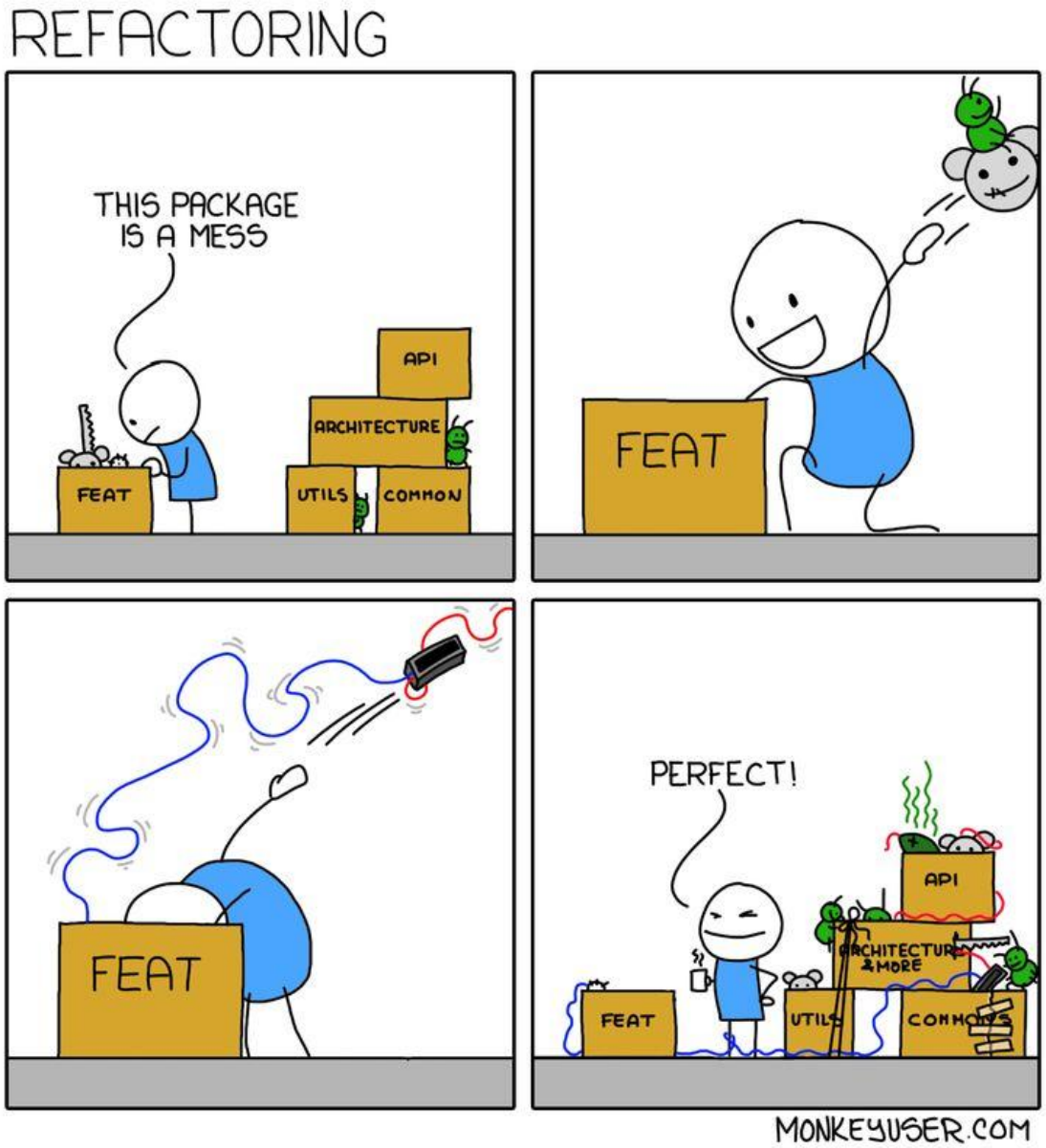
Let scouts do it all: 8 days ⇨



# Conclusions

- EasyJet framework steadily spreading within HH analyses and beyond
  - Demonstrated flexibility to accommodate various analyses needs
- Active developments from the community
  - Improvement of core software
  - New functionalities to both Easyjets and Athena CP algorithms depending on analysis needs
  - Flow-up of CP recommendations and quick updates following AthAnalysis releases
  - Optimisation of output ntuple size
  - Analysis specific developments
- Reliable core performance metrics to be developed
  - Several analyses running nominal on the grid without problems
  - Tuning of jobs with systematics ongoing
- Skeleton post-processing code provided
  - But most analysis using other post-processing frameworks
- More details in the HHFramework tutorial presentation from Louis and Thomas
- Future structure evolutions
  - As more analyses are added it is becoming harder to maintain everything in one repository
  - Considering the use of sparse checkout or splitting to sub-modules
  - Splitting allows central installation of the core package in the future

# Backup


REFACTORING
THIS PACKAGE IS A MESS
API
ARCHITECTURE
FEAT
UTILS
COMMON
FEAT
FEAT
PERFECT!
API
ARCHITECTURE & MORE
FEAT
UTILS
COMMONS
MONKEYUSER.COM