The Evolution of Analysis Models for HL-LHC

Andrea Rizzi, INFN/University of Pisa

November 7th, 2019 - CHEP2019 - Adelaide

Caveat

Credit (for good ideas):

The material presented here is based on discussions and presentations in the HSF DAWG meetings, on pre-CHEP2019 workshop

Blame (for bad interpretations):

The overall view and some ideas are clearly my personal view, and are not necessarily shared by the whole CMS or ATLAS communities.

Also: not covering realtime analysis, as that is a quite different (and cool) topic

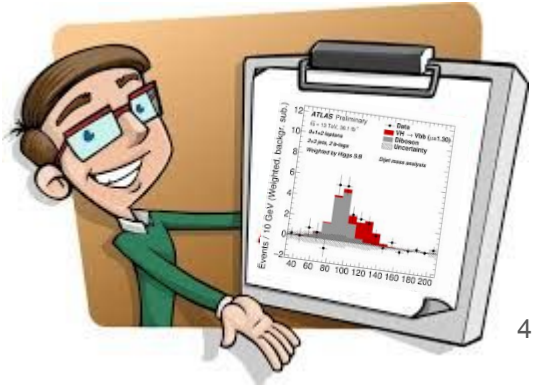
Outline

- What is an analysis model?
- A short history of LHC analysis models
- Difficulties and frustrations in complex experiments
- Ideas for future analysis models

What is an analysis model?

- The computing guy answer:
  - “Specify the way you access the data and the cpus (or other resources), define the tools to use, datasets size, distribution model etc”

- The analysis guy answer:
  - “An analysis model describes how you make a plot, how input variables are grouped and named, how you do systematic variations, reweighting and how you can prepare inputs for fitting tools and/or combine your results with other analysis”



(yeah, this is naive picture, there are plenty of people with deep understanding of both computing and actual LHC analysis)

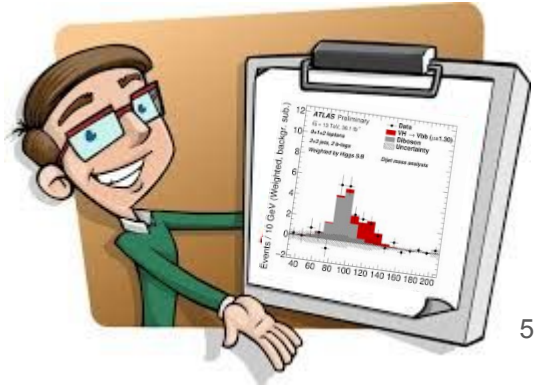
LHC Run1 analysis model (“make do”-model)

The computing guy description:

“*We had Tier0,1,2. We centrally run reconstruction at Tier0-1 then user analyses jobs were run on Tier2 using AOD or XYZ format to make ntuples/plots.*”

The analysis guy description:

“*I used the CERN/MIT/ABC-group ntuples (Jane Smith was doing the ntuples from AOD) with the plotting macros we developed in our institute to make the plots, then we used RooFit to extract Higgs Boson signal significance*”

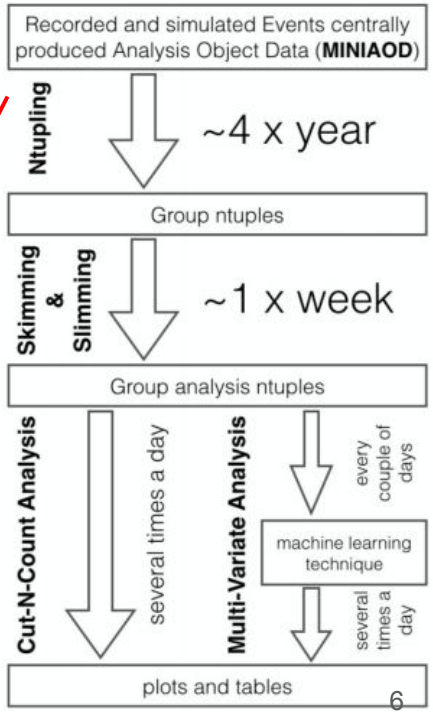


Run 2 (“hic abundant leones”-model)

- Run1 model not scaling to Run2
  - From few billions to tens of billions of events
- Organize the analysis **steps for data reduction**
- Not addressing the final steps:
  - I.e. *“please use your local batch or laptop for plots and fits”*

primary xAODs	produced centrally, stored on grid	single definition	xAOD Format	no corrections
DxAODs		defined by physics group		only AODFix
mini-xAODs	produced by group/analyzer, stored locally	defined by group/analyzer		some/all corrections
n-tuples			TTree	all corrections
histograms	produced by analyzer, stored locally or personal machine	very analysis dependent	(mostly) TH1	
results			analysis dependent	

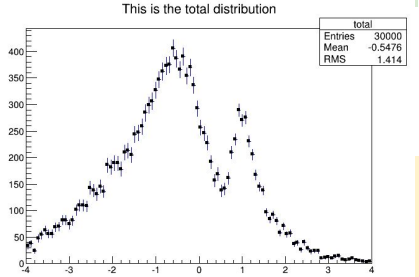
Grid



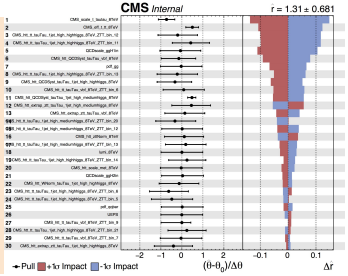
About speed: how fast are Run1/Run2 analyses?

seconds minutes hours days

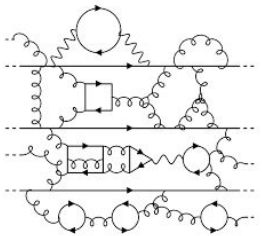
Quick and dirty plot(s) on a single sample



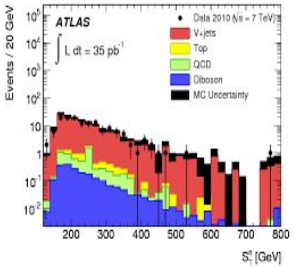
All systematic variations evaluating impact

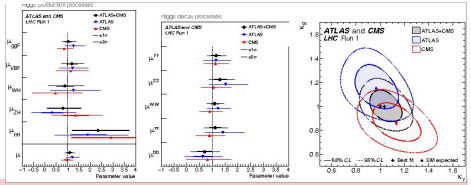


Evaluate a “Matrix Element Method”



All control plots in paper format with some systematics, all backgrounds, etc...

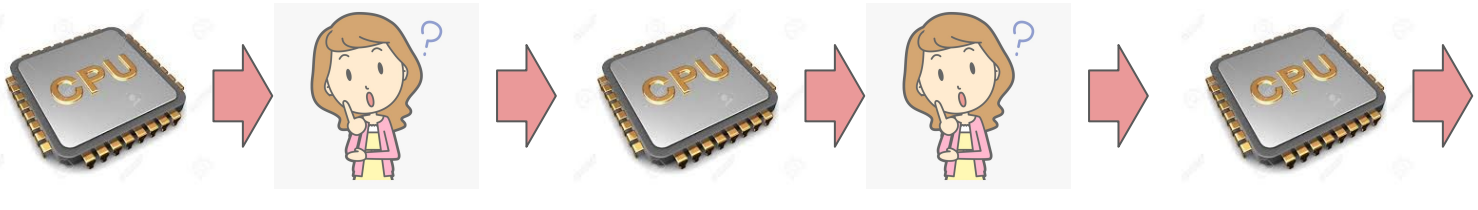


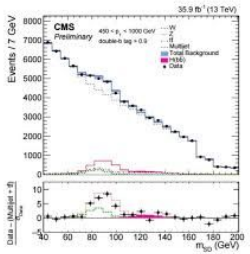


Evaluate a full Higgs combination

Human machine interaction: fast + interactive

- People time is more expensive
  - Exploring ideas, testing new cuts, new techniques, multiple trainings
  - Cannot predict needed human time (e.g. find a bug)





  - Empty human brain cycles are a waste
    - each time I'm waiting idle for my grid jobs a unicorn dies

- I/O as a bottleneck?
  - **Yes** (not always, depends on data source and analysis speed)
  - CPU time/event/core: [10us to 1s]
  - Data to read/event: [100 bytes to few kb]

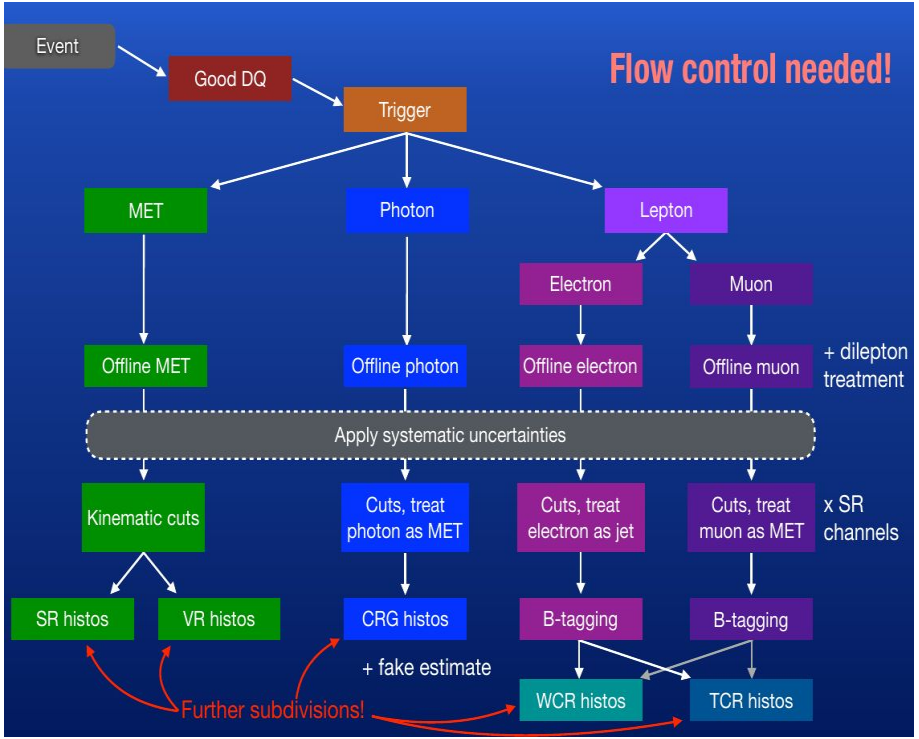
Up to ~100 Mbit/core
(google has 2Gbit/core, do we?)

Are LHC analyses just same as any data science analysis?

(or can we just use the python ecosystem and all the goodies there?)

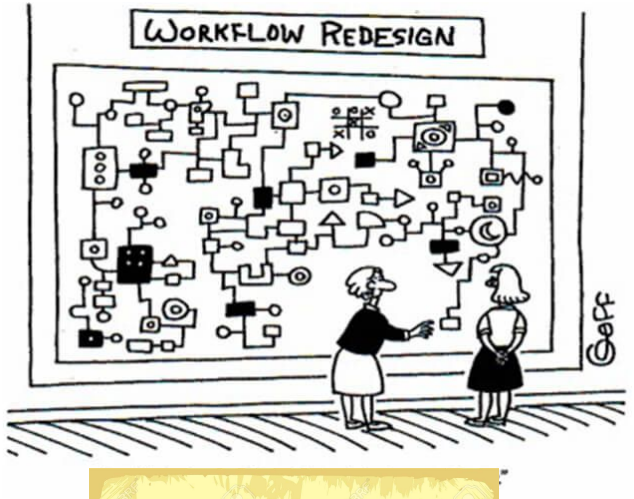
What is a LHC analysis ?

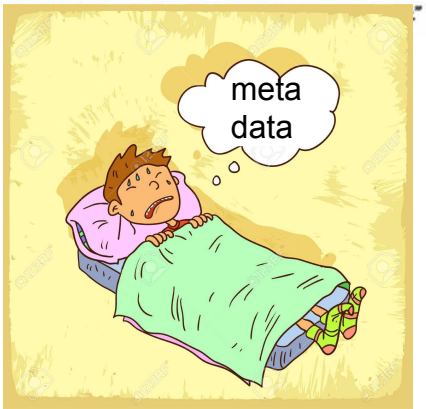
- **Different datasets**:
  - MC(s) vs data
  - different years
  - different final states
  - different phase space regions
- Construction of **models** for S & B
- Estimation of **uncertainties**
- **Partial data observation** (control samples, blinding, etc..)
- **Statistical** interpretation
- Peer reviewing, **reproducibility** & preservation



Complex experiments environments

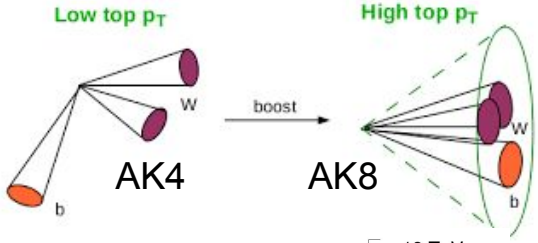
- Mixing of data and metadata:
  - Detector calibrations, changing over time
  - Good runs lists, lumi, efficiencies, and other per dataset metadata
- Development and production are not separated
  - Last-minute change evading your planned analysis model
  - Often need to fetch some missing event branch/column
- Scattered information
  - Events from multiple, partly overlapping, datasets (common format, mostly well organized…)
  - Metadata **nightmare**: some “in file” (in different formats), some in “DB”, some in “twikis/web pages”, some from oral tradition, etc...

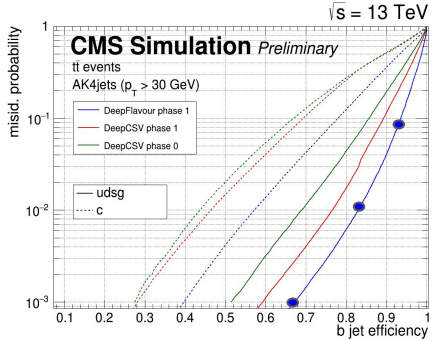




The curse of freedom

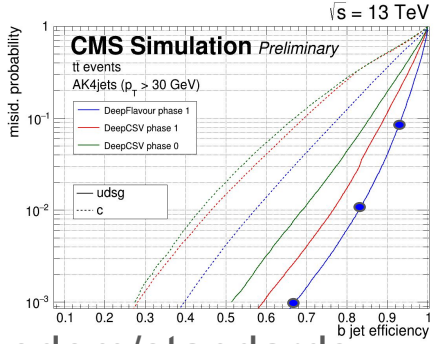
- *"This choice is analysis dependent"* (phys-coord level statement)
  - *"I do not want to take the responsibility to make a decision"* (object group statement)
  - *"I do not know what to use, hence I make a semi-random choice, then I later defend it because I do not want to change my ntuples/ROOT macros"* (student/analyzer confession)
- Analysis choices for:
  - Reco algorithms (my jets/muons are better than yours)
  - ID algorithms
  - Cuts working point
- Do we really need this freedom?
  - Partly...
  - Different final states have different needs
    - Many leptons vs many jets
    - High pt vs low pt
  - …but a few working points are likely enough

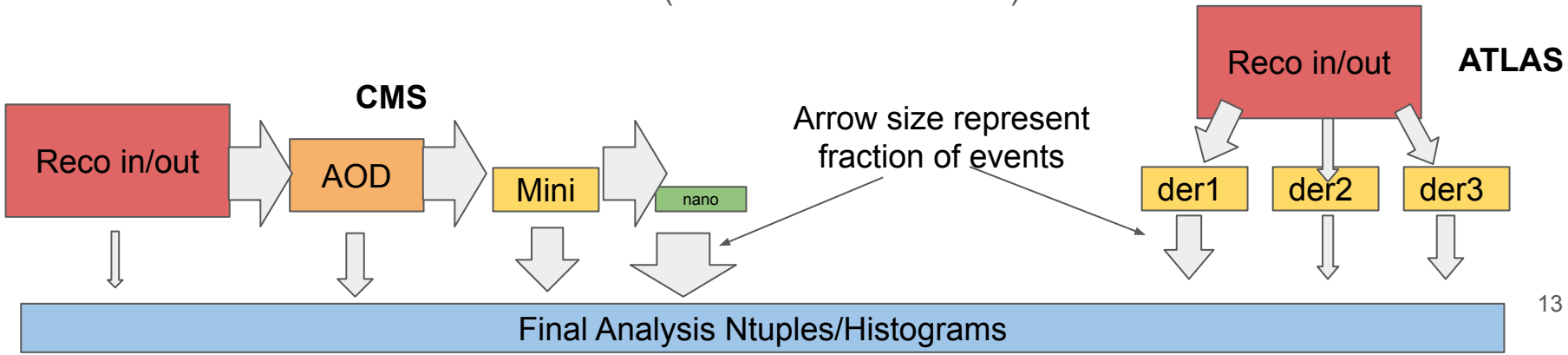




Freedom and standards

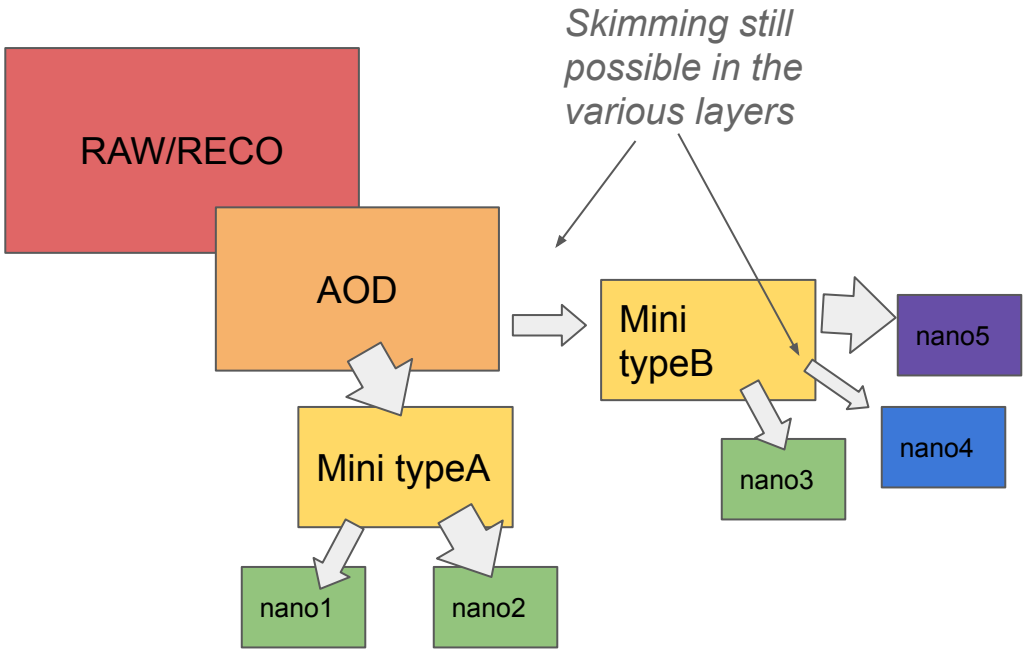
- Freedom is good, so are standards
- Ideal environment
  - Retain **flexibility** to allow any level of customization
  - Provide **good defaults** possibly in a few flavours
  - Keep “defaults” **up to date** with latest greatest idea
- Both CMS and ATLAS Run2 solutions had to trade off freedom/standards
  - ATLAS trains/derivation framework (standardize the skimming code/infrastructure)
  - CMS skims/aod/miniaod/nanoaod (standardize the content)





Hybrid CMS - ATLAS model (Run3?)

- The CMS **1kb/ev NANOAOD** can be prepared in multiple versions:
  - Jet calibration NANOAOD with additional jet stuff
  - Dedicated B-Physics nanoaod
- ATLAS is defining **lite formats** similar to CMS MINI/NANOAOD
  - DAOD_PHYS: 50kb/ev
  - DAOD_PHYSLITE: 10kb/ev



HL-LHC datasets numbers at ~100B events/year

Data tier	Hard event size	Size per PU event	Total Run4 size
AOD	300 kb/event	5 kb/PU ev	100 PB/year
MINIAOD	30 kb/event	0.5 kb/PU ev	10 PB/year
NANOAOD (perhaps x2-3 versions?)	1.5 kb/event	10 - 50 bytes/PU ev (no tracks, little PU scaling)	300 TB/year (x2-3 ?)

Devel for:

Run1 ⇩ Run2 ⇩ Run3 ⇩ Run4

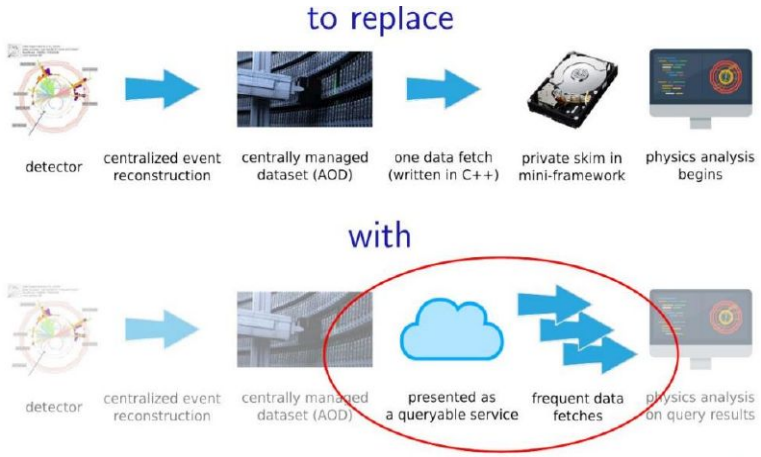
- How about a **PICOAOD**?: 100-200 bytes(i.e. ~50-100 Float16s) ~10-20TB/year
  - Perhaps in Run5? (after we understood new detectors in Run4)
  - Just 4-vectors of hard objects
  - Actual content depending on the final state?

Standardize further formats and software

- Improving event formats:
  - Find **real** differences in analysis choices (and suppress the fake ones)
  - Port analysis code to central software (so that every analysis can benefit)

SUSY selection Higgs selection

- Improving software standardization:
  - Avoid multiple version of ntuplizing code
  - Common services (or at least common code and formats) for similar operations



(see serviceX talk)

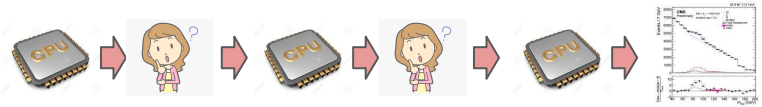
Towards plots

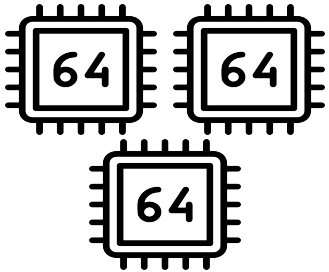
Analysis frameworks

- Dozens of homemade frameworks for plots & ntuples (full of features!)
  - Including interface to statistical interpretation tools
- Common patterns emerging:
  - Prefer configs for cuts and histogram definition instead of actual code
  - Multiple levels of caching (reduced ntuples)
- Total CPU time typically negligible wrt production, but speed matters
- Often poorly written code, rarely parallel
  - Lot of room for improvements
  - Many low hanging fruits (multiple event loops, useless copies of data, …)

Resources and interfaces for analysis

- Need fast interactive (low latency) access to data
- “Laptop analysis” likely not scaling to HL-LHC luminosity
- Developing concept of “analysis facility”
  - Not replacing but complementing current grid
  - Higher I/O requirements
  - Quick availability of hundreds of cores





OR

Possible game changer: Declarative Analysis Languages

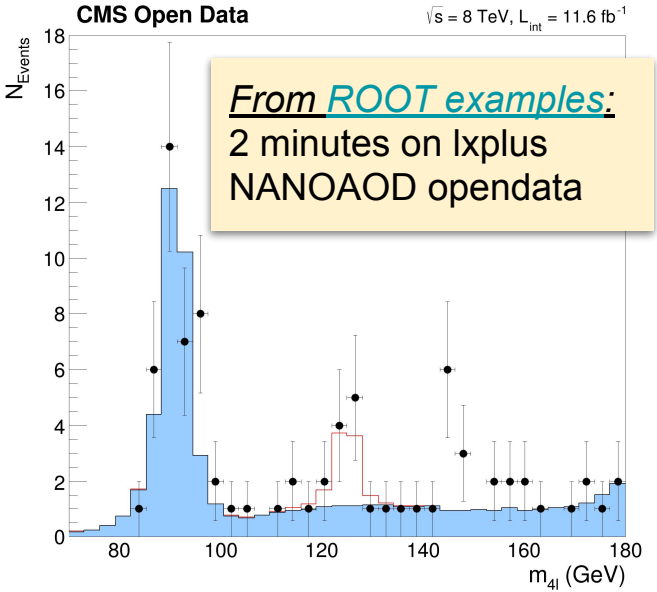
- Describe “What” instead of “How”
  - let a language interpreter/compiler create optimal code (perhaps assuming some defaults)
- Implement HEP specific operations (on top of standard data science ones)
  - Matching objects, handling 4-vector algebra, calibrate, cross clean, etc...
- Several prototypes (see Gordon’s slides from tuesday)
  - Mainly addressing “event => histogram” description
  - Could cover also model/templates building for fitting, fitting feature extraction, fetching of relevant dataset, requesting MC prods

```
DiMu_controlRegion:
    weights: {nominal: weight}
    selection:
        All:
          - {reduce: 0, formula: Muon_pt > 30}
          - leadJet_pt > 100
          - DiMuon_mass > 60
          - DiMuon_mass < 120
          - Any:
            - nCleanedJet == 1
            - DiJet_mass < 500
            - DiJet_deta < 2
```

```
#Match jets to selected loose Leptons
flow.Define("Jet_p4","@p4v(Jet)")
flow.MatchDeltaR("Jet","Lepton",embed=(["pt","pid"],[]))

# * Jet select/cleaning against loose leptons , jet pt > 25 , jet id
flow.DefaultConfig(jetPtCut=25,jetIdCut=0,jetPUIdCut=0)
flow.SubCollection("CleanJet","Jet",'''
 Jet_pt > jetPtCut &&
 Jet_jetId > jetIdCut &&
 Jet_puId > jetPUIdCut &&
 (Jet_LeptonIdx==-1 || Jet_LeptonDr > 0.3)
''')
```

New technologies in HEP

- Interesting new technologies for efficient analysis
  - RDataFrame (C++ / python with C++ snippets), Coffea (python)
  - zfit, new ROOT features, pyhep stuff, etc..
- Declarative Analysis Language can exploit efficient backends



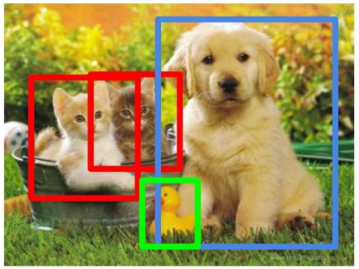
From ROOT examples:
2 minutes on lxplus
NANOAOD opendata

Example coffea syntax

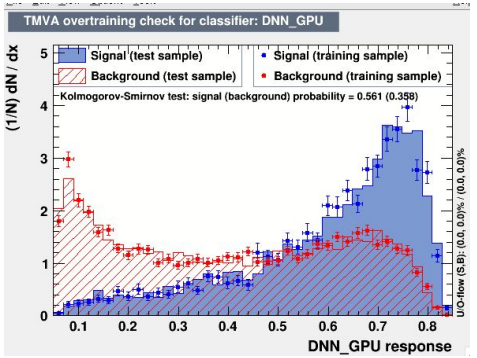
```
ele = electrons[(electrons.p4.pt > 20) &
                                    (np.abs(electrons.p4.eta) < 2.5) &
                                    (electrons.cutBased >= 4)]

mu = muons[(muons.p4.pt > 20) &
                                (np.abs(muons.p4.eta) < 2.4) &
                                (muons.tightId > 0)]
```

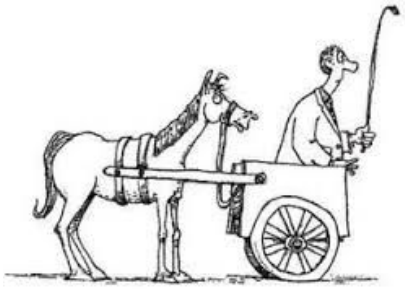
Industry/data science vs HEP

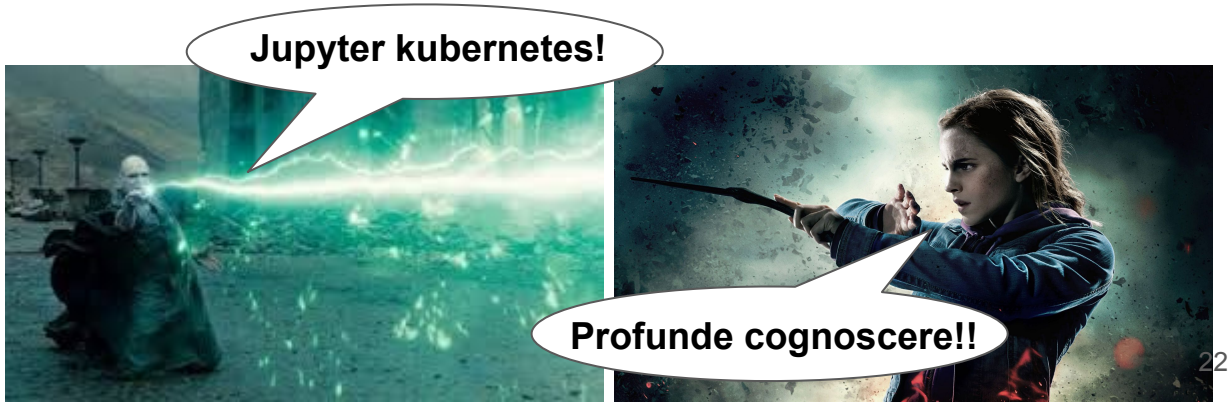
- Industry has no systematics to evaluate
- HEP analysis != recognize cats/dog/duck
  - An analysis is more than **S**/**B** separation
- **Virtualization** is good, **GPU** are useful, **deep learning** is great, etc… but...
  - Let's not use **buzzwords** as magic spells
  - Understanding of the problem comes before proposing a solution for it



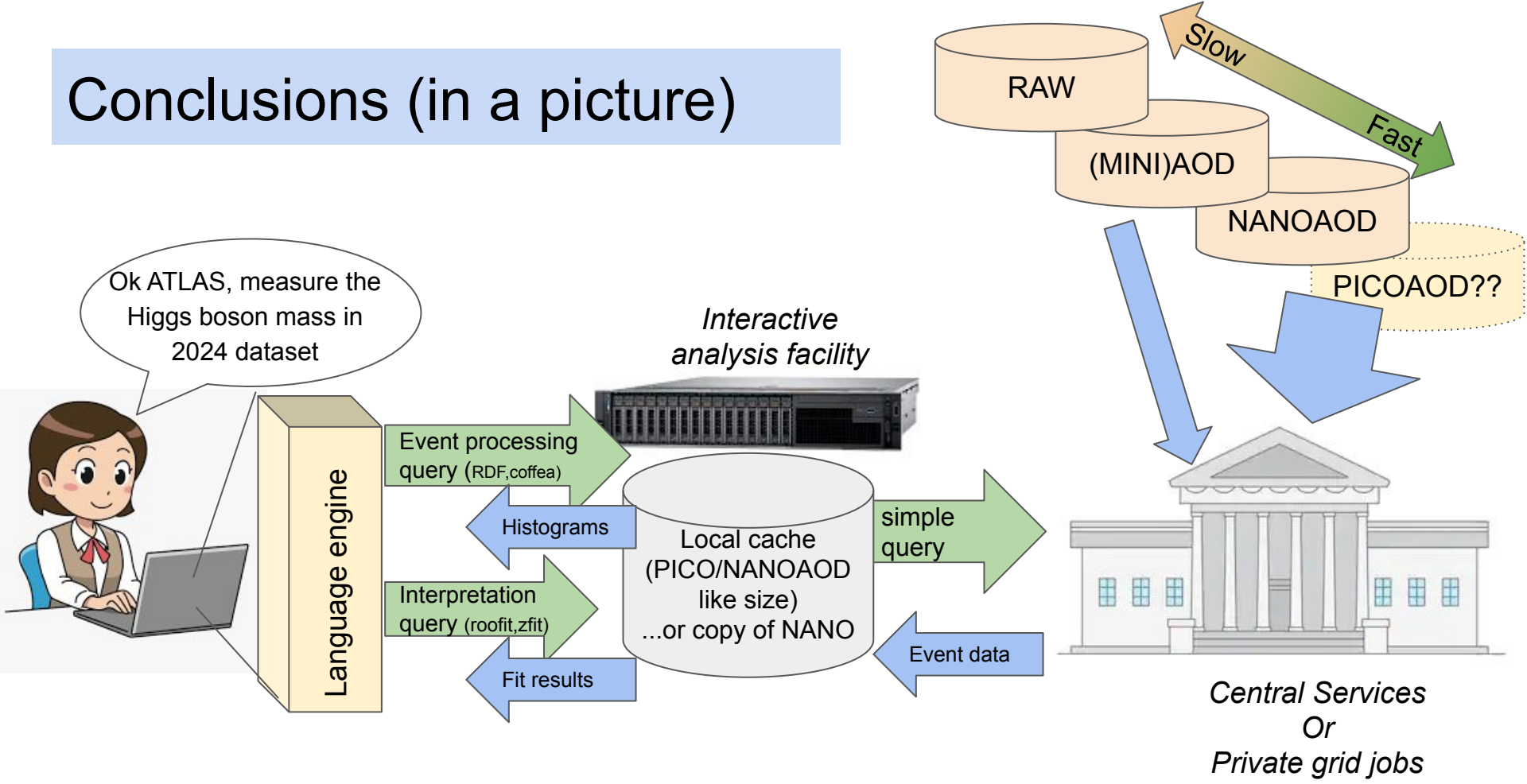
CAT, DOG, DUCK







Conclusions (in a picture)



Backup

Scaling *or why Run2 to Run3-4 is not as Run1 to 2*

- Lumi profile foresee long flat(ish) periods
  - You cannot just “forget 2016” anytime soon (while we (almost) forgot Run1 delivered lumi)
- No energy steps
- HL-LHC is all about integrated luminosity
- Higgs mass does not increase
  - Early skimming not going to reduce much
- Pileup scaling
  - RECO, AOD, MiniAOD size increases with pileup (n-tracks goes up, N-jets goes up)
  - NANOAOD not scaling with PU (Muons, Electrons, high pT jets, etc... are only in the hard event)