HEP software experience

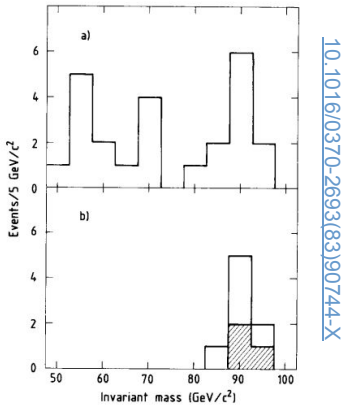
A. C. Marini, K. Pedro, S. Rappoccio, S. Wunsch

On behalf of the CMS Collaboration

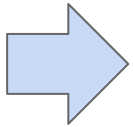
Introduction

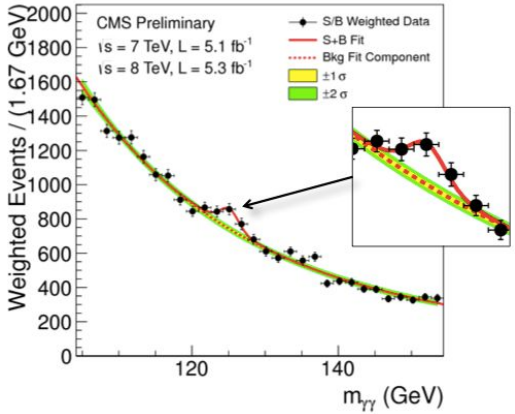
HEP community changed a lot through the years:

- Information available is larger
- Analysis are becoming more complex

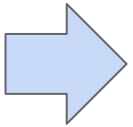


10.1016/0370-2693(83)90744-X

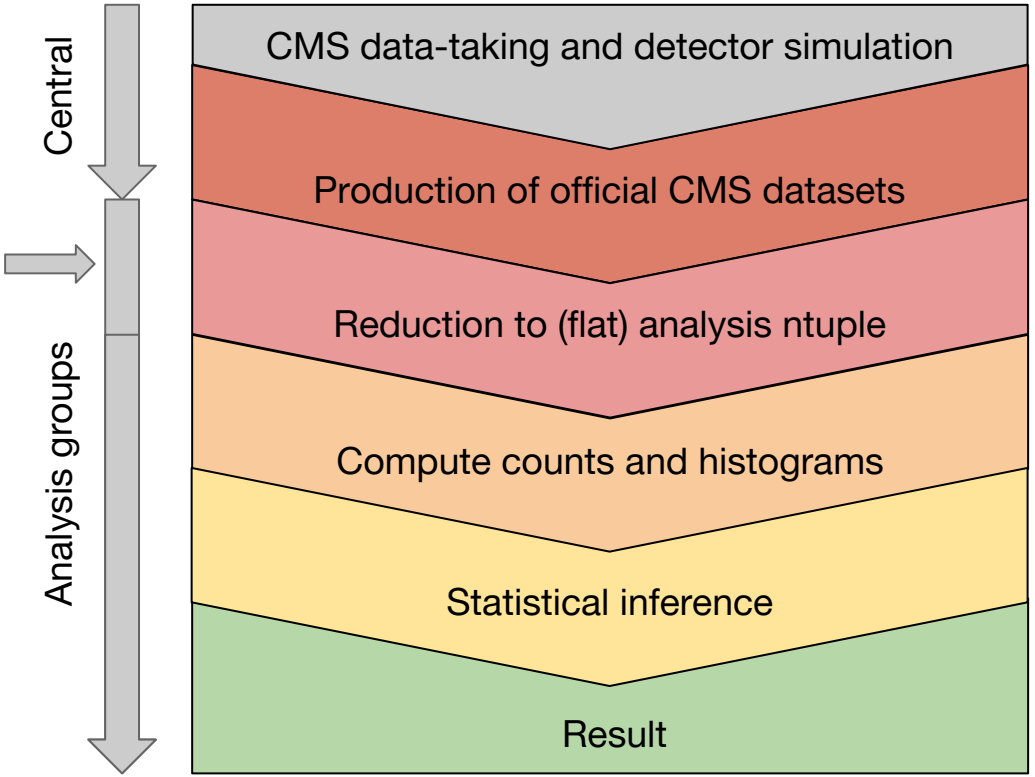




10.1016/j.physletb.2012.08.021



Analysis workflow



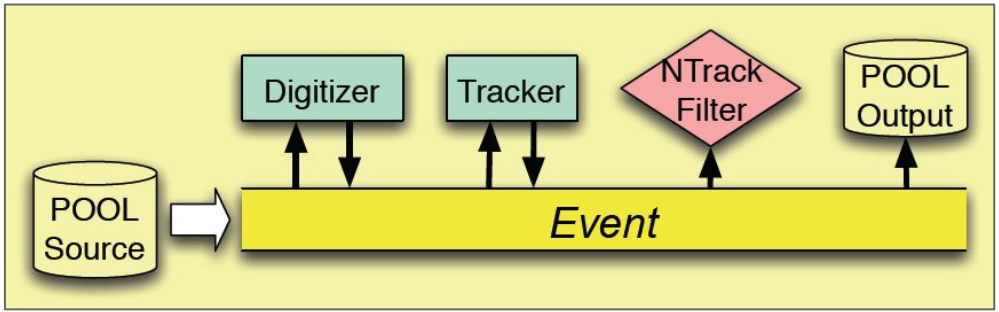
Performed with NanoAOD centrally

May include:
Custom reconstructions
Application of corrections
Calc. of systematic shifts

Central software

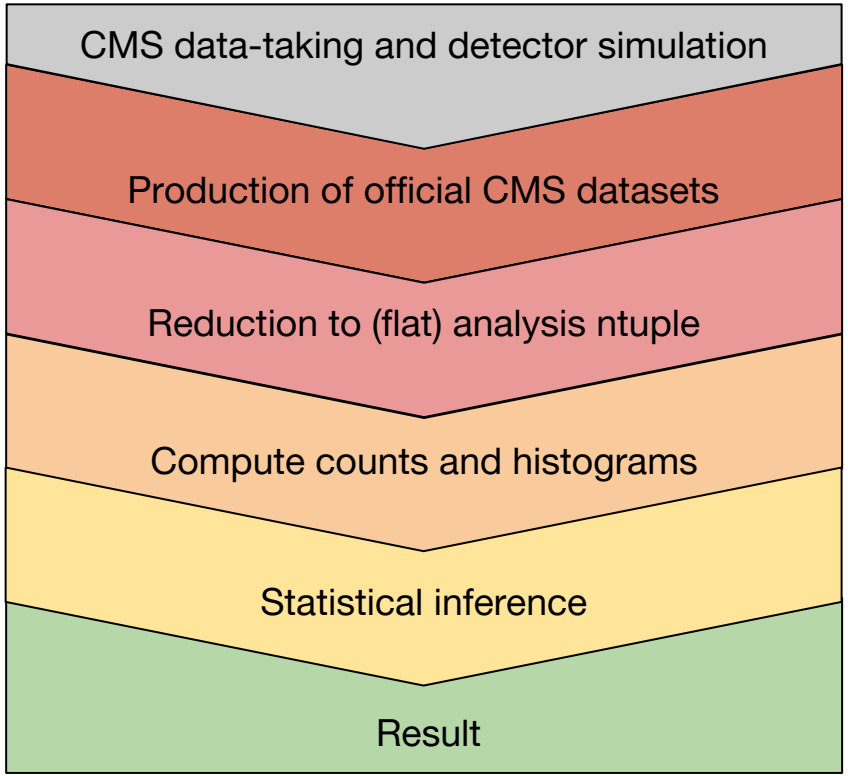
CMSSW

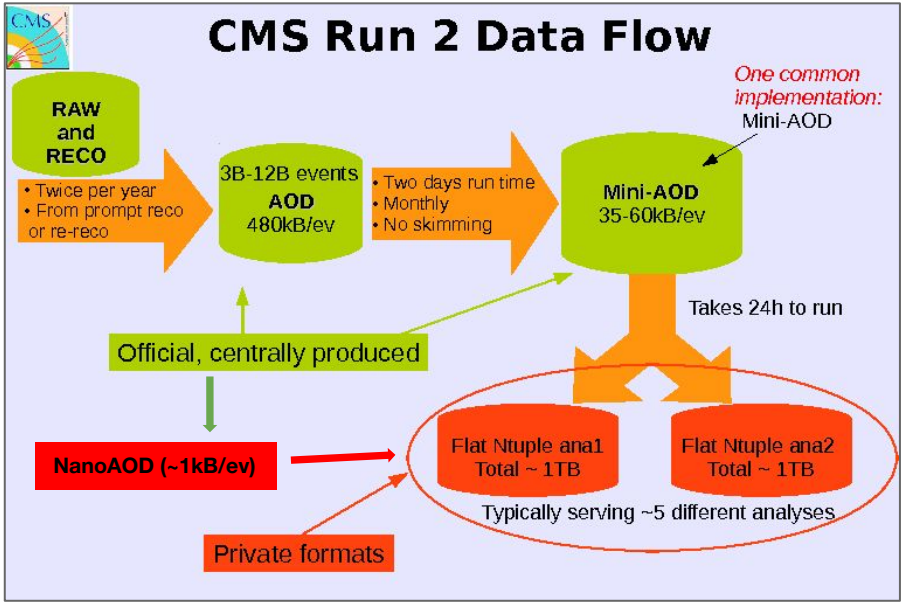
- (Mainly) C++ framework with configuration layer in Python
- Data storage: ROOT files
- Event data model: Collections of (linked) C++ objects
- Data processing: Plug-in modules managed by the framework, operating on the event data



https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSSWFramework

Data flow





Modified https://cds.cern.ch/record/2029414/?ln=de

Development in CMSSW

Advantages:

- Fast (C++ source code)
- Configurable and flexible (python config files)
- Robust framework for event products, multithreading, factory patterns, etc.

Disadvantages:

- Steep learning curve (6 million lines of code)
- Large overhead (100s of internal packages and external libraries)
- Requirements have become stricter over time: (pull request reviews)
  Thread-unsafe spaghetti code no longer allowed in central repo

Upshots:

- Many people use legacy code from Run 1 or early Run 2
  (experts leave the field, code maintenance is limited)
- Some use FWLite to interact with CMSSW objects in Python
  (easier to code, but slow and limited)
- “Recipes” often distributed as private branches to be merged and compiled

Custom Jet Reconstruction

- Common user-level reconstruction (substructure, boosted objects, pileup rejection)
- Various interfaces to the fastjet and fastjet-contrib packages
  - CMSSW
  - Bare C++ / ROOT macros
  - NEW: pyroot (fastjet only, no fastjet-contrib)
- $O(N^2)$ or $O(N \log(N))$ operations, N = number of inputs
  - When used with computation of jet areas, N can be large
  - Can be $O(N^3)$ or higher for some algorithms
- Often runs with several configurations

Individual frameworks and user code

Requirements: SUSY / EXO / B2G / HIG

Mainly Searches:

- Special case analysis needs custom reconstruction:
  central MiniAOD/NanoAOD samples cannot be used, huge computational effort
- Does not correspond to “standard” event hypothesis
  - Collections from larger data formats (e.g. AOD): long-lived particles (nonstandard tracking + vertexing), stopped hadrons in the HCAL (needs information outside of current bunch crossing)
    - May not be hosted as well as smaller datasets, large I/O burden
  - Novel algorithms: custom jet reconstruction, custom flavor tagging, machine learning
    - May be computationally intensive, grid jobs take longer
  - MiniAOD/NanoAOD event content has to balance common needs with special needs
- Usually less stringent systematic uncertainty treatment
- Needs clean and simple interface to statistical fitting tools

Requirements: TOP / SMP / HIG

- Measurements:
  - Unfolding is important
  - Needs robust systematic uncertainty treatment
  - Theoretical uncertainties required at user level

- SMP:
  - Extensive use of prescaled triggers
  - Reaching percent level precision
- Top:
  - Integrated with gen-level definition of top quark reconstructions (Rivet interface)
- HIG:
  - Still statistical limitations from dataset size

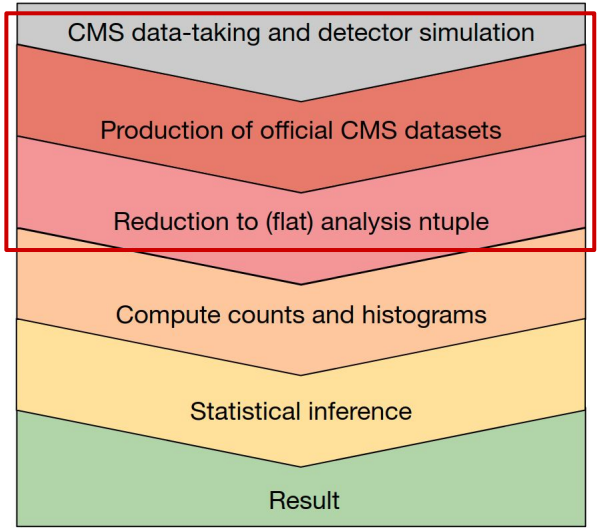
CMS User Survey

- Conducted in January 2017 (full details attached separately)
- Selected conclusions (based on 2016 dataset):
  - ~40 different analysis frameworks maintained privately by CMS users
  - ~300 weeks/yr spent maintaining frameworks (equiv. to 6 people full-time)
  - Collaboration processes full dataset (data + MC) ~1000 times/yr
  - Per processing run: ~1 week (some manual work for jobs to converge), O(10K)+ jobs, ~5 TB total output
  - Majority use CRAB (CMS Remote Analysis Builder) to access the grid

➔ Full Run2 dataset is 4× size of 2016, but CPU and disk resources have not increased by 4× at the same time...

Processing of official datasets

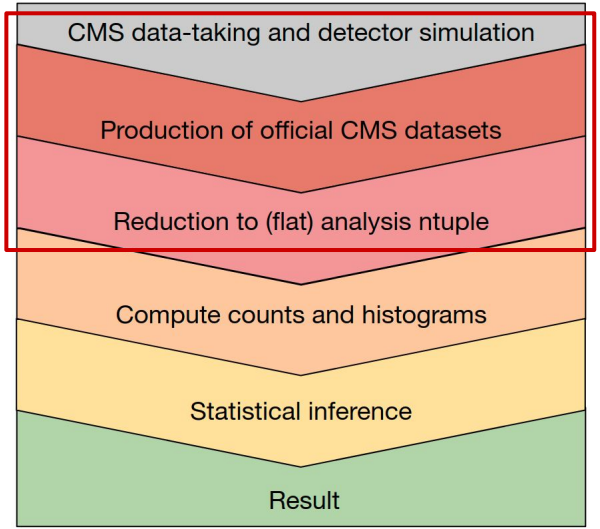
- Tasks:
  - Apply custom reconstruction steps and corrections
  - Apply official recipes
  - Reduce generic samples to (flat) analysis ntuple



- Common software solution:
  - CMSSW is required to read objects and implement custom reconstruction and corrections
  - POG developments and general utilities published through CMSSW
- Scaling:
  - Very good scaling with batch jobs, events are fully independent, well established workflows (but inefficient from processor cache standpoint)

https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideEDMNtuples

Processing of official datasets (2)

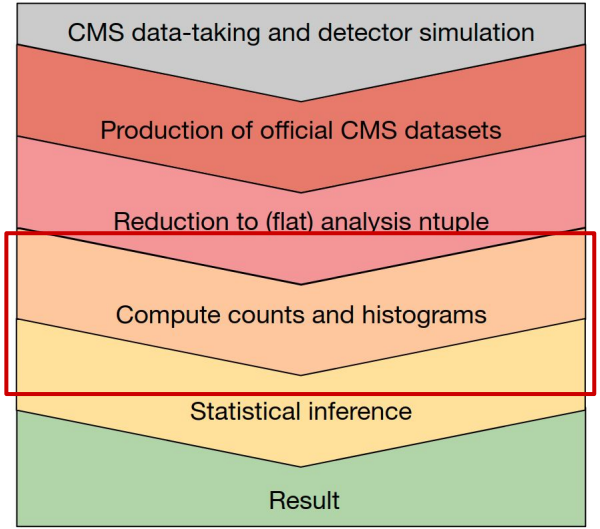


- Case study: TreeMaker framework
- First (?) multi-threaded ntuple production framework
- Central ntuples hosted at FNAL, shared by ~6 analyses in SUS & EXO that use primarily jets; code shared by several more
- ~1 week to process 1 year of data and MC (using CMS Connect to access grid via HTCondor), O(10) evts/sec/core (custom jet algos, taggers, etc.)
- Jobs run on 4 cores, use <2GB memory
- Event sizes: 1-2kb/evt for data, 5-10kb/evt for MC (gen particles and other info take up space) → full Run2 dataset takes >60TB
- Post-ntuple production, skimming step (ROOT-only private code) produces smaller trees with selection cuts applied, reduced number of variables

https://github.com/TreeMaker/TreeMaker

Variety of end-analysis workflows

- Tasks:
  - Going from analysis ntuple to histograms, counts or event list to be fed into statistical analysis and plotting
  - Apply additional reconstruction steps and corrections on reduced dataset, compute systematic shifts

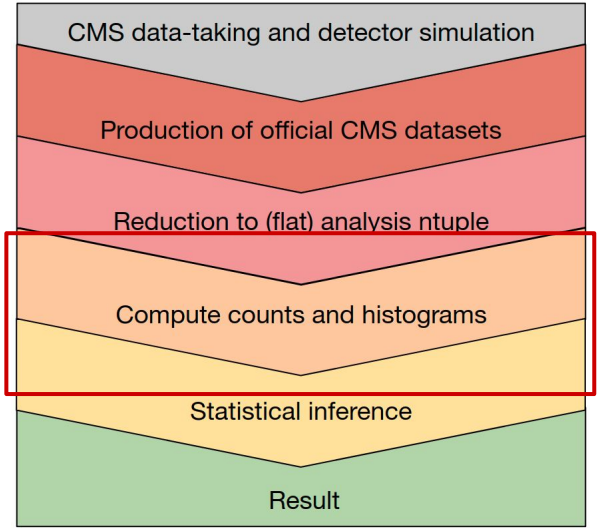


- Variety of software solutions:
  - Custom CMSSW module
  - Stand-alone C++ ROOT scripts/programs or Python/PyROOT scripts
  - `uproot`/`root_numpy`/`root_pandas` and Scipy software stack

https://www.scipy.org/stackspec.html

Variety of end-analysis workflows (2)

- Scaling:
  - Needs to run on large parts of analysis ntuple
  - Batch jobs parallelized per histogram/count/file
  - Run locally exploiting many-core machines
  - Depends strongly on specific analysis



- Challenges:
  - Often runs on a daily basis→
    99% efficiency and “tails” of a batch system slows down turn-around cycle
  - Local processing:
    - Reliable but much less computing power
    - Exploiting many-core machines is key
    - Optimal steering of event loop is non-trivial (produce all needed histograms with a single event loop)

Missing requirements ...

- How to distribute information that everyone needs (metadata?)
  - Cross sections, pileup weights, % negative weights, scale factors…
  - Not contained in event data format
  - Leads to repeated work, inconsistencies
- NanoAOD has centralized post-processing tools:
  - Python-based tools
  - POG-approved recipes
  - Customizable
  - Interface to standard CRAB tools
- Speeds up significantly doing an analysis from scratch

https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookNanoAOD
https://github.com/cms-nanoAOD/nanoAOD-tools

Statistical analysis

Systematic uncertainties

Typical systematic uncertainties:

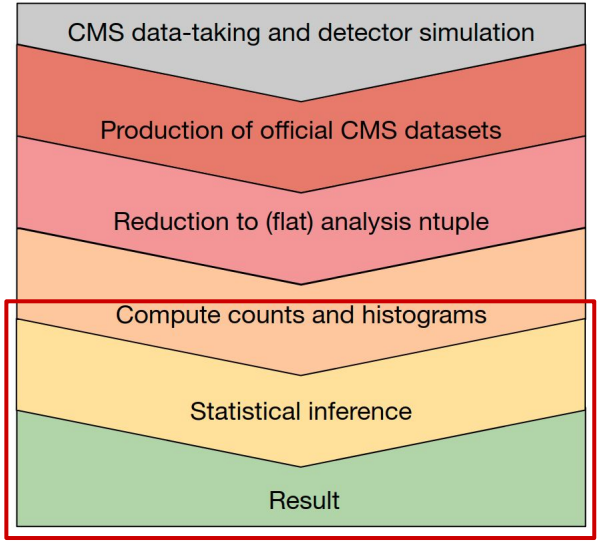
- Jets:
  - Kinematic uncertainties on p_T, angle, and mass
  - Searches: Usually jet energy scale (JES) and resolution (JER) uncertainties, mass scale (JMS) and resolution (JMS) for boosted topologies + substructure
  - Measurements: JES uncertainty is broken into O(30) uncorrelated sources. Duplicates data storage for each (can actually be large)
- Missing p_T: Propagates JES/JER uncertainties from jets
- Muons: Trigger, identification efficiency uncertainties and momentum uncertainties, O(250) variations
- Electrons/Photons: Efficiencies and energy calibration
- Taus: Efficiencies, energy calibration, additional uncertainties for high-p_T objects, uncertainty for fakable objects

Challenges:

- Systematic variations may need copy of analysis ntuple with altered property to propagate the effect
- Requires many counts or histograms, many runs over analysis ntuple
- High-dimensional fit with O(1000) parameters
- May require to process additional datasets

Statistical analysis

- Tasks:
  - Condensed information fitted with statistical model
  - Many high-dimensional fits



- Software solutions:
  - Plain RooStats/RooFit
  - RooStats/RooFit wrapper with implemented statistical models (combine, theta, …)
  - Computational graph frameworks (TensorFlow, …)
  - Scipy software stack (Numpy linalg and minimizer, ...)

- Scaling:
  - See next slide

Accelerating statistical analysis

- Statistical analysis runs often locally, can become a bottleneck for complex analyses
  →Workflow speeds up significantly with local acceleration
  - Multi-threading
  - Vectorization
  - GPU acceleration
- Batch jobs used for example for
  - Sampling toys from statistical model (limit computation)
  - Many independent fits (MSSM: Grid search for exclusion plots)
- Fitting would greatly benefit from analytic differentiation
  - No finite difference approximations of derivatives
    → Fewer likelihood evaluations
  - Accurate derivative → Fewer steps to convergence
  - Modern numerical computation libraries (TensorFlow, …) already provide most functionality including support for heterogeneous computing

https://indico.cern.ch/event/686641/contributions/2867259/attachments/1608597/2553481/tfa.pdf
https://indico.cern.ch/event/750215/contributions/3105052/attachments/1701062/2739491/tffit-Aug15-2018.pdf

Higgs combination tool

- Wrapper around RooFit/RooStats
- Builds statistical model for Higgs analyses
- Provides fitting procedures (profile likelihood, maximum likelihood fit, …)
- Common choice of PDF available for nuisances
- Defines format to serialize statistical model and results (likelihood, counts/templates, …) for later refitting and combination.
- Toy generation allows for less statistical assumption (e.g. Asymptotic approx.)
- Some studies that extensively use toys (Full frequentist constructions) are still prohibitive.

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideHiggsAnalysisCombinedLimit
https://github.com/cms-analysis/HiggsAnalysis-CombinedLimit

Unfolding tools

TUnfold:

- Powerful but unintuitive interface
- Arbitrary number of event categories, but cumbersome interface
- Usually run locally, SVD is fairly fast
- Active development
- Treatment of systematics can be cumbersome (requires N copies of large matrices)

RooUnfold:

- Very intuitive interface
- Easy to do 2D unfolding
- No longer recommended
  - Incorrect chi-squared treatment
  - Few levels of regularization control
  - Support is limited

Higgs Combination tool: limited documentation and accessory tools. Access to the full likelihood.

Python ecosystem (e.g. scipy.linalg.sparse):

- Less geared toward HEP, but fast, simple, highly optimized, and robust
- Interfaces for multiple event categories are cumbersome
- Should be watched, because development is very active

Big data tools

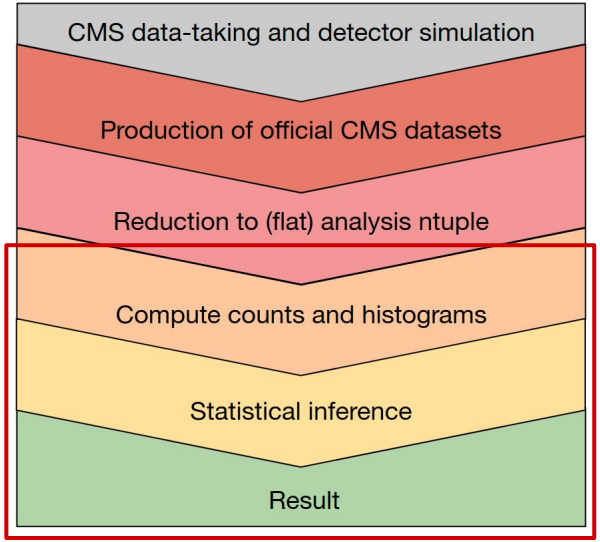
Scipy and machine learning

Tools that allow to process analyses ntuple efficiently:

- ROOT↔numpy: `uproot`, `root_numpy`, `root_pandas`
- Dataset manipulation: dask, pandas, numpy
- Histograms/plotting: seaborn, matplotlib

Huge impact:

- State of the art machine learning tools force analysts to move to numpy/Python
- Experience with Scipy stack and popular ML tools sell well on the job market
- Can resolve scaling issues (CPU usage, data replication)
- More in the next jamboree

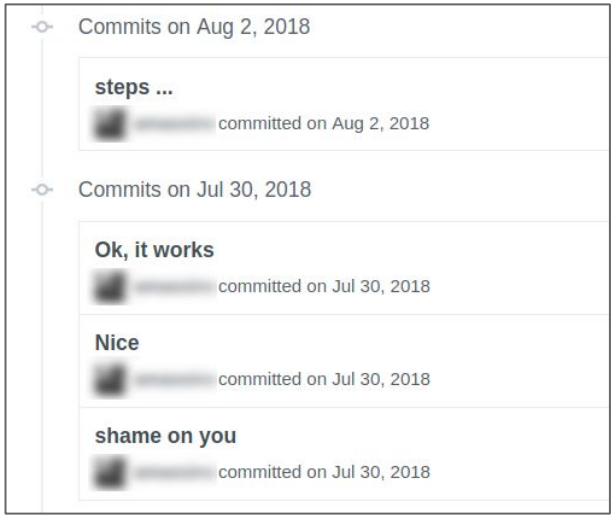


Reproducibility

Reproducible software

- Code management:
  - Central software (CMSSW) well maintained
  - Code versioning widely adopted for user code
  - User code often never intended to go public after publication

- Reproducible software:
  - CVMFS software stacks help a lot to document the used software
  - Cutting-edge machine learning packages often not included / GPU support not provided, complicated freezing of random initialization
  - Complex software environment, also need to connect to conditions database (etc.)



Reproducible analyses

- Common analysis documentation:
  - Analysis notes: Non-public technical documentation of the used techniques and expected physics results
  - Physics analysis summary / journal paper: Official physics documentation, technical details normally removed or condensed to a single sentence

- Challenges:
  - Analyses only reproducible from ANs with huge effort including reimplementation of most analysis steps
  - Each new analysis strategy needs to compare to the previous analysis
    → Impossible under changed conditions without rerunning their code
  - No recommendation / defined workflow to archive analysis software →
    Much effort for individuals to figure it out

Analysis legacy

What should be published for the long term reproducibility?

- Results including bestfit and covariance matrices (and syst. breakdown)
  Will allow for reinterpretation of the results
- Studies to have approximations of the likelihood function are ongoing

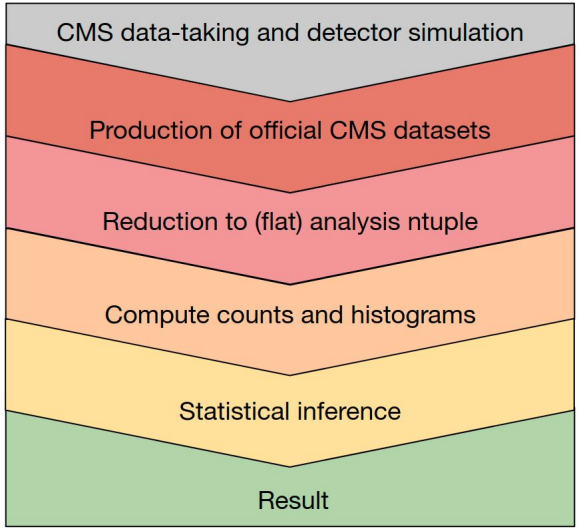
How do we preserve analysis code? (RIVET, HEPDATA,OpenData...)

- If reproducible software is achieved, this can be trivially managed
- Ideal world: user creates an image (e.g. Docker) with their WF steps, that image becomes the public-facing documentation.
- Real world: multitude of workflows exist, but this can be mitigated if there is a major switch to NANOAOD
- Work has begun in CMS to do this
- Ongoing effort to standardize software citations in HEP (CHEP2018)

Summary

- Many similar analysis workflows in CMS
- Data reduction step is most intensive for users:
  - Starts from central software and datasets
  - Produces flat ntuples
  - Used to create outputs for statistical inference
- All steps have limitations:
  - Need to speed up (parallelization, optimization)
  - Need to reduce dataset sizes and duplication
- New technologies being explored

➔ Need more work to be ready to analyze HL-LHC data



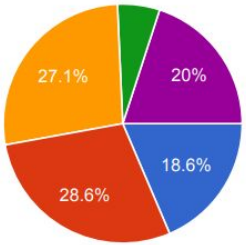
Backup

CMS User Survey: Data Usage

Source of your analysis (before any further reduction/processing... as extracted from DAS)

MiniAOD	**69**	74.2%
AOD	**27**	29%
RECO	**11**	11.8%

What is the size of your full processed sample (data and MC)?

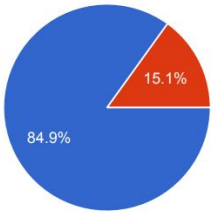


less than 100GB	**13**	18.6%
between 100GB and 1TB	**20**	28.6%
between 1TB and 5TB	**19**	27.1%
between 5TB and 10TB	**4**	5.7%
more than 10TB	**14**	20%

From CMS user analysis tool survey

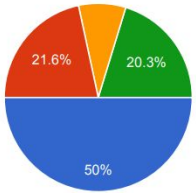
CMS User Survey: Processing datasets

Do you use a personal/group framework and produce private root trees?



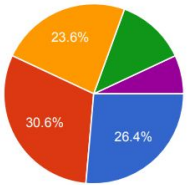
Yes	**79**	84.9%
No	**14**	15.1%

How often do you process the official samples and produce analysis ntuples?



once a month	**37**	50%
twice a month or more	**16**	21.6%
once a week	**6**	8.1%
Other	**15**	20.3%

How long does it take to produce the analysis ntuples?



1-2 day	**19**	26.4%
2 -4 days	**22**	30.6%
4-7 days	**17**	23.6%
8-14 days	**9**	12.5%
more than 2 weeeks	**5**	6.9%

From CMS user analysis tool survey