UNIVERSITÀ
DEGLI STUDI
FIRENZE

Data Analysis in CMS

Adinda De Wit, Clemens Lange, Huilin Qu, Keith Ulmer, Lindsey Gray, Mario Masciovecchio, Piergiulio Lenzi , Pieter David, Sezen Sekmen, Sebastien Wertz, Andrea Rizzi, Florencia Canelli, James Letts, Danilo Piparo, Mariarosaria D'Alfonso, Loukas Gouskos

ROOT Workshop - May 11th, 2022

What is/isn’t covered

- I will discuss **only the final data analysis steps**, i.e. not the simulation/reconstruction of the raw data
- I will cover:
  - Event content for data analysis
  - Data analysis and ideas for the future from both the code and computing perspective
    - Focusing on handling of systematics and event looping
- Most of what I am going to discuss is covered in a **task force effort** that was instituted in September 2021 to identify analysis best practices and make recommendations for the future. The TF is close to the end of its mandate

Bridging the gap between reco and statistical inference

- **An evolving landscape**
- Physics Object Groups (POGs) **producing data/MC corrections and their uncertainties to apply to simulation to bring it in agreement with data**
- A **CrossPOG group** established since long time **to develop and support the NanoAOD data tier** and to **coordinate**, among other things, the **production and dissemination of the corrections/uncertainties**
- Typical analysis approaches:
  - In Run 2 so far:
    - Corrections and uncertainties provided with **generally non-uniform format** by the POGs
    - **Post-processing of NanoAOD** to store analysis specific (esp. systematics information)
    - **Several frameworks**, some general purpose ones, all sharing significant effort in the precise handling of systematics
  - Now and in the future:
    - **Uniformation of the format for SFs** and centrally supported library (correctionlib) for their application
    - Agreed need between coffea and RDF **for low-level support for basic mechanics of handling systematics** (i.e. making variations as computationally cheap as possible)
    - Move towards a all-in-one-go approach from NanoAOD (not mandatory, but effort to make it possible)

The NanoAOD data format

- **General purpose flat TTree**
- Branch layout:
  - n<Object>, <Object>_[pt, eta, ….]
  - With cross referencing indices between collections
- Self documented
- Small event size, quantities stored with configurable precision
- NanoAODs contain calibrated objects, while allowing for re-calibration
- They **do not contain systematic variations**, except few exceptions that are analysis independent (most systematics are analysis-dependent)
- Relatively fast reprocessing
- Centrally produced NanoAODs cover the majority of analysis use cases. Outliers include those requiring “large” event data, e.g. tracks. **NanoAODs with customized event content are possible**
- Code is maintained and reviewed centrally by the CrossPOG.

Legacy (i.e. Run2) analysis workflows

- Typical workflow involved postprocessing code to add branches to NanoAOD, containing corrections and their uncertainties, as well as analysis specific information
- The **code for post-processing was, to some extent, maintained centrally,** but still on best effort basis only
  - **Mainly based on python event loop** (drawback: slow) with pyROOT, with convenient abstractions for physics objects and collections.
- Possibility to save entire copies of NanoAOD **or just the varied/added branches** exploiting friend trees when reading back.
- Different groups developed different, mostly provate, analysis frameworks came to go from post-processed NanoAODs to the input for the statistical inference, coming up with different ways of dealing with the analysis complexity and the corresponding bookkeeping
  - Most effort going into systematics changing the event interpretation (e.g. those changing pTs…). **For those, we swap inputs and rerun everything**

Latest trends in Run2 workflows and the road ahead

- Formalized schema for POG-provided corrections, and corresponding application code (correctionlib).
- Tools emerged, based on both RDF and Coffea, with **convergent evolution** towards:
  - **Python**
  - **Removing the need for the NanoAOD postprocessing step**
  - **Building “abstractions” of the physics objects** when reading the NanoAOD (i.e. putting together all the relevant branches to represent the object in memory)
    - Allow to write the analysis in a simple and concise way. In some cases this goes as far as to define a new “analysis language” with its own formal grammar, in other cases it leverages python
  - Exploiting new **distributed computing infrastructures**, leveraging Dask.
  - (Sometimes) Leveraging **workflow orchestration tools**.

Application of corrections and systematic uncertainties in analysis with RDF/Coffea

- Both frameworks have recent additions **which allow handling of systematics**
  - Both very generally allow users to add variations and create physics objects with varied features
- In RDF land, earlier CMS specific implementations to handle the the branching of the DAG existed in some tools, notably Bamboo and NAIL
- Final implementations and user interfaces still out for questions and trial by fire
  - Very likely the case that framework made atop RDF/coffea will design smooth interfaces for users
  - Objective of coffea/RDF to capture and make easily efficient core functionality of systematic variations
    - “How to vary something”, “what variations are there, how many sigma do they represent”
    - “Are these variations correlated with other variations?”

Example of an RDF based framework: Bamboo

- Goal: **find a way to write analysis that is easy, modifiable, shareable, fast**.
- RDF reduces boilerplate but writing a full analysis including systematics **can still be a lot of work**
- Idea: decorate tree
  a. Provide a **view of the event content as a set of collections of physics objects**
  b. Allow the user to **write expressions with these objects in python**. Systematics are handled seamlessly: the same python object acts as both the nominal and the varied in the expression, and is handled with branching of the DAG behind the scenes
  c. When done, **build RDataFrame behind the scenes and run**.
- Tree decoration d**epends on the tree format**. NanoAOD is supported, but can be adapted also to other formats

```
from bamboo.plots import Plot, EquidistantBinning as EqBin
from bamboo import treefunctions as op

def definePlots(self, t, noSel, sample=None, sampleCfg=None):
    plots = []

    muons = op.select(t.Muon, lambda mu:
        op.AND(mu.pt > 30., op.abs(mu.eta) < 2.4))

    muSel = noSel.refine("1mu", cut=(op.rng_len(muons) == 1))

    plots.append(Plot.make1D("mu_pt", muons[0].pt, muSel,
        EqBin(100, 30., 130.), title="Muon pt"))

    jets = op.select(t.Jet, lambda j: op.AND(j.pt > 30., os.abs(j.eta) < 2.4))

    mu4JetSel = muSel.refine("1mu_4j", cut=(op.rng_len(jets) >= 4))

    plots.append(Plot.make1D("jet_pt", op.map(jets, lambda j: j.pt),
        mu4JetSel, EqBin(100, 30., 130.), title="All jets pt"))

    return plots
```

```
from bamboo.scalefactors import get_correction

elIDSF = get_correction("EGM_POG_SF_UL.json", "UL-Electron-ID-SF",
    params={ "pt": lambda el: el.pt, "eta": lambda el: el.eta,
             "year": "2018UL", "WorkingPoint": "Loose" },
    systParam="ValType", systNomName="sf",
    systName="elID", systVariations=("sfup", "sfdown"))
    # resulting variations in bamboo: elIDup, elIDdown

looseEl = op.select(tree.Electron, lambda el: el.looseId)

withDiEl = noSel.refine("withDiEl",
    cut=(op.rng_len(looseEl) >= 2),
    weight=[ elIDSF(looseEl[0]), elIDSF(looseEl[1]) ])
```

Computing resources dedicated to analysis

- AFs are emerging in several CMS institutes
- Mainly focused on allowing **modern interactive or quasi-interactive workflows** on large volumes of data
  - The paradigm is moving from “shoot jobs and wait (and hope)” to “run an interactive master job that, behind the scenes, finds the resources, splits, runs and collects the output seamlessly”
- **Convergent evolution here as well**!
  - Mostly based on JupyterHub
  - Mostly leveraging DASK
- Well advanced prototypes:
  - Coffea-casa at Nebraska, INFN-AF at CNAF, ElasticAF at FNAL
- Work in progress at CIEMAT, MIT, Purdue, and maybe elsewhere
- See also the talk by L. Gray at the WLCG meeting and the recent related HSF forum kickoff

Concluding remarks

- Data analysis tools are evolving in CMS
- Rich set of tools emerging in the collaboration on top of RDF and Coffea
- Effort is generally towards finding simpler, more expressive and compact ways of writing an analysis
- Convergent evolution towards:
  - Starting from NanoAODs
  - With expressive code
  - With the opportunity to process the NanoAODs directly
- Interest towards new computing infrastructures
- The landscape is evolving, no clear one-fit-all solution, but many out for trial by fire

Backup