Thoughts on storing and manipulating Covariance Info in Rivet

Louie Corpe (UCL)

LHC-EW WG: Jets and EW bosons Meeting

04 May 2018

Introduction

- Comparing generator performance typically uses **data/MC ratio in a set of observables**... But as ∫(Lumi) grows, **simple ratio no longer suitable** since (correlated) systematic uncertainties > stat uncertainties

- Start looking into how to **evaluate GoF in presence correlated systematic uncertainties**. What tools are needed to **manipulate covariance info**/correctly evaluate **generator agreement with data** ?

- Used **augmented YODA format** and **standalone functions**. Work ongoing with Rivet devs to **back-propagate this work into Rivet/YODA**

- Disclaimer: this talk details some of the methods, and suggestions arising from my discussions with ATLAS, Rivet developers. I am not speaking for ATLAS collaboration as a whole or for Rivet developers!

Scope and relevant questions

- One possible workflow, which I've used so far:
    - Access HEPData record for a given RIVET analysis. Most importantly, needs full breakdown of experimental uncertainties for each observable. (***How should we be storing covariance info on HEPData?***)
    - Mark-up Data YODA file with variation in each bin for each named uncertainty source. Currently using a YAML-format annotation in text-based YODA files. (***How should we be storing covariance info in YODA files?***)
    - Built covariance matrix from the uncertainty breakdown (2 methods possible) (***How to get Cov matrix from Error breakdown?***)
    - If MC prediction comes with Theory uncertainties, do the same and add covariance matrices together. Use covariance matrix to evaluate GoF using the covariance info. (***What is appropriate GoF measure?***)

Storing Cov info on HEPdata

HEPData submission format is not currently set up for storing covariance info. Also no agreed format… here are my two cents.

- **Best-case scenario:** Errors split by individual syst contribution per bin
  Pros: can reconstruct Cov matrix + **correlate w/ other measurements**
  Cons: assumes errors fully correlated across/between distributions and up/down variation defined consistently

- **Second-best scenario:** Exact cov matrix provided directly by analysts
  Pros: Cov matrix directly in hand
  Cons: Not always possible to correlate w/ other measurements…
  Probably need to do this anyway for statistical correlations..

- **To be avoided**: Errors split by **aggregated source** only(eg stat & syst)
  Pros: Better than ignoring correlations completely…
  Cons: Cov matrix can only be roughly approximated…
  We should really be doing better than this

Storing covariance info in YODA/Rivet

- **The workaround I have adopted for now:** store detailed uncertainty breakdown as **YODA annotations in YAML format**: trivial to store and read, and can handle complex structures.
- But this is a fairly crude approach... but works out of the box.

```
BEGIN YODA_SCATTER2D /EXAMPLE
Corr={0: {stat: {dn: -1., up: 1.}, eff: {dn: -0.1, up: 0.1}}, 1: {stat: {dn: -1.73205,
Path=/EXAMPLE
Title=
Type=Scatter2D
# xval      xerr-       xerr+       yval        yerr-       yerr+
1.00000e+00  0.500000e+00  0.500000e+00  1.00000e+00  1.00000e+00  1.00000e+00
2.00000e+00  0.500000e+00  0.500000e+00  3.00000e+00  1.73205e+00  1.73205e+00
3.00000e+00  0.500000e+00  0.500000e+00  9.00000e+00  3.00000e+00  3.00000e+00
END YODA_SCATTER2D
```

```
import yoda, yaml
hists = yoda.read('/afs/cern.ch/user/l/lcorpe/work
h=hists['/ATLAS_2017_I1514251/d01-x06-y01']
corr= h.annotation('Corr')
a = yaml.load(corr)
a.keys()
--> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
a[0].keys()
--> ['alphas', 'stat', 'scale', 'norm', 'pdf']
a[0]['stat']
--> {'dn': -0.01772649, 'up': 0.01772649}
```

- **Better:** modify the YODA "Point" class. Currently has 1 central value and errors (up,down). Replace errors with vector of (label, up,down) pairs to store arbitrary number of variations.
- Rivet workshop this month - watch this space

Producing a Cov Matrix from uncertainty breakdown

- **Two methods to produce Cov matrix** from uncert breakdown:
  - **Direct propagation** of errors (outer product for correlated, diag^2 for uncorrelated). E.g in Professor 1.4.0, used for tuning
    - **Fast, easy to implement**, but **cannot handle asymm errors**
  - **From pseudo-experiments (toys)**. E.g as done in top group to evaluate goodness of fit
    - **Slow, depends on nToys** but **handles asymmetric errors**
- Cov matrices reconstructed from (sufficiently) granular uncertainty breakdown likely to be **sufficient for most use cases**
- Open question: **how to handle local correlations** (eg neighbouring bins, but not across the whole distribution)?
- Both options can be propagated into YODA

Goodness of Fit options

- For now, use **a simple χ2** which accounts for the cov matrices:

$$\chi^2 = (\vec{x} - \vec{\mu})^T C^{-1} (\vec{x} - \vec{\mu}) \tag{4}$$

where $\vec{x}$ is the vector of data, $\vec{\mu}$ the interpolated MC prediction and C the covariance matrix of the data.

- In future, more complex goodness of fit measures may be explored.
  - eg Nuisance parameter representation? as used by HERAFitter

$$\chi^2(m,b) = \sum_i \frac{\left[\mu_i - m_i\left(1 - \sum_j \gamma_j^i b_j\right)\right]^2}{\delta_{i,\mathrm{unc}}^2 m_i^2 + \delta_{i,\mathrm{stat}}^2 \mu_i m_i \left(1 - \sum_j \gamma_j^i b_j\right)} + \sum_j b_j^2,$$

https://arxiv.org/pdf/1410.4412.pdf (eq 20)

Summary

- Including covariance info into GoF calculations is becoming increasingly critical in evaluating generator performance.
- What's needed?
- Tools to include cov info in YODA files
- Need for a common HEPData convention.
    - My suggestion is a breakdown of uncertainty
    - The cov matrix can then be constructed from that
- Some tools may be back-propagated into YODA/Rivet
    - Work ongoing
- Watch this space

[BACKUP]