Proposal for recommendation to store correlated uncertainties

Louie Corpe (UCL)

LHC EW WG : Jets and Bosons meeting, 18.02.19

Introduction

- Recently (well, december...) gave a talk at the LHCEWWG workshop on topic of storing bin-bin correlations
  - Two main points:
    - YODA now has a format to store uncertainty breakdowns
    - HEPData now has a mechanism to propagate the uncertainty breakdown into YODA
  - With this, we can rebuild the systematic covariance matrix.
- It was suggested to formulate recommendations in a public document, to set a 'standard' on how to store covariance info for analyses in HEPData
- This talk is intended to prompt discussion about what the document should look like. Two options possible, as far as I can see. I'll discuss each briefly and open a discussion.

Reminder of key points of talk at LHCEWWG workshop

- Bin-bin correlation info increasingly important as LHC gathers more data.
- For systematics bin-bin corrs, it is enough to provide the breakdown of uncertainties in each bin: reconstruct matrix with either direct or toys method.
- HEPdata.net already has functionality to store uncertainty breakdown.
  - Thanks to recent efforts, YODA/Rivet has a format to handle this, so can be used directly for data/MC comparison.
- Provided some examples of the full workflow
- Provided a small library of functions to perform the main tasks in manipulating the covariance info/error breakdown
- Future work needed to also store correlation matrix directly in HEPData and YODA/Rivet

Proposed structure. Option A: longer reference document

Option A

Contents

1 Introduction **2**

2 Motivation: What improvements could be brought by having access to covariance information? **5**
- 2.1 Goodess of fit and MC tuning 5
  - 2.1.1 More accurate valiation of generator predictions 5
  - 2.1.2 Global GoF across measurements, and across experiments? 5
  - 2.1.3 MC generator tunin,g eg with Professor 5
- 2.2 BSM re-interpretation 5
  - 2.2.1 eg simplified likelihoods, CONTUR 5
  - 2.2.2 CONTUR 5

3 Survey of existing infrastructure **5**
- 3.1 HEPData 5
- 3.2 Rivet and Yoda 5
- 3.3 Interplay between HEPData and Rivet 5
- 3.4 RECAST 5
- 3.5 Others: Root, numpy 5

4 Recommendations and proposed workflow **5**
- 4.1 Possible formats to store covariance information 5
  - 4.1.1 As a covariance matrix 5
  - 4.1.2 As an uncertainty breakdown 5
- 4.2 Converting from uncertainty breakdown to covariance matrix 5
  - 4.2.1 Direct method 5
  - 4.2.2 Pseudo-experiments (toys) method 5
- 4.3 Pros and cons of each approach 5
- 4.4 Systematic uncertainties stored as error breakdowns on HEPData 5
  - 4.4.1 Proposed conventions for symmetric, asymmetric errors etc 5
- 4.5 Statistical uncertainties stored correlation matrices on HEPData 5
  - 4.5.1 In the future: storage of Bootstrap replicas so that we can correlate between measuremnts? 5
- 4.6 Seamless propagation of covariance infomation from HEPData to Rivet/YODA 5
- 4.7 A common library of tools to manipulate YODA files with covariance information 5

5 Technical developments needed for the proposed worklow **5**
- 5.1 Recently completed 5
  - 5.1.1 Produce a common library of tools 5
  - 5.1.2 A format to store error breakdowns in YODA files 5
  - 5.1.3 Propagation of uncertainty breakdown from HEPData record to YODA 5
- 5.2 Still needed 5
  - 5.2.1 A format to store covariance matrices in YODA and HEPData 5
  - 5.2.2 Modifications of existsing HEPData records to match recommendations of this document 5

6 Examples of usage of covariance information **5**
- 6.1 To estimate goodness of fit between data and simulation 5
- 6.2 For re-interpretation of BSM search results 5

7 Outlook and next steps **5**

Appendix **6**

A Definitions of technical terms **6**
- A.1 Covariance matrix 6
- A.2 Correlation matrix 6
- A.3 Goodness of fit 6

B Pseudocode examples **6**
  - B.0.1 Error breakdown to covariance matrix : Direct method 6
  - B.0.2 Error breakdown to covariance matrix : Pseudo-experiments (toys) method 6

- motivate why storage of correlation information is important, survey what infrastructure currently exists and what is lacking, then to make the proposal itself, and follow up with an example.

Criticisms of option A

- A lot of the content of the proposed structure in option A is 'text-book' stats or already documented.
  - Motivation section covers well-known aspects, could just mention these in introduction?
  - options for storing the correlation information (cov matrices or breakdown), the methods for converting from one option to the other, the pros and cons for each approach are also 'not new' and already well known by many in the community. Perhaps just reference previous documents ?
  - Making public bootstrap replicas are presented as something to be done "in the future"... but I have later learnt this has already been done by some HEPData entries ! Perhaps we describe this as part of recommendation?
  - discussion about the possibility of correlating the JES between experiments: such correlations have already been evaluated between ATLAS and CMS, and used in the combination of top mass and cross section measurements since a few years

Criticisms of option A

Perhaps the document should therefore be made much lighter, and really just be a step-by-step practical guide, with link to useful references and examples?

It depends if we would like the document to be :

- a comprehensive reference on the topic of storing and using covariance info

  or

- a shorter technical document which gives detailed instruction so each step, without giving the details of the stats/formulas which underpin it

Proposed structure. Option B: shorter technical document

Option B

Contents

1 Introduction 2
1.1 Briefly decribe motivation for including cov info in HEPData, eg future tuning/re-interpreation 4

2 Recent/upcoming technical developments 4
2.1 A format to store error breakdowns in YODA files 4
2.2 A format to store covariance matrices in YODA and HEPData 4
2.3 A common library of tools to manipulate YODA files with covariance information 4
2.4 Seamless propagation of covariance infomation from HEPData to Rivet/YODA 4
2.5 Modifications of existsing HEPData records and Rivet REF data to match recommendations of this document 4

3 Recommendations and proposed workflow 4
3.1 Systematic uncertainties stored as error breakdowns on HEPData 4
3.1.1 Proposed conventions for symmetric, asymmetric errors etc 4
3.2 Statistical uncertainties stored correlation matrices on HEPData 4
3.2.1 Proposed conventions 4
3.2.2 Avanced example: Bootstrap replicas 4

4 Step-by-step examples 4
4.1 Uploading Error breakdowns to HEPData 4
4.2 Uploading covaraince matrices to HEPData 4
4.3 Propagating from HEPData to YODA 4
4.4 Estimating goodness of fit between data and simulation in Rivet 4
4.5 For re-interpretation of BSM search results eg CONTUR 4

5 Outlook and next steps 4

- Describe recent technical developments in the topic. Make recommendations for the workflow, and give some step by step examples.
- Do not explicitly discuss details of various methods to store correlations, formulas, etc…
  - But give an extensive list of references and resources for further reading.

Summary

- Before we do anything, we should decide what the aim of the document is?
  - A: To be a comprehensive reference of methods, formulas and technical details, with recommendations and examples?
  - B: Do not go into detail about the ‘textbook’ stuff, and just keep the document short and technical, giving recommendations and examples for how to upload information on HEPData
- Would A be useful or overkill? Stick to B for simplicity ? Other ideas?
- One important point is that it would be great to converge on the storage of covariance matrices in HEPData and Yoda before making the document public:
  - That way we can show examples for all aspects of the proposed workflow rather than just for some.
  - I’m discussing with Rivet/Yoda authors, but if more people want to joint he effort, it will get done faster!

[Discussion!]

Introduction

- **Data/MC comparison in presence of correlated systematic uncertainties is an increasingly important topic in HEP**:
  - Presently, comparisons typically made with **ratio of data/MC** in a set of observables (eg with Rivet)...
  - As LHC collects more data, and systematic uncertainties come to dominate analyses, **simple ratio plot may no longer be suitable** since **systematics may be correlated between bins.**
- **Similarly, for BSM re-interpretations, need to account for correlations between bins...**
- In both cases, **need access to systematic covariance matrix from measurements** (and for modelling/theory uncertainties for theory).

Introduction (ii)

- Moving forward we **need covariance matrices** to **compare data/MC** for measurements, and to **perform re-interpretations**... **what now?**

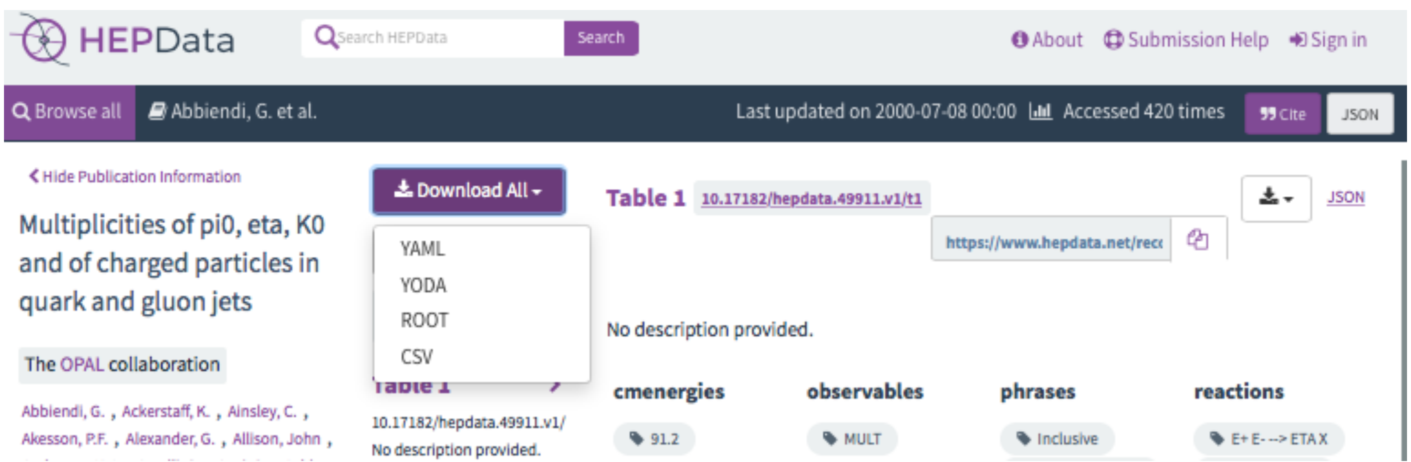
    1. **Where** should covariance information be **accessible from**?
    2. **How** should we **communicate covariance info**?
    3. **What technical developments** are needed?
    4. **How can we use the covariance info**?
    5. **Where do we go from here**?

- Collaboration with Rivet and HEPData authors to try to **define a workflow to allow a reliable access to covariance information**

Storing covariance info

1. **Where** should covariance information be accessible from?

- Given it's status as the main archive of experimental results, **HEPData.net seems like the obvious choice**.
  - Paper data **uploaded in custom YAML-based format**. Supports **N-dimensional histograms** and **unlimited number of uncertainty labels** for each point (symmetric or asymmetric)
  - Has **internal converter from YAML to ROOT, YODA and CSV**
  - All data are then available to download for Data/MC comparison or re-interpretation



Storing covariance info

1. **<u>Where</u>** should covariance information be accessible from?

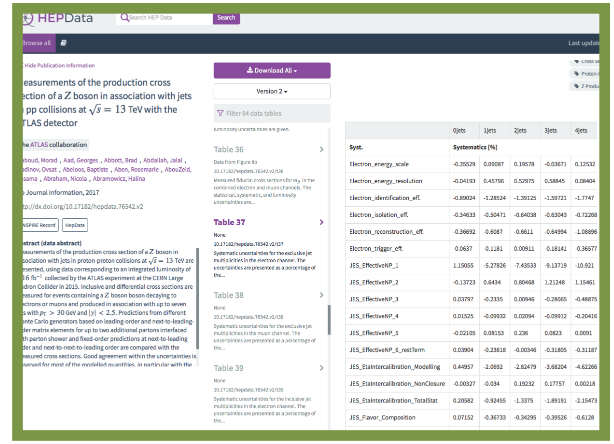
- **HEPData format is already very flexible**: indeed many analyses upload covariance information... BUT
  - until now, **no common convention on format for covariance**
  - **some info only been available in the YAML** format...
  - ...**rather than the widely-used YODA** (Rivet) format which is already the standard for Data/MC comparisons.
    - **YODA not designed with covariance matrices** in mind.

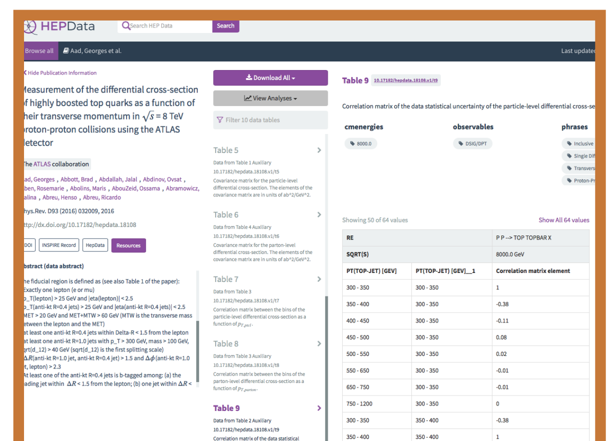
- The question then becomes:
  - What's the most useful way to store covariance info?
  - What needs to be changed in the YODA format to be able to store this info in a useable way?
  - How to convert the YAML-based info into YODA format when downloading from [HEPData.net]?

Communicating covariance info

2. **How** should we communicate covariance info?

- **There are two schools of thought…**
    - Each point on HEPData has **error breakdown** split by individual systematic contribution. **Cov matrix can be reconstructed and correlated with other measurements (one day, across experiments??)**
    - **Explicit covariance matrix** provided directly. **Not always possible to correlate w/ other measurements…**
- **The error breakdown** has emerged as the priority, although eventually we should be able to support both with HEPData/Yoda





Communicating covariance info

- **Two methods to produce a cov matrix** from error breakdown:

  - **Direct propagation** of errors (outer product for correlated, diagonal^2 for uncorrelated). E.g in Professor 1.4.0, used for tuning in ATL-PHYS-PUB-2015-007/
    - **Fast, easy to implement**, but cannot handle asymm errors

  - **From pseudo-experiments (toys)**. E.g as done in ATLAS top group to evaluate their goodness of fit.
    - Slow, depends on Ntoys but **can handle asymmetric errors**

- **Pseudo-code for both method are in the backup!**

Technical devs for covariance info

3. **What** technical developments are needed to store cov info?

- **For the error breakdown:**
  - HEPData YAML format **already has functionality to add extra labels for error breakdown** for each data point
  - **Need:** A format to **store the breakdown in YODA files**
  - **Need**: To modify HEPData internal converter to **propagate the uncertainty breakdown from YAML to new YODA** format
- **For exact covariance matrix**
  - HEPData YAML format **can store 2D histograms, which could be used to store a covariance matrix directly**
  - **Need:** a way to **link a particular bin in a distribution to a row in the covariance matrix** (both for YAML format and for YODA)
  - **This will be needed eventually (if nothing else, for stat correlations!) but we focus on error breakdown for now**

Technical devs for covariance info

3. **What** technical developments are needed to store cov info?

- **For the error breakdown:**
  - HEPData YAML format **already has functionality to add extra labels for error breakdown** for each data point
  - **Need:** A format to **store the breakdown in YODA files**
  - **Need**: To modify HEPData internal converter to **propagate the uncertainty breakdown from YAML to new YODA** format
- *For exact covariance matrix*
  - *HEPData YAML format **can store 2D histograms, which could be used to store a covariance matrix directly***
  - ***Need:** a way to **link a particular bin in a distribution to a row in the covariance matrix** (both for YAML format and for YODA)*
  - ***This will be needed eventually (***
    correlations!) but we focus on e

In discussion with YODA authors about best way to do this

Technical devs for covariance info

Need: A format to **store the breakdown in YODA files**

- LC / A. Buckley (Rivet/Yoda dev) implemented technical solution
- --> **store uncertainty breakdown of a YODA object as an annotation**. Upgraded YODA to be able to handle YAML code as annotations…. in this example, 2 sources of uncertainty (‘`stat`’ and ‘`eff`’)

```
BEGIN YODA_SCATTER2D /EXAMPLE
ErrorBreakdown={0: {stat: {dn: -1., up: 1.}, eff: {dn: -0.1, up: 0.1}}, 1: {stat: {dn: -1.73205
, up: 1.73205}, eff: {dn: -0.15, up: 0.15}}, 2: {stat: {dn: -3, up: 3}, eff: {dn: -0.09, up: 0.
09}}}
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
~
~
```

This is available NOW

NB: In future release, planning to store this as ‘additional columns’ for each error source: more readable!)

Technical devs for covariance info

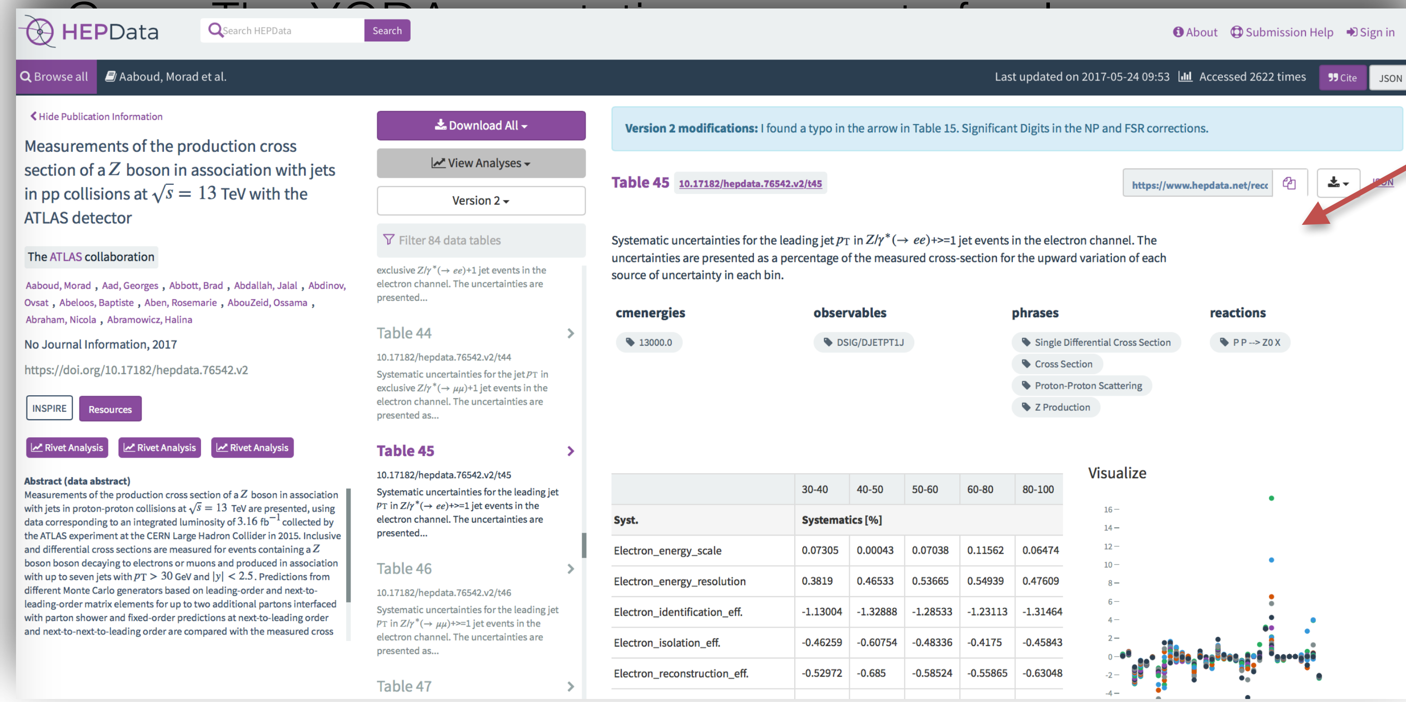
Need: To modify HEPData internal converter to **propagate the uncertainty breakdown from YAML to new YODA** format

- **Cons:** The YODA annotations are sort of ugly (but Scatter*D with extra columns will be nicer...)
- **Pros:**
  - This **sets a standard where previously there was none**.
  - Re-uses existing HEPData YAML upload format.
  - This means that historical HEPData entries where a breakdown was provided (even if just stat, sys, lumi) will be able to be downloaded as YODA with annotations...

- ...but **HEPData entries which used different conventions** (eg uploaded error breakdowns as additional tables) **need to be tweaked to be used in this way**.

Technical devs for covariance info

Need: To modify HEPData internal converter to **propagate the uncertainty breakdown from YAML to new YODA** format



Sub-optimal: separate table with uncertainties… for some other table..

We need to make sure HEPData Validators within the experiments are aware that they should upload their uncertainties using Labels, not separate tables

- …but **HEPData entries which used different conventions** (eg uploaded error breakdowns as additional tables) **need to be tweaked to be used in this way**.

Technical devs for covariance info

Need: To modify HEPData internal converter to **propagate the uncertainty breakdown from YAML to new YODA** format

Inspire ID / Rivet Name	AOs	Error Sources
ins1304688 on HEPData ATLAS_2014_I1304688	d07-x01-y01 d01-x01-y01 d06-x01-y01 d09-x01-y01 d04-x01-y01 d08-x01-y01 d02-x01-y01 d05-x01-y01 d03- ... (9 unique AOs)	stat sys,B_tagging sys,Backgrounds_Other sys,Color_Recon sys,Fragmentation sys,IFSR sys,JER sys,JES_ ... (41 unique sources)
-	-	-
ins1467454 on HEPData ATLAS_2016_I1467454_EL	d02-x01-y01 d03-x01-y01 d01-x01-y01 (3 unique AOs)	stat sys,cor1 sys,cor10 sys,cor11 sys,cor12 sys,cor13 sys,cor14 sys,cor15 sys,cor16 sys,cor17 sys,co ... (38 unique sources)
	d30-x01-y01 d36-x01-y01 d09-x01-y02 d35-x01-y01 d29-x01-y01 d31-x01-y01 d38-x01-y01 d34-x01-y01 d37- ... (14 unique AOs)	stat sys,IDres cor sys,MC unc sys,MSres cor sys,bgMC unc sys,diboson cor sys,iso cor sys,lumi sys,mu ... (14 unique sources)
	d12-x01-y01 d11-x01-y01 d10-x01-y01 (3 unique AOs)	stat sys,alpha_s sys,pdf sys,pi sys,scale
	d09-x01-y01 d12-x01-y02 d06-x01-y01 d13-x01-y01 d11-x01-y02 d17-x01-y01 d14-x01-y01 d05-x01-y01 d16- ... (14 unique AOs)	invalid ErrorBreakdown
	d22-x01-y01 d06-x01-y02 d21-x01-y01 d27-x01-y01 d05-x01-y02 d25-x01-y01 d24-x01-y01 d19-x01-y01 d23- ... (12 unique AOs)	stat sys,Eres cor sys,Escale cor sys,MC unc sys,bgMC unc sys,diboson cor sys,id cor sys,iso cor sys, ... (16 unique sources)
	d18-x01-y01 d04-x01-y02 (2 unique AOs)	stat sys,Eres cor sys,Escale cor sys,bgMC unc sys,diboson cor sys,id cor sys,iso cor sys,iso unc sys ... (15 unique sources)
-	-	-

I've produced a list of which HEPdata entries (those with a Rivet routine) actually have uncertainty breakdowns in the right format: Full list here https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library/blob/master/RivetAnalysesWithErrorBreakdown.md

- But **o(30) HEPData entries** already have the full error breakdown, with **O(100) more having at least stat/syst** (which may already be enough for many use cases ?)

4. **How can we use the covariance info**?

- **Data/MC comparison...**

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

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

ATL-PHYS-PUB-2015-007/

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

4. **How can we use the covariance info**?

- **Re-interpretation**
  - Simplified likelihoods

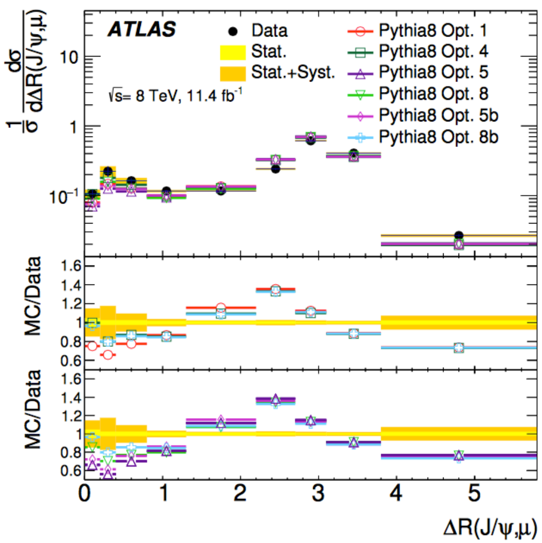
$$\mathcal{L}_S(\mu,\theta) = \prod_{i=1}^{N} \frac{(\mu \cdot s_i + b_i + \theta_i)^{n_i} e^{-(\mu \cdot s_i + b_i + \theta_i)}}{n_i!} \cdot \exp\left(-\frac{1}{2}\theta^T \mathbf{V}^{-1}\theta\right), \quad (4)$$

where **V** represents the covariance matrix for the expected background contributions across the search regions CMS-NOTE-2017-001 https://cds.cern.ch/record/2242860

  - Tools like CONTUR, which can set exclusions on BSM models based on library of SM measurements (could use covariance matrix to increase sensitivity wrt current 'most-sensitive-bin' approach) (See backup)

Example of full workflow

1. HEPData submissions using the error labels to store breakdown in each bin



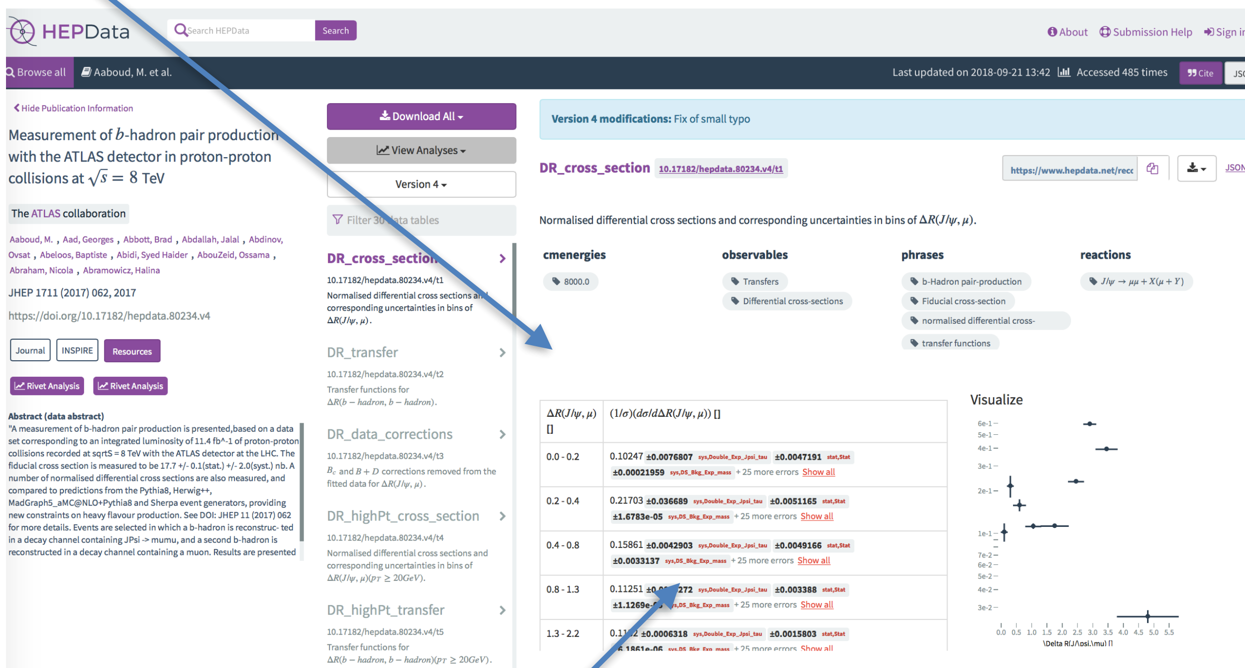
https://arxiv.org/pdf/1705.03374.pdf (Fig6)


```
dependent_variables:
- header:
    name: $(1/\sigma)(d\sigma/d\Delta R(J/\psi,\mu))$
    units: ''
  values:
  - errors:
    - label: sys,Double_Exp_Jpsi_tau
      symerror: 0.0076807
    - label: stat,Stat
      symerror: 0.0047191
    - label: sys,DS_Bkg_Exp_mass
      symerror: 0.00021959
    - label: sys,Trigger
      symerror: 4.4163e-05
    - label: sys,Bc
    (...)
    value: 0.10247
  - errors:
    - label: sys,Double_Exp_Jpsi_tau
      symerror: 0.036689
    - label: stat,Stat
      symerror: 0.0051165
    - label: sys,DS_Bkg_Exp_mass
      symerror: 1.6783e-05
    - label: sys,Trigger
      symerror: 1.7375e-05
    - label: sys,Bc
      symerror: 0.0035781
    (...)
    value: 0.21703
  - errors:
    - label: sys,Double_Exp_Jpsi_tau
      symerror: 0.0042903
    - label: stat,Stat
      symerror: 0.0049166
    - label: sys,DS_Bkg_Exp_mass
      symerror: 0.0033137
```

Example of full workflow

2. Upload to HEPData

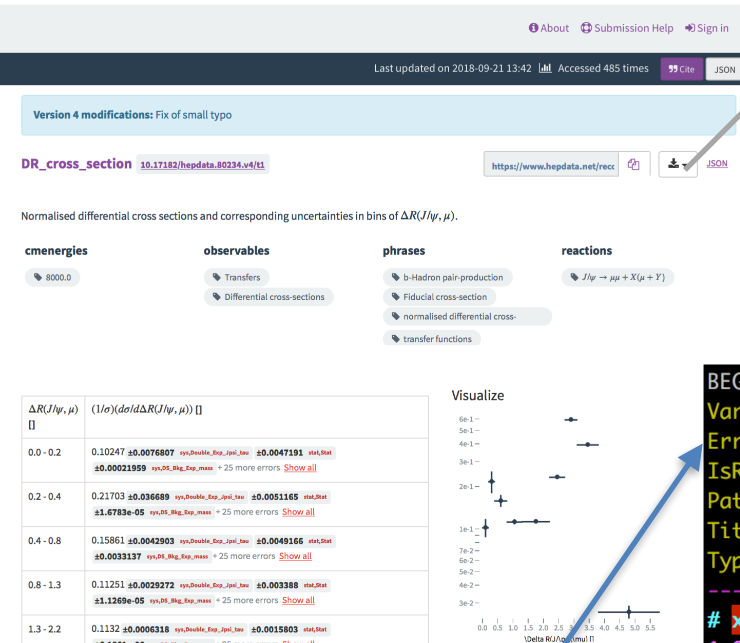
```
dependent_variables:
- header:
    name: $(1/\sigma)(d\sigma/d\Delta R(J/\psi,\mu))$
    units: ''
  values:
  - errors:
    - label: sys,Double_Exp_Jpsi_tau
      symerror: 0.0076807
    - label: stat,Stat
      symerror: 0.0047191
    - label: sys,DS_Bkg_Exp_mass
      symerror: 0.00021959
    - label: sys,Trigger
      symerror: 4.4163e-05
    - label: sys,Bc
    (...)
    value: 0.10247
  - errors:
    - label: sys,Double_Exp_Jpsi_tau
      symerror: 0.036689
    - label: stat,Stat
      symerror: 0.0051165
    - label: sys,DS_Bkg_Exp_mass
      symerror: 1.6783e-05
    - label: sys,Trigger
      symerror: 1.7375e-05
    - label: sys,Bc
      symerror: 0.0035781
    (...)
    value: 0.21703
  - errors:
    - label: sys,Double_Exp_Jpsi_tau
      symerror: 0.0042903
    - label: stat,Stat
      symerror: 0.0049166
    - label: sys,DS_Bkg_Exp_mass
      symerror: 0.0033137
```

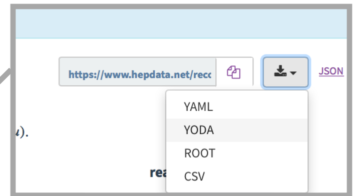


Note that uncertainty breakdown appears here!

Example of full workflow

3. Download as YODA (also possible to download whole HEPData as YODA, and to do via terminal (curl/wget) rather than manually)





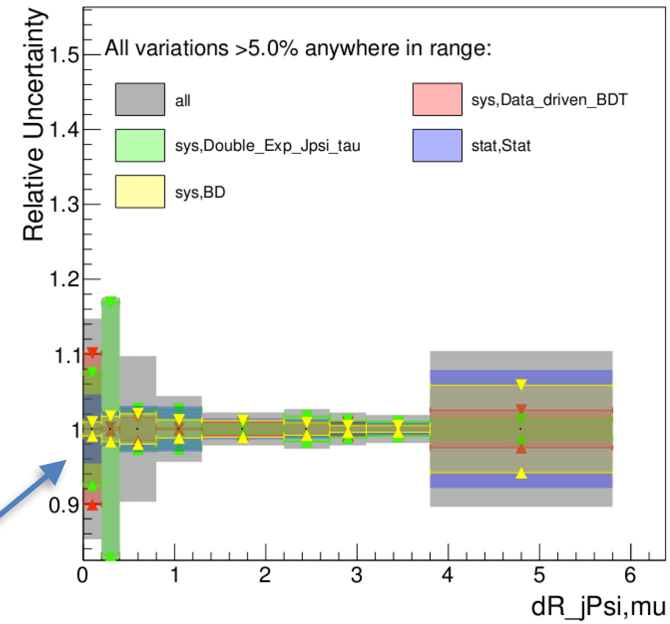

```
BEGIN YODA_SCATTER2D_V2 /REF/ATLAS_2017_I1598613/d01-x01-y01
Variations: [""]
ErrorBreakdown: {0: {'stat,Muon_Fit_Stat': {dn: -0.0016163, up: 0.0016163}, 'stat,Stat': {dn: -0.00
IsRef: 1
Path: /REF/ATLAS_2017_I1598613/d01-x01-y01
Title: doi:10.17182/hepdata.80234.v4/t1
Type: Scatter2D
---
# xval   xerr-   xerr+   yval   yerr-   yerr+
1.000000e-01 1.000000e-01 1.000000e-01 1.024700e-01 1.510541e-02 1.510541e-02
3.000000e-01 1.000000e-01 1.000000e-01 2.170300e-01 3.808887e-02 3.808887e-02
6.000000e-01 2.000000e-01 2.000000e-01 1.586100e-01 1.543319e-02 1.543319e-02
1.050000e+00 2.500000e-01 2.500000e-01 1.125100e-01 4.944104e-03 4.944104e-03
1.750000e+00 4.500000e-01 4.500000e-01 1.132000e-01 2.492913e-03 2.492913e-03
2.450000e+00 2.500000e-01 2.500000e-01 2.334300e-01 6.205529e-03 6.205529e-03
2.900000e+00 2.000000e-01 2.000000e-01 5.950400e-01 1.306213e-02 1.306213e-02
3.450000e+00 3.500000e-01 3.500000e-01 3.955500e-01 7.406674e-03 7.406674e-03
4.800000e+00 1.000000e+00 1.000000e+00 2.590500e-02 2.697030e-03 2.697030e-03
END YODA_SCATTER2D_V2
```

Note that the ErrorBreakdown Annotation can be *very* long…

Example of full workflow

4. Sanity check.. print the relative uncertainty from the main error sources

```
BEGIN YODA_SCATTER2D_V2 /REF/ATLAS_2017_I1598613/d01-x01-y01
Variations: [""]
ErrorBreakdown: {0: {'stat,Muon_Fit_Stat': {dn: -0.0016163, up: 0.0016163}, 'stat,Stat': {dn: -0.0047
IsRef: 1
Path: /REF/ATLAS_2017_I1598613/d01-x01-y01
Title: doi:10.17182/hepdata.80234.v4/t1
Type: Scatter2D
---
# xval   xerr-   xerr+   yval   yerr-   yerr+
1.000000e-01  1.000000e-01  1.000000e-01  1.024700e-01  1.510541e-02  1.510541e-02
3.000000e-01  1.000000e-01  1.000000e-01  2.170300e-01  3.808887e-02  3.808887e-02
6.000000e-01  2.000000e-01  2.000000e-01  1.586100e-01  1.543319e-02  1.543319e-02
1.050000e+00  2.500000e-01  2.500000e-01  1.125100e-01  4.944104e-03  4.944104e-03
1.750000e+00  4.500000e-01  4.500000e-01  1.132000e-01  2.492913e-03  2.492913e-03
2.450000e+00  2.500000e-01  2.500000e-01  2.334300e-01  6.205529e-03  6.205529e-03
2.900000e+00  2.000000e-01  2.000000e-01  5.950400e-01  1.306213e-02  1.306213e-02
3.450000e+00  3.500000e-01  3.500000e-01  3.955500e-01  7.406674e-03  7.406674e-03
4.800000e+00  1.000000e+00  1.000000e+00  2.590500e-02  2.697030e-03  2.697030e-03
END YODA_SCATTER2D_V2
```



Note that in this example the uncertainties fluctuate uniformly across all bins!

Example of full workflow

5. Build cov matrix using your favourite method... (here direct since errs symm!)
(Could this step eventually be provided by YODA macros?)

```
BEGIN YODA_SCATTER2D_V2 /REF/ATLAS_2017_I1598613/d01-x01-y01
Variations: [""]
ErrorBreakdown: {0: {'stat,Muon_Fit_Stat': {dn: -0.0016163, up: 0.0016163}, 'stat,Stat': {dn: -0.0047
IsRef: 1
Path: /REF/ATLAS_2017_I1598613/d01-x01-y01
Title: doi:10.17182/hepdata.80234.v4/t1
Type: Scatter2D
---
# xval    xerr-    xerr+    yval    yerr-    yerr+
1.000000e-01  1.000000e-01  1.000000e-01  1.024700e-01  1.510541e-02  1.510541e-02
3.000000e-01  1.000000e-01  1.000000e-01  2.170300e-01  3.808887e-02  3.808887e-02
6.000000e-01  2.000000e-01  2.000000e-01  1.586100e-01  1.543319e-02  1.543319e-02
1.050000e+00  2.500000e-01  2.500000e-01  1.125100e-01  4.944104e-03  4.944104e-03
1.750000e+00  4.500000e-01  4.500000e-01  1.132000e-01  2.492913e-03  2.492913e-03
2.450000e+00  2.500000e-01  2.500000e-01  2.334300e-01  6.205529e-03  6.205529e-03
2.900000e+00  2.000000e-01  2.000000e-01  5.950400e-01  1.306213e-02  1.306213e-02
3.450000e+00  3.500000e-01  3.500000e-01  3.955500e-01  7.406674e-03  7.406674e-03
4.800000e+00  1.000000e+00  1.000000e+00  2.590500e-02  2.697030e-03  2.697030e-03
END YODA_SCATTER2D_V2
```

```
def makeCovarianceMatrix( ao ):
   nbins = ao.numPoints
   corr = yaml.load(ao.annotation('ErrorBreakdown'))
   dummyM = np.outer(range(nbins), range(nbins) )
   covM = np.zeros(dummyM.shape)
   systList= corr[0].keys()
   totErr = np.zeros(nbins)
   for sname in systList:
     systErrs = np.zeros(nbins)
     for ibin in range (nbins):
       nomVal =  ao.points[ibin].y
       systErrs[ibin]=((abs(corr[ibin][sname]['up']))+
(abs(corr[ibin][sname]['dn'])))*0.5
    if ('stat' in sname):
        covM += np.diag(systErrs*systErrs)
      else:
        covM += np.outer(systErrs,systErrs)
   return covM
```

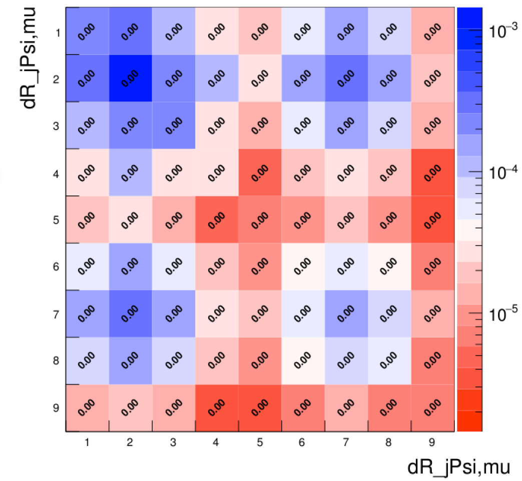
```
[[ 2.28173303e-04  3.29435630e-04  1.10111396e-04  3.00190004e-05
   1.97108219e-05  6.44227526e-05  1.42961513e-04  7.34611846e-05
   1.24176489e-05]
 [ 3.29435630e-04  1.45076182e-03  2.13759613e-04  1.17289194e-04
   3.15759532e-05  1.67029999e-04  3.01962740e-04  1.73371644e-04
   2.01983707e-05]
 [ 1.10111396e-04  2.13759613e-04  2.38183304e-04  2.50477381e-05
   1.48120466e-05  5.77436557e-05  1.41292172e-04  8.21119517e-05
   1.16452355e-05]
 [ 3.00190004e-05  1.17289194e-04  2.50477381e-05  2.44441690e-05
   4.31815221e-06  1.66529144e-05  2.96900263e-05  1.78323553e-05
   3.46781334e-06]
 [ 1.97108219e-05  3.15759532e-05  1.48120466e-05  4.31815221e-06
   6.21461535e-06  8.29854738e-06  1.63399228e-05  9.27310836e-06
   2.96303612e-06]
 [ 6.44227526e-05  1.67029999e-04  5.77436557e-05  1.66529144e-05
   8.29854738e-06  3.85085901e-05  5.93264837e-05  3.32963622e-05
   6.16708420e-06]
 [ 1.42961513e-04  3.01962740e-04  1.41292172e-04  2.96900263e-05
   1.63399228e-05  5.93264837e-05  1.70619184e-04  7.23961275e-05
   1.24297352e-05]
 [ 7.34611846e-05  1.73371644e-04  8.21119517e-05  1.78323553e-05
   9.27310836e-06  3.32963622e-05  7.23961275e-05  5.48588151e-05
   6.77928486e-06]
 [ 1.24176489e-05  2.01983707e-05  1.16452355e-05  3.46781334e-06
   2.96303612e-06  6.16708420e-06  1.24297352e-05  6.77928486e-06
   7.27396970e-06]]
```

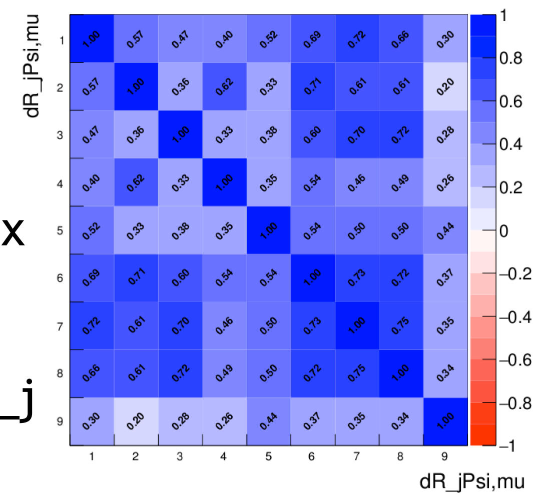
Example of full workflow

6. Sanity check the covariance matrix by plotting it!

```
[[ 2.28173303e-04   3.29435630e-04   1.10111396e-04   3.00190004e-05
   1.97108219e-05   6.44227526e-05   1.42961513e-04   7.34611846e-05
   1.24176489e-05]
 [ 3.29435630e-04   1.45076182e-03   2.13759613e-04   1.17289194e-04
   3.15759532e-05   1.67029999e-04   3.01962740e-04   1.73371644e-04
   2.01983707e-05]
 [ 1.10111396e-04   2.13759613e-04   2.38183304e-04   2.50477381e-05
   1.48120466e-05   5.77436557e-05   1.41292172e-04   8.21119517e-05
   1.16452355e-05]
 [ 3.00190004e-05   1.17289194e-04   2.50477381e-05   2.44441690e-05
   4.31815221e-06   1.66529144e-05   2.96900263e-05   1.78323553e-05
   3.46781334e-06]
 [ 1.97108219e-05   3.15759532e-05   1.48120466e-05   4.31815221e-06
   6.21461535e-06   8.29854738e-06   1.63399228e-05   9.27310836e-06
   2.96303612e-06]
 [ 6.44227526e-05   1.67029999e-04   5.77436557e-05   1.66529144e-05
   8.29854738e-06   3.85085901e-05   5.93264837e-05   3.32963622e-05
   6.16708420e-06]
 [ 1.42961513e-04   3.01962740e-04   1.41292172e-04   2.96900263e-05
   1.63399228e-05   5.93264837e-05   1.70619184e-04   7.23961275e-05
   1.24297352e-05]
 [ 7.34611846e-05   1.73371644e-04   8.21119517e-05   1.78323553e-05
   9.27310836e-06   3.32963622e-05   7.23961275e-05   5.48588151e-05
   6.77928486e-06]
 [ 1.24176489e-05   2.01983707e-05   1.16452355e-05   3.46781334e-06
   2.96303612e-06   6.16708420e-06   1.24297352e-05   6.77928486e-06
   7.27396970e-06]]
```

Draw covariance matrix

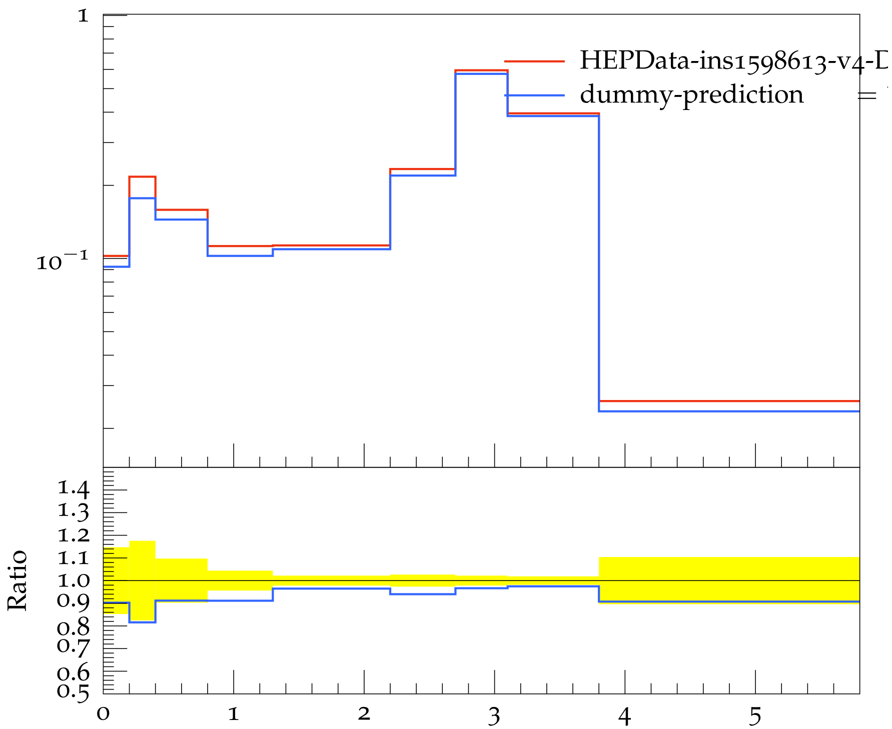




Correlation matrix obtained by dividing each bin by sigma_i*sigma_j

Example of full workflow

7. Compare to some MC prediction (here just a dummy which I made up by manually changing the Y-axis values... just to illustrate !)



chi2/NDOF w/ covariance info =10.18 / 9
chi2/NDOF w/o covariance info = 19.07 / 9

including cov info here reduces the chi2, since the difference could be explained by a single coherent down fluctuation in all bins!

A common Library of Tools

We have put together a small library of tools/example to help users manipulate covariance info the YODA ErrorBreakdown format: https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library

- how to download a HEPData entry in YODA from a python script:
- check if a YODA file has a valid ErrorBreakdown
- make covariance matrix from ErrorBreakdown (direct or toys method)
- make correlation matrix from covariance matrix
- plot the individual variations

example of full workflow provided : https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library/blob/master/example.py

all functions in library documented : https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library/blob/master/RivetAnalysesWithErrorBreakdown.md

table of which HEPData entries (those which have a rivet routine) have ErrorBreakdowns

A common Library of Tools

Put together small library of tools/example to help users manipulate covariance info in ErrorBreakdown format: https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library :

- download HEPData entry in YODA from python script:
- check if a YODA file has a valid ErrorBreakdown
- make cov matrix from ErrorBreakdown (direct or toys method)
- make correlation matrix from covariance matrix
- plot the individual variations

example of full workflow provided : https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library/blob/master/example.py

all functions in library documented : https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library/blob/master/RivetAnalysesWithErrorBreakdown.md

getCorrelationInfoFromWeb(inspireID=Non

`inspireID` String (inspireID of HEPData record you want to dow
`dlformat` String [default=yoda] (the download format, yoda, yan

This method downloads a HEPData entry from the web, given an

hasAsymmetricErrors(ao)

`ao` Yoda AO

Check if the ErrorBreakdown of a given analysis object has asymr

hasValidErrorBreakdown(ao)

`ao` Yoda AO

Check if the ErrorBreakdown is non-empty and has not error sour

makeCovarianceMatrix(ao, ignore_corrs=F

`ao` Yoda AO
`ignore_corrs` Bool (option to ignore correlations and treat every

Build the covariance matrix for an AO with the direct method trea
label contains 'stat' as uncorrelated between bins and others as c
Fast but needs symmetric errors.

table of which HEPData entries (those which have a rivet routine) have ErrorBreakdowns https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library/blob/master/RivetAnalysesWithErrorBreakdown.md

Recap

- Need to be **smarter about data/MC agreement**. We need to start using **using covariance info** for Data/MC comparison and re-interpretations
- Ideally, access **detailed breakdown of errors** and/or exact cov matrices. Cov matrix can be re-built from error breakdown (direct or toys method)
- **HEPData.net should be able to provide this information** in YODA format.
- Put in place developments soYODA to store uncertainty breakdown, and HEPData to **propagate uncertainty breakdowns stored as labels**.
- After reconstructing covariance matrix from breakdown, we can use it in a chi2, or in a simplified likelihood, etc... as needed. Scripts to do this provided in (growing) library: https://gitlab.cern.ch/lhcewkwg/lhcewkwg-vjets/correlations-library
- **So we have the basic workflow in place 😀**

Where do we go from here? (for discussion..)

- Can we use **YODA annotations** to **link covariance matrix** (eg stored as Scatter2D) **to histograms** ? (Need to define some sensible format, but in principle straightforward...). Would need a way to specify this information in HEPData.net upload format.
- Provided **tools to convert error breakdown into cov matrix** with either toys or direct method (even across multiple distributions)... but should we eventually **backwards-propagate into Rivet mk-plots and YODA ?**
- For error breakdown... could we aim to **synchronise definitions of main uncertainty sources across experiments** (eg lumi? JES?) so that we could build correlation matrix across analyses?
- Now that we have a potential 'standard' how do we **ask HEPData submitters to use this standard format consistentl**y? (Game from HEPData has already bought this to attention of relevant contacts in CMS/ATLAS)
- Should we **revisit older HEPData entries to comply with this format**?
    - Already o(30) which can be used right now

Direct/Toys method pseudo-code error-breakdown --> cov matrix

- Method 1 : Direct propagation

Matrix	No	Toys							
0	1	2	3	4	5	6	7	8	9
793.000	19.200	88.000	28.800	23.200	64.000	56.000	24.800	9.600	0.560
19.200	109.800	26.400	8.640	6.960	19.200	16.800	7.440	2.880	0.168
88.000	26.400	1802.000	39.600	31.900	88.000	77.000	34.100	13.200	0.770
28.800	8.640	39.600	234.970	10.440	28.800	25.200	11.160	4.320	0.252
23.200	6.960	31.900	10.440	118.660	23.200	20.300	8.990	3.480	0.203
64.000	19.200	88.000	28.800	23.200	640.000	56.000	24.800	9.600	0.560
56.000	16.800	77.000	25.200	20.300	56.000				
24.800	7.440	34.100	11.160	8.990	24.800				
9.600	2.880	13.200	4.320	3.480	9.600				
0.560	0.168	0.770	0.252	0.203	0.560				

```
 uncorr = numpy.array(all_errs_uncorrelated)
#here, stat error
 corr = numpy.array(all_errs_correlated)
#here, syst error
 covM = numpy.zeros(nBins,nBins) # square
matrix of zeros
 for u in uncorr.T:  #uncorrelated component
      covM += numpy.diag(u*u) #diagonal
      matrix of errs squared
```

- Method 2: Toys (here N=10000)

0	1	2	3	4	5
794.702	21.436	58.744	27.411	24.436	57.594
21.436	111.482	32.040	11.204	8.724	23.120
58.744	32.040	1801.295	40.565	30.649	93.656
27.411	11.204	40.565	228.914	10.386	29.533
24.436	8.724	30.649	10.386	119.954	26.615
57.594	23.120	93.656	29.533	26.615	636.709
48.741	17.057	81.589	25.179	20.436	55.279
26.206	5.240	32.921	11.831	9.528	24.605
8.589	3.714	14.573	5.396	4.263	8.933
0.463	0.170	0.941	0.213	0.206	0.707

```
toys_abs  = np.zeros( [ ntoys, nbins ] ) #array of ntoys pseudodatasets
for itoy in range( ntoys ): #loop over each toy
 delta = np.zeros( nbins )
 for syst in systematics():
  if (uncorrelated):  #uncorrelated case (stat here)
     for n in range(nbins):
      l = rand3gen.Gaus(0,1) #bin-bin independent stat fluctuations
      # random number thrown once per bin to model this
      s = syst['up'][n] if l >= 0. else syst['down'][n] )
      #pick up error for positive shift, down for negative shift
      delta[n] += abs(l) * s
  else:  # correlated systematics (syst here)
     l = rand3gen.Gaus(0,1) #now the random number is thrown
     #once per systematic, not once per bin
     for n in range(nbins):
       s = syst['up'][n] if l >= 0. else syst['down'][n]
       delta[n] += abs(l)  * s
 for n in range(nbins):
    toys_abs[itoy][n] =  ( 1. + delta[n] ) #fill the toy datasets
C_abs = np.cov( toys_abs, rowvar=0 ) #numpy method calculates the covarianc
# each entry is <XY>-<X><y>, where < > denotes the average over the ntoys
```

CONTUR

- Example: CONTUR uses **Rivet Analyses** to **set exclusion limits on new signal** models
- For now, largely assumes **total uncertainty is uncorrelated** between bins: fine for peaking signal model where sensitivity is driven by single bin…
- … but what about **non-resonant signals**? Need experimental **covariance** information **to accurately set exclusion limits**!

From 'CONTUR' paper: https://arxiv.org/pdf/1606.05296.pdf