

# CERN Summer Student Report: A new tool for LHCb to interact with the HEPData repository

Harvey Turner

Supervised by William Barter, Adam Davis and Chris Burr

September 2021

## 1 Abstract

This report summarises the work I did as a summer student with the LHCb working group at CERN from July to September 2021. The work I did focused on creating a new tool to convert *.tex* data files into *.yaml* files, such that they can be submitted to the HEPData repository. This would ultimately allow for greater accessibility of LHCb data.

## 2 Background

### 2.1 HEPData

HEPdata is an online repository for experimental particle physics data [1]. It aims to simplify and facilitate collaboration with its central, open database where researchers in the field can always find the latest results. Data submitted to HEPData must be in *yaml* format. Converting data into this format is currently done mostly by hand or by writing bespoke code, which can be tedious and time consuming. Consequently, less than 15% of LHCb data is currently recorded in HEPdata.

The aim of this project was to create a tool that can automate much of this process. The tool will take as input a *.tex* file, as any published paper will have its tables in LaTeX format. *.tex* files can also be downloaded for any table found in `lhcbproject.web.cern.ch/lhcbproject/Publications`, so this is the most convenient input file type for submitting more LHCb data to HEPData.

## 2.2 Tools used during the project

The code was written in Python, with use of the *arparse* and *numpy* modules. gitlab was used for data management. The documentation for HEPData submission served as the main reference point when designing *.yaml* files [2]. The Python hepdata-validator package (<https://github.com/HEPData/hepdata-validator>) was used to validate *.yaml* files, as was <http://www.yamllint.com/>.

## 3 Using the tool

The program is designed to be run from the command line. To do this, it makes use of the *arparse* module. Details about the file are specified in the command line arguments:

- -file specifies the name of the file, e.g. -file “Table\_3.tex”
- -format specifies the contents of each column in the table:
  - i denotes an independent variable, and d denotes a dependent variable
  - sys denotes a systematic error, stat a statistical error, lumi a luminosity error, asym an asystematic error, or e for an unspecified error. Anything else will be interpreted as a custom error label.

Format items should be delimited by a space, for example, -format “i d stat sys”

- -covariance and -correlation are flags to signal that the file is a covariance/correlation matrix. In this case, a format need not be specified
- -qual is an optional addition to specify a text file where qualifiers for the data can be found, e.g. -qual “Qualifiers.txt”
- -remove is an optional argument to specify particular strings that should be removed from the input text. The tool has been written to identify and remove common LaTeX formatting commands, but the remove argument can be used to specify more obscure LaTeX strings that should be removed. An unlimited number of strings can be entered as a list delimited by spaces, for example -remove “\string1 \string2 \string3”

## 4 Internal architecture

The tool is written in Python, and makes use of the *arparse* and *numpy* modules. It first reads in the *.tex* file specified by the user line by line. It aims to strip the LaTeX commands and special characters which are only used for formatting, and leave only what is directly related to the data. Given the large number of commands and packages that can be used in LaTeX, however, it cannot be guaranteed that all formatting commands will be removed automatically: the user has the option to specify other commands/strings that should also be removed.

Before the yaml file is created, the data from the *.tex* file is stored in intermediate objects. The program uses three classes to handle data:

- *uncertainty*: an instance of this class contains the numerical value of the uncertainty (or a pair of numbers in the asymmetric case) and a description of the type of uncertainty. The description can be “sys”, “stat”, “lumi”, “asym”, “e” (as they are used in the -format flag), or a custom error label.
- *value*: this represents a measurement and its associated uncertainties. It contains a principal value, and a list of uncertainty objects (which can optionally be empty).
- *variable*: this represents a column in a table: it contains a header (in yaml format), and a list of value objects. It also has a boolean attribute that signals whether it is an independent variable or not, and an optional qualifiers attribute that contains the name of the text file containing the qualifiers (if specified by user).

The data is read in from the *.tex* file, and the program separates the data from the table into a 2D numpy array called *data\_array*. The items in the array will be value objects. In the correlation/covariance case, each item in the table is simply converted into a value object. Otherwise, the program makes reference to the user-specified format: an entry in a column that corresponds to an “i” or a “d” becomes a new value object, but all other entries are converted to uncertainty objects and then appended to the appropriate value object.

A list called *variables* is created to store variable objects. The program then goes through *data\_array* column by column, again making reference to the format specified by the user. If a column corresponds to “i” or “d” in the format specified by the user, then a new variable object is created and appended to *variables*. Each entry in that column is then appended to the variable object as a value object. The variable header is determined using

a custom function called *extract\_header* which isolates the column header and reformats it to suit *yaml* standards.

Alternatively, in the correlation/covariance case, *variables* will contain 3 variable objects. The first two are independent and represent the x and y axes. The third is dependent and contains all the correlations/covariances. The independent variable objects will contain repeated values ordered in such a way that all data can be encoded (see [https://hepdata-submission.readthedocs.io/en/latest/data\\_yaml.html#two-dimensional-measurements](https://hepdata-submission.readthedocs.io/en/latest/data_yaml.html#two-dimensional-measurements) for details).

Because correlation/covariance matrices are symmetric about the leading diagonal, they are often given in papers in triangular format, with dashes or empty spaces in the other half. This is not acceptable for HEPData, however, as all spaces in a table must be filled; it will replace dashes or empty spaces with zeros. Therefore, the program is designed to identify whether a correlation/covariance matrix is triangular, and if so, to reflect the data about the diagonal so that the empty spaces are filled.

For each class, the *\_\_repr\_\_(self)* function is overwritten so that each object will be automatically be outputted in yaml format, so writing the object to the *.yaml* file is simple. The output file is created with the same name as the input file, with *.tex* replaced by *.yaml*. Each variable object in *variables* object is written to the file, with qualifiers included if specified.

## 5 Testing the program

The code was first tested with the data from <https://lhcbproject.web.cern.ch/lhcbproject/Publications/LHCbProjectPublic/LHCb-PAPER-2016-021.html>. This involved converting 11 files, some of which were correlation/covariance matrices. These revealed several bugs in the code which needed to be resolved. Once this was done, the *.yaml* files were uploaded to HEPData (<https://www.hepdata.net/record/110162>). The submission was then reviewed and approved.

A similar conversion was then performed on 17 data files from <https://lhcbproject.web.cern.ch/lhcbproject/Publications/LHCbProjectPublic/LHCb-PAPER-2020-018.html>. This again exposed several issues with the code that were resolved, which included adding a feature to handle asymmetric uncertainties. The converted files were then uploaded to HEPData (<https://www.hepdata.net/record/110430>), and are currently under review.

## 6 The future of the tool

I hope that this tool will be of use to the LHCb group. In testing, the records at <https://www.hepdata.net/record/110162> and <https://www.hepdata.net/record/110430> were made. This involved running a total of 28 files, all of which were successfully converted to *yaml* files suitable for HEPData submission. It has been designed to handle the general case of files such as these. For such files, I hope that the tool should be able to create the corresponding *yaml* files quickly and without a need to manually modify the input files, output files or code. In non-general cases (for example where an obscure LaTeX package has been used), the `remove` argument can be used from the command line to specify additional LaTeX commands to be removed, so the tool should still be an efficient option for HEPData submission.

## 7 Acknowledgements

I would like to thank CERN for the incredible opportunity to participate in the summer student programme. I am very grateful to everyone whose work made this programme and the fantastic accompanying lectures possible, especially given this year's difficult circumstances. I would particularly like to give a warm thank you to William Barter, Adam Davis and Chris Burr for their continued support and guidance with this project.

## References

- [1] About HEPData. "<https://www.hepdata.net/about>". Accessed: 01/09/2021.
- [2] HEPData Submission. "<https://hepdata-submission.readthedocs.io/en/latest/>". Accessed: 01/09/2021.