Instructions for HEP Data Tool

General Information:

Main file for the submission of a paper is: “document.yaml”

* This file links all data tables into one submission
* Specifies auxiliary files such as scripts used to create data, ROOT files, or even links to github
* Publication info are taken directly from the INSPIRE record and does not need to be included in the .yaml file

For CMS users🡪 would be advisable to run everything on lxplus

$ scram v1 projects CMSSW CMSSW\_10\_2\_3

$ cd CMSSW\_10\_2\_3/src

$ cmsenv

$ cd ..

$ virtualenv -p $(which python) hepdata\_lib\_env

$ cd hepdata\_lib\_env

$ source bin/activate

$ python -m pip install hepdata\_lib

The library “hepdata\_lib” aims to offer tools for two operations.

1. Reading data from formats such as ROOT, txt into python lists
2. Writing data from python lists into the HEP Data YAML-based format

Reading Data

* Reading data from plain text: numpy tool 🡪 loads column\_wise data from txt and returns a python list
  + >>import numpy as np
  + >>my\_array = np.loadtxt(“file.tx”)
* Reading from ROOT files (TGraph, TH1, TH2)
  + RootFileReader class (included in the hepdata\_lib environment)
  + To read each type:
    - TGraph, TGraphErrors, TGraphAssymErrors:
      * RootFileReader.read\_graph
    - TH1
      * RootFileReader.read\_hist\_1d
    - TH2
      * RootFileReader.read\_hist\_2d
* Example for reading ROOT files

>>from hepdata\_lib import RootFileReader

>>reader = RootFileReader(“myRootFile.root”)

>>data = reader.read\_hist\_1d(“directory/th1\_name”)

>>print(data)

Since a graph/th1 is just a set of (x,y) pairs, for each point the “data” variable is a [dictionary](https://www.w3schools.com/python/python_dictionaries.asp) that has (at least) two key values:

1. Key “x” 🡪 list of all x values (Bin centre)
2. Key “y” 🡪 list of all y values (Bin Content)

For TH2 objects:

1. Key “x” 🡪 list of all x values (x-axis)
2. Key “y” 🡪 list of all y values (y-axis)
3. Key “z” 🡪 list of all values of the hist

EXAMPLE

1. Create a Submission

>> from hepdata\_lib import Submission

>> submission = Submission()

1. Let’s assume that I want to get a Figure 4a from page 12 of a paper.

>> from hepdata\_lib import Table

>> table = Table(“Figure 4a”)

>> table.description = “a simple description of the table”

>> table.location = “Data from figure 4a, located on page 12”

>> table.keywords[“observables”] = [“N”]

>> table.add\_image(“input/filename/file.pdf”)

1. Read now the actual data from several ROOT files

>> from hepdata\_lib import RootFileReader

>> reader = RootFileReader(“rootFile.root”)

>> reader\_data = RootFileReader(“rootFileData.root”)

>> reader\_signal= RootFileReader(“rootFileSignal.root”)

>> TotalBkg = reader.read\_hist\_1d(“th1\_bkg”)

>> ttMC = reader.read\_hist\_1d(“th1\_tt”)

>> Data = reader\_data.read\_hist\_1d(“th1\_data”)

….

The reader.read\_hist\_1d returns a dictionary

The available keys for the aforementioned dictionary are: [‘x’, ‘y’, ‘x\_edges’, ‘dy’]

1. Now define the variables

>> from hepdata\_lib import Variable

#let’s assume that x-axis is the b-quark mass and y-axis is the Number of events

>> mmed = Variable(“M\_{bH}”, is\_*independent* = True, *is\_binned* = False, *units* = “GeV”)

>> mmed = signal[“x”] #from the dictionary signal get the values with key “x”

>> sig = Variable(“Number of events”, is\_*independent* = True, *is\_binned* = False, *units* = “”)

>> sig.values = signal[“y”]

>> data\_var = Variable (“Number of events”, is\_*independent* = True, *is\_binned* = False, *units* = “”)

>> data\_var.values = Data[“y”]

>> total\_bkg = Variable (“Number of events”, is\_*independent* = True, *is\_binned* = False, *units* = “”)

>> total\_bkg.values = TotalBkg [“y”]

….

Same for all the processes!!

From every dictionary you get the respected “y”

1. Associated uncertainties

>> from hepdata\_lib import Uncertainty

>> unc\_totalBkg = Uncertainty(“total unc”, *is\_symmetric* = True)

>> unc\_totalBkg.values = TotalBkg[“dy”] *#from the dictionary get the key dy = uncertainty of the histogram*

>> unc\_data = Uncertainty(“data\_unc”, *is\_symmetric* = True)

>> unc\_data.values = Data[“dy”]

>> total\_bkg.add\_uncertainty(unc\_totalBkg) *#add the unc to the variable*

>> data\_var.add\_uncertainty(unc\_data)

….

1. Now add the variables to the “Table” object and then the “Table” object to the “Submission” object and create the files:

>>table.add\_variable(mmed)

>>table.add\_variable(data\_var)

>>table.add\_variable(total\_bkg)

>>table.add\_variable(sig)

>>submission.add\_table(table)

>>submission.create\_files(“example\_output”) *#this is the directory in which the .yaml files will be created*

Class TableAndHist

The purpose of this class is to make our life easier when applying tables in the hepdata tool

Idea:

All you need is 3 dictionaries:

To begin with, let’s assume that you have 3 tables (histograms in the pdf file/paper): pdf1, pdf2 and pdf3

1. for each table in the paper you need a some information to initialize the “Table” object

This means you will have a dictionary called tableInfo of type:

tableInfo["pdf1"]=[description1, location1, keywords1, image1]

tableInfo["pdf2"]=[description2, location2, keywords2, image2]

tableInfo["pdf3"]=[description3, location3, keywords3, image3]

2. Now you have to assign each table obj with the correct histograms taken from the root file:

The root file has to have the right tree structure. File-->tableName/Histograms[1,...N]

so you again have a dictionary like this:

tableInfo["pdf1"]=[histo1pdf1, histo2pdf1, histo3pdf1, histo4pdf1]

tableInfo["pdf2"]=[histo1pdf2, histo2pdf2, histo3pdf2, histo4pdf2]

tableInfo["pdf3"]=[histo1pdf3, histo2pdf3, histo3pdf3]

3. For each histogram you need to have some information

Here we want the user to create a bulk dictionary with the histogram name as a key and some info as data:

For example:

histogramInformation["histo1pdf1"]=[xvar, yvar, histoTitle, isIndependent ,isBinned]

histogramInformation["histo1pdf2"]=[xvar, yvar, histoTitle]

.

.

.

histogramInformation["histo3pdf3"]=[xvar, yvar, histoTitle]

where isBinned and is isIndependent are Booleans.

First of all, the initial procedure is the same. The user needs to create a “Submission” object and the directory where the root files are..

>>submission = Submission()

>>submission.read\_abstract("examples/abstract.txt")

>>submission.add\_link("cds", "https://cds.cern.ch/record/2709851?")

>>submission.add\_link("cadiLine", "http://cms.cern.ch/iCMS/analysisadmin/cadilines?line=TOP-18-013&tp=an&id=2177&ancode=TOP-18-013")

>>submission.add\_record\_id(2709851, "cds")

>>rootFile="examples/exampleDocument.root"

The user needs to create or import a dictionary of the type:

#The first dictionary includes information about each table object

>>tableInfo={}

>>tableInfo["pdf1"]=["description1", "location1", keywords, "examples/pdf1.pdf"]

>>tableInfo["pdf2"]=["description2", "location2", keywords, "examples/pdf2.pdf"]

>>tableInfo["pdf3"]=["description3", "location3", keywords, "examples/pdf3.pdf"]

Then the user needs to create a second dictionary which includes the histograms that are contained in each table:

>>tableHistograms={}

>>tableHistograms["pdf1"]=["histo1pdf1", "histo2pdf1", "histo3pdf1", "histo4pdf1"]

>>tableHistograms["pdf2"]=["histo1pdf2", "histo2pdf2", "histo3pdf2", "histo4pdf2"]

>>tableHistograms["pdf3"]=["histo1pdf3", "histo2pdf3", "histo3pdf3"]

Finally the user has to create a dictionary that contains the information for each histogram:

>>histogramInformation={}

>>histogramInformation["histo1pdf1"]=["x1","y1","title11",True,False,"UNITS"]

>>histogramInformation["histo2pdf1"]=["x1","y1","title21",True,False,"UNITS"]

>>histogramInformation["histo3pdf1"]=["x1","y1","title31",True,False,"UNITS"]

>>histogramInformation["histo4pdf1"]=["x1","y1","title41",True,False,"UNITS"]

>>histogramInformation["histo1pdf2"]=["x2","y2","title12",True,False,"UNITS"]

>>histogramInformation["histo2pdf2"]=["x2","y2","title22",True,False,"UNITS"]

>>histogramInformation["histo3pdf2"]=["x2","y2","title32",True,False,"UNITS"]

>>histogramInformation["histo4pdf2"]=["x2","y2","title42",True,False,"UNITS"]

>>histogramInformation["histo1pdf3"]=["x3","y3","title13",True,False,"UNITS"]

>>histogramInformation["histo2pdf3"]=["x3","y3","title23",True,False,"UNITS"]

>>histogramInformation["histo3pdf3"]=["x3","y3","title33",True,False,"UNITS"]

Now you have to iterate on all keys that the tableInfo dictionary has. In each iteration, create a “TableAndHisto” object:

The input for the object to be initiated is:

1. Key name (the pdf file name) , 2. rootFile (not the path!)

>> table=[]

>> tableInfoKeys = tableInfo.keys() *#contains the keys as a list*

>> for i in range(0,len(tableInfoKeys)):

>> table.append(TableAndHisto(tableInfoKeys[i],rootFile))

Now the user has created an array of “TableAndHisto” objects.

Next step is to pass the histogram that are included in each table

**REMINDER/WARNING: THE HISTOGRAMS HAVE TO BE IN THE CORRECT ROOT FILE SUBDIRECTORY!!!**

Iterate all over the tableInfo keys (number of tables you want to add)

>> for i in range(0,len(tableInfoKeys)):

>> #1st thing is to give the table information for each key taken

>> #from the tableInfo

>> table[i].init\_table(tableInfo[tableInfoKeys[i]])

>> #for each table we have a list of histograms in the

>> #tableHistograms dictionary. We want to iterate on all the histograms

>> #that have this key. For each of these histograms we add them to the

>> #TableAndHisto object

>> for hName in tableHistograms[tableInfoKeys[i]]:

>> #you take the name from the above iteration and use it to find

>> #the corresponding histogmram 🡪 the name works as the key for

>> #the histogramInformation dictionary and returns a list on info

>> table[i].addHistogram(hName, histogramInformation[hName])

>> #add the table in the submission

>> submission.add\_table(table[i].table)

>> #create the files in the input dir

>> submission.create\_files("example\_output\_finalTest")

>> for it in table: #delete all the objects

>> del it