2) Combine module

Combine module uses the latino structure [1] for datacard production. One can produce the datacards input histograms by using AnalysisCMS, LatinoAnalysis (mkShapes.py) or another framework if it is keeping the specific way of histograms storage.

The code is composed by several scripts [2]. A brief description is now provided:

MassPointList.txt:

This text file lists all the mass suffixes used to distinguish between the different points of mass used in this analysis. Each of one suffixes correspond to a separate .root file where the histograms of that point of mass are stored. This can be optimized for individual needs.

* From now, a massPoint of MassPointList.txt will be denoted by **$1**

configuration.py:

The main script where you set the variables (**variables.py**), the samples (**sample.py**), the cuts (**cuts.py**), the kind of sample (**structure.py**), and the uncertainties (**nuisances.py**) you are going to include in your datacard. Also set the *lumi*, and *output/input directions*.

mkPseudoData.py: ./mkPseudoData.py --pycfg=configurations.py

Input: configuration.py
Output: 01_PseudoDataSmeared.root
(in the same folder of all your .root Ex: minifiles/rootfiles/nominal/Stop/)

The Higgs Combine Tool needs a data yield. If there is not any, or you are blinded, this script can produce a pseudodata histograms for all the variables in *variables.py* in all cuts of *cuts.py* and store them under the usual directory of samples. These histograms are filled with the yields of all samples listed in *samples.py* and marked as background in *structure.py*. Also the statistic error of each bin is stored by doing *SetBinError(iBin, math.sqrt(abs(yValue)))*. Apart from the numeric value of yields, it may be interesting imitate the shape of the data distributions. A kind of smearing of the yields in each bin has been included for that purpose. This procedure must be revaluated by each analyzer.

mkShapesFromHistos.py: ./mkShapesFromHistos.py --pycfg=configuration.py --signalPoint=$1

Input: configuration.py && $1
Output: Shapes/Shapes_$1.root

This script produce the input file *Shapes_$1.root* for make the datacard with *mkDatacards.py* . This file contains for each cut, each variable and for each channel all the histograms that Combine Higgs Tool needs. The script basically create a 1 folder per cut and channel where store 1 folder per variable where keep one histo per sample and one histo per uncertainty (*nuisance*) to be considered. An example below. Each nuisance-histogram is created only for the selected sample in **nuisances.py**

The script also sets the lumi and checks the bad bins ie. bin with content < 0. Setting this content to 0.001. we are safe of Combine complaints

It creates two kind of histograms in function of the statistic uncertainty:

histo_SampleName_statUp (Down) -> histo per sample (global statistic uncertainty)
histo_SampleName_iBin_#bin_statUp (Down) -> histo per sample and bin (shape statistic uncertainty)

And rename conveniently the other histograms as follow:

histo_SampleName -> histos per sample
histo_SampleName_SystematicUp (Down) -> histos per sample and systematic uncertainty

For now, it requires rootfile having levels called as the cuts in cuts.py (to be more precise, as "cutname" in the "cutname_channel") and the samples in samples.py to be called as the rootfiles name (one for each process). This can be made more flexible if the tools will be used in the future.

Ex: Shapes/Shapes_mStop-250to350_Sm300_Xm125.root

mkDatacards.py: ./mkDatacards.py --pycfg=configuration.py --inputFile=./Shapes/Shapes$1.root

Input: configuration.py && ./Shapes/Shapes$1.root
Output: outputDirDatacard / cutName / variableName / datacard.txt
outputDirDatacard / cutName / variableName / shapes / histos_cut_channel .root

This script produce the datacard (txt file) and store them together with the histos under a folder. This folder will be the input of Combine Higgs Tool.

For now the outputDirDatacard is a folder called tempDatacards/. Also, at this moment there is a duplicated work storing the histograms (*** see datacards latinos questions). This will be soon checked in detail.

createPointDatacards.sh: ./createPointDatacards.sh $1

This script run ./mkShapesFromHistos.py and ./mkDatacards.py for one point of mass ($1) of *MassPointList.txt*. Also move tempDatacards/* to /Datacards/MassPoint$1/

createDatacards.sh: ./createDatacards.sh

The code uses recursively ./createPointDatacards.sh $1 for all points of mass ($1) of *MassPointList.txt*.

At this moment we have the folder Datacard/ , here we have a subfolder per point of mass, inside of each one, we find one folder per cut. Inside these “cut folders” we have one folder per variable and finally inside this “variable folders” we get the real inputs of Higgs Combine Tool:

datacard.txt shapes/histos_cut_channel .root

```
[bchazinq@lxplus0034 T2tt]$ ls Datacards/MassPoint_mStop-400to1200_Sm450_Xm363/
SR1_NoTag_ee  SR1_Tag_ee  SR2_NoTag_ee  SR2_Tag_ee  SR3_NoTag_ee  SR3_Tag_ee  datacardFinal.root  datacardSR1T.txt  datacardSR2V.txt  datacardTag.txt
SR1_NoTag_em  SR1_Tag_em  SR2_NoTag_em  SR2_Tag_em  SR3_NoTag_em  SR3_Tag_em  datacardFinal.txt   datacardSR1V.txt  datacardSR3T.txt
SR1_NoTag_mm  SR1_Tag_mm  SR2_NoTag_mm  SR2_Tag_mm  SR3_NoTag_mm  SR3_Tag_mm  datacardNoTag.txt   datacardSR2T.txt  datacardSR3V.txt
[bchazinq@lxplus0034 T2tt]$ ls Datacards/MassPoint_mStop-400to1200_Sm450_Xm363/SR1_NoTag_ee/
MT2ll
[bchazinq@lxplus0034 T2tt]$ ls Datacards/MassPoint_mStop-400to1200_Sm450_Xm363/SR1_NoTag_ee/MT2ll/
datacard.txt  shapes
[bchazinq@lxplus0034 T2tt]$ ls Datacards/MassPoint_mStop-400to1200_Sm450_Xm363/SR1_NoTag_ee/MT2ll/shapes/
histos_SR1_NoTag_ee.root
[bchazinq@lxplus0034 T2tt]$
```

Next step is to combine all these datacards following the analysis criteria. In this case, stop analysis requires a complex combination of regions.

For each point of mass, we want to combine the three channels of each cut. Then, combine again the resulting datacards in two ones, Tag vs NoTag. At last, combine once again this two datacards to have the Final datacard.

For that purpose, two scripts were written:

combinePointDatacards.sh: ./combinePointDatacards.sh $1

> This script executes the code in “combineCards.py” (code of Higgs Combine Tool) in the sequence described above for each point of mass. The resulting data card, *datacardFinal.txt,* is stored in Datacards/MassPoin$1/

combineDatacards.sh: ./combineDatacards.sh

> The code uses recursively ./combinePointDatacards.sh $1 for all points of mass ($1) of *MassPointList.txt*.

The last step will be compute an estimation of the observed and expected limit for each point of mass. The **asymptotic CL_S method** is the option used here. More information about this method or another one in [4].

Three files to use:

runPointLimit.sh : ./runPointLimit.sh $1

> This script run Combine Higgs Tool with the asymptotic method using the last datacard of the combination, *Datacards/MassPoint$1/datacardFinal.txt* and print out only the expected limit on the signal strength r.
>
> The output is just the rootfile, higgsCombine$1.Asymptotic.mH120.root, automatically created when run the Tool. The script just change the name and the location of the file into *Datacard/MassPoint_$1/datacardFinal.root*.
>
> This rootfile is a root tree limit that contains the limit values and other bookkeeping information. This tree is used to make the exclusions plots. In our case, we will use a homemade script *MassScan.C*

runLimits.sh: ./runLimits.sh

The code uses recursively ./runPointLimit.sh $1 for all points of mass ($1) of *MassPointList.txt*.

runLimitsLxbatch.sh: bsub -q 8nh runLimitsLxbatch.sh MassPointList.txt

This script is very useful if you have many points of mass and big datacards. It simply executes *runPointLimit.sh $line* in the lxplus queues where $line is one point of mass of *MassPointList.txt*. Since you can divide your *MassPointList.txt* in several .txt files, you can send several jobs at the same time improving the computing limits.

There is an special script:

hwwtools.py

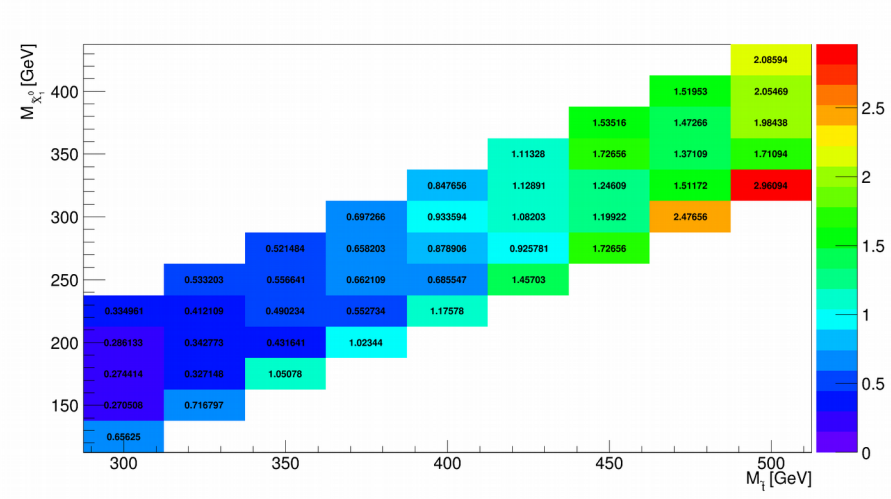
It is designed as a tool to use other codes and help with the execution in Latino Analysis framework. It has at least one important function needed to execute the basis codes: *mkPseudoData.py*, *mkShapesFromHIstos.py*, *mkDatacards.py* . At this moments it is mandatory to keep this code in the repository.

Finally,

MassScan.C: compile and run with root (root -l; .L MassScan.C++ ; MassScan())

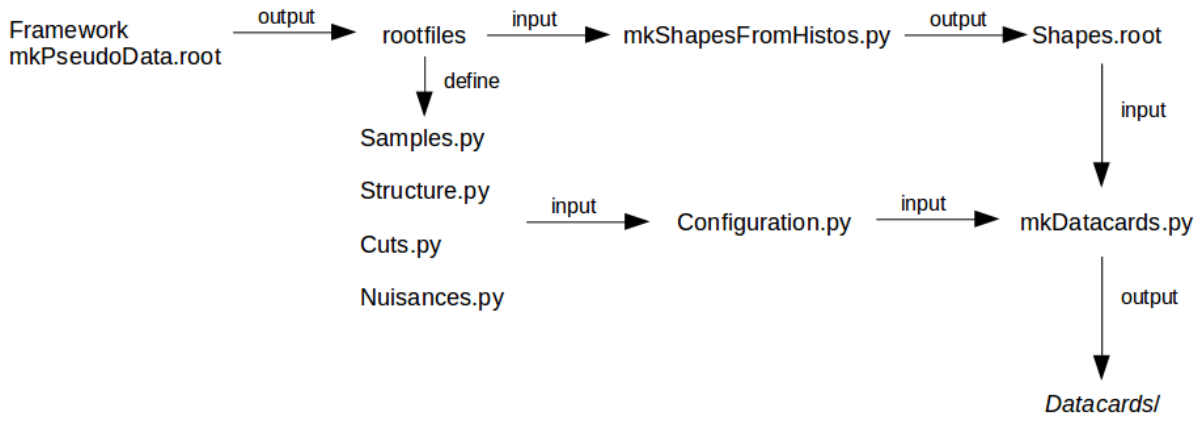
Input: Datacard/MassPoint_$1/datacardFinal.root

Output: A kind of exclusion plot where the quantity represented is the median expected limit (50%)

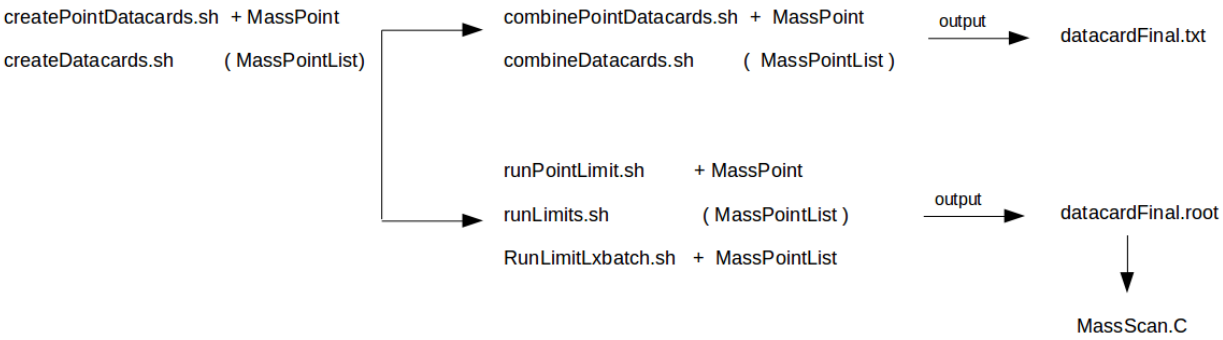


The general work flow producing datacards and computing limits:

Basic:



Specific Point / List of mass:



Note:

The whole module is optimized for one kind of analysis, but all the script inside it can be easily modified for the analyzer needs.

Studying 45 points of (Stop, LSP) mass took this time:

~ 10’ create pseudodata
~ 20’ create datacards and combine them
~ 2 hs computing limits in Lxplus queues. 9 jobs, 5 points of mass in each job.

***Validation Region fit
https://github.com/chazinb/PlotsConfigurations/blob/master/Configurations/T2tt/VR1Fit.txt

*** Datacards latinos questions:
https://docs.google.com/document/d/1rg0qbXX1EiHmbWb1bsWx7xYaNvVPYtkYYDlkC3O5hZ0/edit

[1] https://github.com/latinos/LatinoAnalysis/tree/master/ShapeAnalysis
[2] https://github.com/chazinb/AnalysisCMS/tree/master/combine/T2tt
[3]https://dl.dropboxusercontent.com/u/8772690/HWW/2015/Oct/24/Oct-28_Latino_Framework_Massironi.pdf
[4] https://twiki.cern.ch/twiki/bin/viewauth/CMS/SWGuideHiggsAnalysisCombinedLimit#Computing_limits_with_the_asympt