The Multidimensional (MDFit) tutorial.

Agnieszka Dziurda[1]

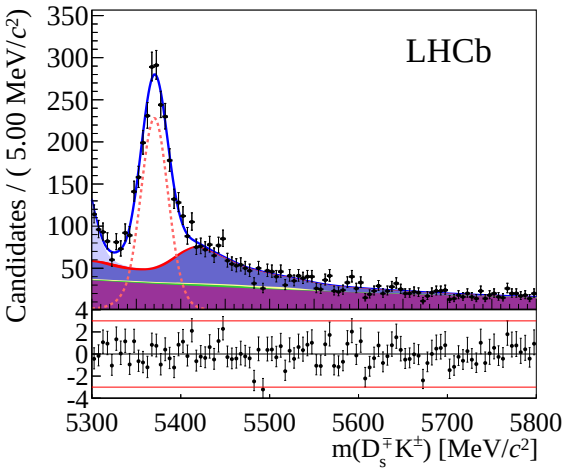
[1]Institute of Nuclear Physics PAN (IFJ), Krakow, Poland

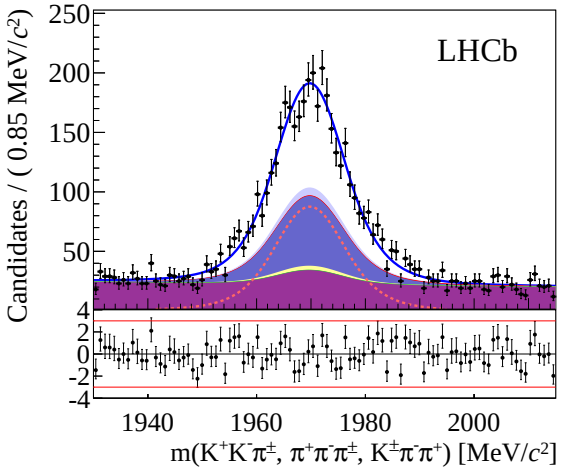
09.07.2015

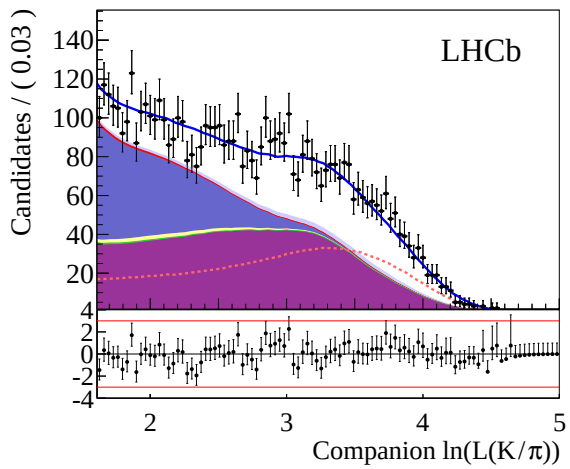
Motivation

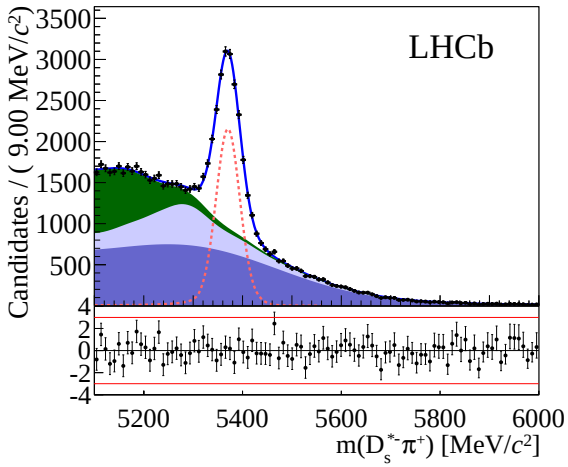
- The purpose of this tutorial is show how to use new version of MDFit.
- Today we will go through main fitting procedure of the MDFit.
- In the `B2DXFitters/data` You can find four config files:
  `Bs2DsKConfigForNominalMassFit.py`
  `Bs2DsPiConfigForNominalMassFit.py`
  `Bs2DsstPiConfigForNominalMassFitOld.py`
  `SignalConfigForFit.py`
  and also .txt files with paths to data/MC files:
  `config_Bs2Dsh2011TDAna_Bs2DsK.txt`
  `config_Bs2Dsh2011TDAna_Bs2DsPi.txt`
  `config_Bs2DsstPi_old.txt`
- In the `B2DXFitters/scripts` are python scripts:
  `prepapreWorkspace.py`
  `fitMDFitter.py`
  `plotMDFitter.py`
  `fitSignal.py`

Motivation









How to get all of these results?

Preparing workspace using prepareWorkspace.py.

Preparing workspace

- The procedure of obtaining entire workspace is quite long, here I will only highlight some useful examples.
- For other steps of tutorial you can pick up ready workspaces from:
  `/afs/cern.ch/work/a/adudziak/public/workspace/tutorial`
  for $B_s^0 \to D_s^- \pi^+$: `work_bsdspi.root`
  for $B_s^0 \to D_s^{\mp} K^{\pm}$: `work_bsdsk.root`
  for $B_s^0 \to D_s^{*-} \pi^+$: `work_bsdsstpi.root`

Preparing workspace

Available options:

- `--debug` : print debug information while processing,
- `--save work.root, -s work.root` : save data sets and PDFs in the work.root files,
- `--initial work.root, -i work.root` : load the workspace work.root and continue obtaining shapes,
- `--workName name` : set the name of loaded workspace work.root,
- `--configName config` : set name of configuration file,
- `--Data` : obtain data sets,
- `--DataBkg` : obtain backgrounds from data
  only used in case of $B^0 \to D^- \pi^+$ as background to $B_s^0 \to D_s^- \pi^+$
  and $B_s^0 \to D_s^- \pi^+$ as background to $B_s^0 \to D_s^{\mp} K^{\pm}$
- `--DataBkgPID` : obtain PIDK shape for backgrounds taken from data,
- `--MC` : obtain backgrounds from MC samples,
- `--MCPID` : obtain PIDK shapes for MC backgrounds (by default it sets –MC to be true)
- `--Signal` : obtain data sets for Signal,
- `--SignalPID` : obtain PIDK shapes for Signal,
- `--Comb` : obtain combinatorial background sample (in case of providing `RooKeysPdf`),
- `--CombPID` : obtain combinatorial PIDK shape,
- `--noRooKeysPdf` : do not obtain `RooKeysPdf` for samples (for debugging)

Preparing workspace - how does it work?

In the config files (for example `data/Bs2DsKConfigForNominalMassFit.py`) you can find:

```
# considered decay mode
configdict["Decay"] = "Bs2DsK"
configdict["CharmModes"] = {"NonRes","PhiPi","KstK","KPiPi","PiPiPi"}
# year of data taking
configdict["YearOfDataTaking"] = {"2011"}
# stripping (necessary in case of PIDK shapes)
configdict["Stripping"] = {"2011":"17"}
# integrated luminosity in each year of data taking (necessary in case of PIDK shapes)
configdict["IntegratedLuminosity"] = {"2011": {"Down": 0.59, "Up": 0.44}}
# file name with paths to MC/data samples
configdict["dataName"]   = "../data/config_Bs2Dsh2011TDAna_Bs2DsK.txt"
```

based on this information necessary labels are created, and the samples from `config_Bs2Dsh2011TDAna_Bs2DsK.txt` are read according to labels.

```
#Bs2DsK PhiPi 2011
/afs/cern.ch/work/a/adudziak/public/Bs2DsKFitTuple/
MergedTree_Bs2DsX_MD_OFFLINE_DsK_PhiPi.root
MergedTree_Bs2DsX_MU_OFFLINE_DsK_PhiPi.root
DecayTree
DecayTree
###
```

```
#Bs2DsPi KstK 2011
/afs/cern.ch/work/a/adudziak/public/Bs2DsKFitTuple/
MergedTree_Bs2DsX_MD_OFFLINE_DsPi_KstK.root
MergedTree_Bs2DsX_MU_OFFLINE_DsPi_KstK.root
DecayTree
DecayTree
###
```

Warning : Please keep in mind that .txt files will be replaced soon by more user friendly python scripts.

Preparing workspaces: example 1

Example of usage, obtaining data sets:

- $B_s^0 \to D_s^{\mp} K^{\pm}$:

  ```
  python prepareWorkspace.py --Data --debug
  --configName Bs2DsKConfigForNominalMassFit --save work_dsk.root
  ```

- According to config file:

```
# considered decay mode
configdict["Decay"] = "Bs2DsK"
configdict["CharmModes"] = {"NonRes","PhiPi","KstK","KPiPi","PiPiPi"}
# year of data taking
configdict["YearOfDataTaking"] = {"2011"}
# stripping (necessary in case of PIDK shapes)
configdict["Stripping"] = {"2011":"17"}
```

  5 $D_s^{\mp}$ modes are obtained, by default separately for magnet polarity, the year of data taking is specified as 2011.

- You can look at the control plots in `scripts/Plot`.
- The data sets and variables are added to `work_dsk.root`:

```
variables
---------
(BDTG,BacCharge,BacP,BacPIDK,BacPT,BeautyMass,BeautyTime,BeautyTimeErr,CharmMass,MistagOS,MistagOS_calib,MistagSS,MistagSS_calib,TagDecOS,TagDecSS,nTrack
arge,tagDecOSnnetKaon,tagDecSSElectron,tagDecSSKaon,tagOmegaComb,tagOmegaOSElectron,tagOmegaOSKaon,tagOmegaOSMuon,tagOmegaOSVtxCharge,tagOmegaOSnnetKaon,

datasets
--------
RooDataSet::dataSetBs2DsK_down_kstk_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,BacCharge,tagDecOSKaon,tagOmegaOSK
DecOSVtxCharge,tagDecSSElectron,tagOmegaSSKaon,tagDecOSMuon,tagOmegaOSVtxCharge,tagOmegaOSElectron,tagOmegaOSnnetKaon,TagDecSS,TagDecOS,MistagSS,MistagOS
mb)
RooDataSet::dataSetBs2DsK_up_kstk_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,BacCharge,tagDecOSKaon,tagOmegaOSKao
cOSVtxCharge,tagDecSSElectron,tagOmegaSSKaon,tagDecOSMuon,tagOmegaOSVtxCharge,tagOmegaOSElectron,tagOmegaOSnnetKaon,TagDecSS,TagDecOS,MistagSS,MistagOS,M
)
```

Preparing workspaces: example 1

- Data sets contain `BasicVariables` defined in `Bs2DsKConfigForNominalMassFit`:

```
# basic variables
configdict["BasicVariables"] = {}
configdict["BasicVariables"]["BeautyMass"]    = { "Range" : [5300,    5800    ], "InputName" : "lab0_MassFitConsD_M"}
configdict["BasicVariables"]["CharmMass"]     = { "Range" : [1930,    2015    ], "InputName" : "lab2_MM"}
configdict["BasicVariables"]["BeautyTime"]    = { "Range" : [0.4,     15.0    ], "InputName" : "lab0_LifetimeFit_ctau"}
configdict["BasicVariables"]["BacP"]          = { "Range" : [3000.0,  650000.0], "InputName" : "lab1_P"}
configdict["BasicVariables"]["BacPT"]         = { "Range" : [400.0,   45000.0 ], "InputName" : "lab1_PT"}
configdict["BasicVariables"]["BacPIDK"]       = { "Range" : [1.61,    5.0     ], "InputName" : "lab1_PIDK"}
configdict["BasicVariables"]["nTracks"]       = { "Range" : [15.0,    1000.0  ], "InputName" : "nTracks"}
configdict["BasicVariables"]["BeautyTimeErr"] = { "Range" : [0.01,    0.1     ], "InputName" : "lab0_LifetimeFit_ctauErr"}
configdict["BasicVariables"]["BacCharge"]     = { "Range" : [-1000.0, 1000.0  ], "InputName" : "lab1_ID"}
configdict["BasicVariables"]["BDTG"]          = { "Range" : [0.3,     1.0     ], "InputName" : "BDTGResponse_1"}
configdict["BasicVariables"]["TagDecOS"]      = { "Range" : [-1.0,    1.0     ], "InputName" : "lab0_TAGDECISION_OS"}
configdict["BasicVariables"]["TagDecSS"]      = { "Range" : [-1.0,    1.0     ], "InputName" : "lab0_SS_nnetKaon_DEC"}
configdict["BasicVariables"]["MistagOS"]      = { "Range" : [ 0.0,    0.5     ], "InputName" : "lab0_TAGOMEGA_OS"}
configdict["BasicVariables"]["MistagSS"]      = { "Range" : [ 0.0,    0.5     ], "InputName" : "lab0_SS_nnetKaon_PROB"}
```

- but also some `AdditionalVariables` defined in `Bs2DsKConfigForNominalMassFit`:

```
# additional variables in data sets
configdict["AdditionalVariables"] = {}
configdict["AdditionalVariables"]["tagOmegaSSKaon"]      =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_SS_Kaon_PROB"}
configdict["AdditionalVariables"]["tagDecSSKaon"]        =  { "Range" : [ -2.0, 2,0 ], "InputName" : "lab0_SS_Kaon_DEC"}
configdict["AdditionalVariables"]["tagOmegaOSMuon"]      =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_OS_Muon_PROB"}
configdict["AdditionalVariables"]["tagDecOSMuon"]        =  { "Range" : [ -2.0, 2,0 ], "InputName" : "lab0_OS_Muon_DEC"}
configdict["AdditionalVariables"]["tagOmegaOSElectron"]  =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_OS_Electron_PROB"}
configdict["AdditionalVariables"]["tagDecSSElectron"]    =  { "Range" : [ -2.0, 2,0 ], "InputName" : "lab0_OS_Electron_DEC"}
configdict["AdditionalVariables"]["tagOmegaOSKaon"]      =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_OS_Kaon_PROB"}
configdict["AdditionalVariables"]["tagDecOSKaon"]        =  { "Range" : [ -2.0, 2,0 ], "InputName" : "lab0_OS_Kaon_DEC"}
configdict["AdditionalVariables"]["tagOmegaOSnnetKaon"]  =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_OS_nnetKaon_PROB"}
configdict["AdditionalVariables"]["tagDecOSnnetKaon"]    =  { "Range" : [ -2.0, 2,0 ], "InputName" : "lab0_OS_nnetKaon_DEC"}
configdict["AdditionalVariables"]["tagOmegaOSVtxCharge"] =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_VtxCharge_PROB"}
configdict["AdditionalVariables"]["tagDecOSVtxCharge"]   =  { "Range" : [ -2.0, 2,0 ], "InputName" : "lab0_VtxCharge_DEC" }
```

Preparing workspaces: example 1

- Data are selected using specified in `Bs2DsKConfigForNominalMassFit` requirements labeled as `configdict["AdditionalCuts"][mode]["Data"]`:

```
# additional cuts applied to data sets
configdict["AdditionalCuts"] = {}
configdict["AdditionalCuts"]["All"]    = { "Data": "lab2_TAU>0",            "MC" : "lab2_TAU>0&&lab1_M>200",
                                           "MCID":True, "MCTRUEID":True, "BKGCAT":True, "DsHypo":True}
configdict["AdditionalCuts"]["KKPi"]   = { "Data": "lab2_FDCHI2_ORIVX > 2", "MC" : "lab2_FDCHI2_ORIVX > 2"}
configdict["AdditionalCuts"]["KPiPi"]  = { "Data": "lab2_FDCHI2_ORIVX > 9", "MC" : "lab2_FDCHI2_ORIVX > 9"}
configdict["AdditionalCuts"]["PiPiPi"] = { "Data": "lab2_FDCHI2_ORIVX > 9", "MC" : "lab2_FDCHI2_ORIVX > 9"}

# children prefixes used in MCID, MCTRUEID, BKGCAT cuts
# order of particles: KKPi, KPiPi, PiPiPi
configdict["DsChildrenPrefix"] = {"Child1":"lab5","Child2":"lab4","Child3": "lab3"}
```

where:

cuts defined as `All` are applied to all 10 data samples,
cuts defined as `KKPi` are applied to [NonRes,KstK,PhiPi],
cuts defined as `KPiPi` (`PiPiPi`) are applied to KPiPi (PiPiPi).

Preparing workspaces: example 2

Example of usage, obtaining data sets:

- $B_s^0 \to D_s^{*-}\pi^+$:

```
python prepareWorkspace.py --Data --debug
--configName Bs2DsstPiConfigForNominalMassFit --save work_dsstpi.root
```

- According to config file:

```
# considered decay mode
configdict["Decay"] = "Bs2DsstPi"
configdict["CharmModes"] = {"KKPi"}
# year of data taking
configdict["YearOfDataTaking"] = {"2012","2011"}
# stripping (necessary in case of PIDK shapes)
configdict["Stripping"] = {"2012":"20", "2011":"20r1"}
```

- You can look at the control plots in `scripts/Plot`.
- The data sets and variables are added to `work_dsstpi.root`:

```
variables
---------
(BDTG,BacCharge,BacP,BacPIDK,BacPT,BeautyMass,BeautyTime,BeautyTimeErr,CharmMass,MistagOS,MistagOS_calib,MistagSS,MistagSS_calib,TagDecOS,TagDecSS,nTrack
s,tagDecComb,tagOmegaComb)

datasets
--------
RooDataSet::dataSetBs2DsstPi_down_kkpi_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,BacCharge,TagDecSS,TagDecOS,Mis
tagSS,MistagOS,MistagSS_calib,MistagOS_calib,tagDecComb,tagOmegaComb)
RooDataSet::dataSetBs2DsstPi_up_kkpi_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,BacCharge,TagDecSS,TagDecOS,Mista
gSS,MistagOS,MistagSS_calib,MistagOS_calib,tagDecComb,tagOmegaComb)
RooDataSet::dataSetBs2DsstPi_down_kkpi_2012(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,BacCharge,TagDecSS,TagDecOS,Mis
tagSS,MistagOS,MistagSS_calib,MistagOS_calib,tagDecComb,tagOmegaComb)
RooDataSet::dataSetBs2DsstPi_up_kkpi_2012(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,BacCharge,TagDecSS,TagDecOS,Mista
gSS,MistagOS,MistagSS_calib,MistagOS_calib,tagDecComb,tagOmegaComb)
```

Preparing workspaces: example 2

- The additional variables are not specified.
- However, in addition to `BasicVariables`, to the data sets are added calibrated mistags as well as combined tagging decision and mistag:
  `MistagSS_calib,MistagOS_calib,tagDecComb,tagOmegaComb`
- The calibration of mistag is done in `Bs2DsstPiConfigForNominalMassFit` by:

```
# tagging calibration
configdict["TaggingCalibration"] = {}
configdict["TaggingCalibration"]["OS"] = {"p0": 0.3834, "p1": 0.9720, "average": 0.3813}
configdict["TaggingCalibration"]["SS"] = {"p0": 0.4244, "p1": 1.2180, "average": 0.4097}
```

Preparing workspace: example 3

Example of usage, obtaining MC samples

- $B_s^0 \to D_s^{*-}\pi^+$:

```
python prepareWorkspace.py --MC --debug
--configName Bs2DsstPiConfigForNominalMassFit --save work_dsstpi.root
```

- the PDFs are weighted according to integrated luminosity for year and magnet polarity.

```
# integrated luminosity in each year of data taking (necessary in case of PIDK shapes)
configdict["IntegratedLuminosity"] = {"2011": {"Down": 0.59, "Up": 0.44}}
```

- the `RooKeysPdfs` are obtained separately for:
  year of data taking,
  beauty and charm meson masses.
- You can configure `RooKeysPdf` in `config_Bs2DsstPi_old.txt`:

```
#MC FileName KKPi MD 2011
{"Mode":"Bs2DsRho",
 "FileName":"/afs/cern.ch/work/a/abertoli/public/DsstrPi/bdt_dev/Filter_Bs2DsstPi_Bs2Dsrho_dw.root",
 "TreeName":"tuple;1"}
{"Mode":"Bs2DsstRho",
 "FileName":"/afs/cern.ch/work/a/abertoli/public/DsstrPi/bdt_dev/Filter_Bs2DsstPi_Bs2Dsstrho_dw.root",
 "TreeName":"tuple;1",
 "Smooth":3.5}
###
```

Preparing workspace: example 4

- MC data sets are cut according to requirements in `configdict["AdditionalCuts"][mode]["MC"]` using the same logic as for data.
- MC data sets contain all `BasicVariables`.
- You can look at the obtained shapes in `Plot/data_with_template_*.pdf`
- You can check correlations between variables: `Plot/corr_*.pdf`
- The output of procedure is:

```
variables
---------
(BDTG,BacP,BacPIDK,BacPT,BeautyMass,BeautyTime,BeautyTimeErr,CharmMass,MistagOS,MistagSS,TagDecOS,TagDecSS,nTracks)

p.d.f.s
-------
RooKeysPdf::PhysBkgBs2DsRhoPdf_m_both_2011[ x=BeautyMass ] = 0.00078788
RooKeysPdf::PhysBkgBs2DsRhoPdf_m_both_2011_Ds[ x=CharmMass ] = 0.0195995
RooKeysPdf::PhysBkgBs2DsRhoPdf_m_both_2012[ x=BeautyMass ] = 0.00078788
RooKeysPdf::PhysBkgBs2DsRhoPdf_m_both_2012_Ds[ x=CharmMass ] = 0.0195995
RooKeysPdf::PhysBkgBs2DsstRhoPdf_m_both_2011[ x=BeautyMass ] = 0.00198886
RooKeysPdf::PhysBkgBs2DsstRhoPdf_m_both_2011_Ds[ x=CharmMass ] = 0.0196867
RooKeysPdf::PhysBkgBs2DsstRhoPdf_m_both_2012[ x=BeautyMass ] = 0.00198886
RooKeysPdf::PhysBkgBs2DsstRhoPdf_m_both_2012_Ds[ x=CharmMass ] = 0.0196867

datasets
--------
RooDataSet::dataSetMC_Bs2DsRho_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsstRho_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsRho_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsstRho_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsRho_up_2012(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsstRho_up_2012(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsRho_down_2012(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsstRho_down_2012(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
```

Preparing workspace: example 5

Example of usage, obtaining PIDK shapes for MC samples

- $B_s^0 \to D_s^- \pi^+$:

```
python prepareWorkspace.py --MCPID --debug
--configName Bs2DsPiConfigForNominalMassFit --save work_dsstpi.root
--noRooKeysPdf
```

- firstly MC samples are obtained, then PIDK shapes.
- I set `--noRooKeysPdf` due to avoiding long computing of `RooKeysPdfs` for invariant masses. Please keep in mind that usually you want to obtain these PDFs as well!
- in the `Bs2DsPiConfigForNominalMassFit` are defined calibration samples as input to procedure:

```
#weighting for PID templates
configdict["ObtainPIDTemplates"] = { "Variables":["BacPT","nTracks"], "Bins":[20,20] }
configdict["Calibrations"] = {}
configdict["Calibrations"]["Pion"]   = { "FileNameUp":"/afs/cern.ch/work/a/adudziak/public/workspace/CalibrationSamples/CalibDst_Up_Pi_PID0_Str17.root",
                                         "FileNameDown":"/afs/cern.ch/work/a/adudziak/public/workspace/CalibrationSamples/CalibDst_Down_Pi_PID0_Str17.root",
                                         "WorkName":"RSDStCalib"}
configdict["Calibrations"]["Kaon"]   = { "FileNameUp":"/afs/cern.ch/work/a/adudziak/public/workspace/CalibrationSamples/CalibDst_Up_K_PID0_Str17.root",
                                         "FileNameDown":"/afs/cern.ch/work/a/adudziak/public/workspace/CalibrationSamples/CalibDst_Down_K_PID0_Str17.root",
                                         "WorkName":"RSDStCalib"}
configdict["Calibrations"]["Combinatorial"]   = { "FileNameUp":"/afs/cern.ch/work/a/adudziak/public/workspace/work_Comb_DsPi_5358.root",
                                                  "FileNameDown":"/afs/cern.ch/work/a/adudziak/public/workspace/work_Comb_DsPi_5358.root",
                                                  "WorkName":"workspace"}
```

- These samples are already cut for PIDK on the bachelor.
- The 2D weighting is done using `BacPT` and `nTracks` variables with 20 bins in each directory.

Preparing workspace: example 5

Output:

```
variables
---------
(BDTG,BacP,BacPIDK,BacPT,BeautyMass,BeautyTime,BeautyTimeErr,CharmMass,MistagOS,MistagSS,TagDecOS,TagDecSS,nTracks)

p.d.f.s
-------
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bd2DPi_down_2011[ x=BacPIDK ] = 48044.4
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bd2DPi_up_2011[ x=BacPIDK ] = 70340.7
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bs2DsK_down_2011[ x=BacPIDK ] = 22155.3
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bs2DsK_up_2011[ x=BacPIDK ] = 34129
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bs2DsRho_down_2011[ x=BacPIDK ] = 19859.9
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bs2DsRho_up_2011[ x=BacPIDK ] = 27229.6
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bs2DsstPi_down_2011[ x=BacPIDK ] = 43477.1
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Bs2DsstPi_up_2011[ x=BacPIDK ] = 66192.8
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Lb2LcPi_down_2011[ x=BacPIDK ] = 39017.9
RooBinned1DQuinticBase<RooAbsPdf>::PIDKShape_Lb2LcPi_up_2011[ x=BacPIDK ] = 57636.7

datasets
--------
RooDataSet::dataSetMC_Bd2DPi_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsRho_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsstPi_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Lb2LcPi_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsK_up_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bd2DPi_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsRho_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsstPi_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Lb2LcPi_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
RooDataSet::dataSetMC_Bs2DsK_down_2011(BeautyMass,CharmMass,BacPIDK,BeautyTime,BeautyTimeErr,BDTG,BacP,BacPT,nTracks,TagDecSS,TagDecOS,MistagSS,MistagOS)
```

- You can look at the obtained shapes: `Plot/data_with_template_BacPIDK_*.pdf`

Preparing workspaces

Available options

- $B_s^0 \to D_s^{\mp} K^{\pm}$:

  `--Data`, `--DataBkg`, `--DataBkgPID`, `--MC` `--MCPID`, `--Signal`, `--SignalPID`, `--CombPID`

- $B_s^0 \to D_s^{-} \pi^{+}$:

  `--Data`, `--DataBkg`, `--DataBkgPID`, `--MC` `--MCPID`, `--Signal`, `--SignalPID`, `--CombPID`

- $B_s^0 \to D_s^{*-} \pi^{+}$:

  `--Data`, `--MC`, `--MCPID`, `--Signal`, `--SignalPID`, `--Comb`

You can play more on your own!

Preparing workspace

For completeness:

- In case of $B_s^0 \to D_s^- \pi^+$ and $B_s^0 \to D_s^{\mp} K^{\pm}$ you can look at `scripts/Csh/prepare_workspace.csh` and execute it from `script` directory by:

  ```
  source Csh/prepare_workspace.csh work_dsk.root
     Bs2DsKConfigForNominalMassFit > & log_work_dsk.txt&
  ```

  file content:

  ```
  python prepareWorkspace.py --Data --debug -s $1 --configName $2
  python prepareWorkspace.py --DataBkgPID --debug -i $1 -s $1 --configName $2
  python prepareWorkspace.py --MCPID --debug -i $1 -s $1 --configName $2
  python prepareWorkspace.py --SignalPID --debug -i $1 -s $1 --configName $2
  python prepareWorkspace.py --CombPID --debug -i $1 -s $1 --configName $2
  ```

- Warning : Obtaining entire workspaces for $B_s^0 \to D_s^- \pi^+$ and $B_s^0 \to D_s^{\mp} K^{\pm}$ takes some time due to `RooKeyPdf`, so for tutorial I don't recommend to try go through entire procedure.

Fitting using runMDFitter.py.

Fitting

- You can pick up ready workspaces from:
  `/afs/cern.ch/work/a/adudziak/public/workspace/tutorial`
  for $B_s^0 \to D_s^- \pi^+$: `work_bsdspi.root`
  for $B_s^0 \to D_s^\mp K^\pm$: `work_bsdsk.root`
  for $B_s^0 \to D_s^{*-} \pi^+$: `work_bsdsstpi.root`
- For running MDFit we use runMDFitter.py script.
- Warning : Please be aware what is content of workspaces:
  - $B_s^0 \to D_s^- \pi^+$ and $B_s^0 \to D_s^\mp K^\pm$:
    2011 data,
    5 Ds modes = {NonRes, KstK, PhiPi, KPiPi, PiPiPi}
    both magnet polarities, saved separately
    3D fit
  - $B_s^0 \to D_s^{*-} \pi^+$:
    2011 and 2012 data,
    1 Ds mode KKPi.
    both magnet polarities, saved separately
    only fit to Bs mass
  - You can perform your fits only within above conditions.

Fitting: options

- `--debug` : print debug information while processing,
- `--save WS_MDFit_Results.root` : save results to `WS_MDFit_Results.root` file
- `--sample both` : choose polarity of sample,
  possibilities sample ={ down, up, both },
  `--sample both` sets simultaneous fit to magnet polarities.
- `--merge` : merge magnet polarities, sets combined fit to magnet polarities,
- `--mode all` : set charm meson final states,
  possibilities mode ={ nonres, phipi, kstk, kkpi, kpipi, pipipi, all, 3modeskkpi },
- `--year 2011` : set year of data taking,
  possibilities year ={ 2011, 2012, run1 },
  `--year run1` sets simultaneous fit to year of data taking.
- `--dim` : set number of dimension of the fit
  possibilities dim ={ 1, 2, 3 },
- `--fileName work.root` : workspace with your data and PDFs.
- `--workName workspace` : name of workspace.
- `--fileData work2.root` : use in case you want to load data from different workspace than PDFs.
- `--sweight` : compute sWeights.
- `--sweightName` : set name of output file with sWeights.
- `--configName config` : set name of configuration file,

Fitting: settings

Access to the signal shape via config file:

```
# Bs signal shapes
configdict["BsSignalShape"] = {}
configdict["BsSignalShape"]["type"]    = "DoubleCrystalBall"
configdict["BsSignalShape"]["mean"]    = {"All":5367.51}
configdict["BsSignalShape"]["sigma1"]  = {"2011": {"NonRes":1.0717e+01,  "PhiPi":1.1235e+01,  "KstK":1.0772e+01,  "KPiPi":1.1268e+01,  "PiPiPi":1.1391e+01}, "Fixed":True}
configdict["BsSignalShape"]["sigma2"]  = {"2011": {"NonRes":1.6005e+01,  "PhiPi":1.7031e+01,  "KstK":1.5339e+01,  "KPiPi":1.9408e+01,  "PiPiPi":1.7647e+01}, "Fixed":True}
configdict["BsSignalShape"]["alpha1"]  = {"2011": {"NonRes":2.2118e+00,  "PhiPi":2.2144e+00,  "KstK":2.0480e+00,  "KPiPi":2.3954e+00,  "PiPiPi":2.0930e+00}, "Fixed":True}
configdict["BsSignalShape"]["alpha2"]  = {"2011": {"NonRes":-2.4185e+00, "PhiPi":-2.1918e+00, "KstK":-2.0291e+00, "KPiPi":-3.4196e+00, "PiPiPi":-2.3295e+00}, "Fixed":True}
configdict["BsSignalShape"]["n1"]      = {"2011": {"NonRes":1.0019e+00,  "PhiPi":1.1193e+00,  "KstK":1.2137e+00,  "KPiPi":9.8202e-01,  "PiPiPi":1.2674e+00}, "Fixed":True}
configdict["BsSignalShape"]["n2"]      = {"2011": {"NonRes":3.1469e+00,  "PhiPi":3.6097e+00,  "KstK":6.5735e+00,  "KPiPi":5.2237e-01,  "PiPiPi":4.0195e+00}, "Fixed":True}
configdict["BsSignalShape"]["frac"]    = {"2011": {"NonRes":6.1755e-01,  "PhiPi":7.0166e-01,  "KstK":5.8012e-01,  "KPiPi":7.8103e-01,  "PiPiPi":7.0398e-01}, "Fixed":True}
configdict["BsSignalShape"]["scaleSigma"] = { "2011": {"frac1": 1.22, "frac2":1.28}}
```

You can:

- set shape parametrization separately for each fitted D_s final state and data year,
- fix or float parameters using "Fixed" variable,
- if necessary scale widths of double Crystal Ball.
- parametrization for beauty meson mass via `configdict["BsSignalShape"]`,
- parametrization for charm meson mass via `configdict["DsSignalShape"]`

In the package so far the supported shapes are:

- double Crystal Ball (`DoubleCrystalBall`)
- double Gaussian (`DoubleGaussian`)
- double Crystal Ball with fixed widths but common for both of them scaling parameter (`DoubleCrystalBallWithWidthRatio`)

Fitting: settings

Access to the combinatorial shape via config file:

```
# combinatorial background
configdict["BsCombinatorialShape"] = {}
configdict["BsCombinatorialShape"]["type"] = "ExponentialPlusGauss"
configdict["BsCombinatorialShape"]["cB"]         = {"2012": {"KKPi":-0.00354546},  "Fixed":False}
configdict["BsCombinatorialShape"]["fracComb"]   = {"2012": {"KKPi":0.3}.          "Fixed":True}
configdict["BsCombinatorialShape"]["meanComb"]   = {"2012": {"KKPi":5299.22},      "Fixed":True}
configdict["BsCombinatorialShape"]["widthComb"]  = {"2012": {"KKPi":182.448},     "Fixed":True}
```

You can:

- set shape separately for each fitted D_s final state and data year,
- fix or float parameters using "Fixed" variable,
- parametrization for beauty meson mass via `configdict["BsCombinatorialShape"]`,
- parametrization for charm meson mass via `configdict["DsCombinatorialShape"]`
- parametrization for bachelor PIDK via `configdict["PIDKCombinatorialShape"]`

In the package so far the supported shapes are:

- double Exponential (`DoubleExponential`)
- single Exponential (`Exponential`)
- Exponential plus Signal (`ExponentialPlusSignal`)
- Exponential plus Gaussian (`ExponentialPlusGauss`)
- RooKeysPdf (`RooKeysPdf`)
- separate treatment for PIDK shape.

Fitting: settings

Access to yields via config file:

```
#expected yields
configdict["Yields"] = {}
configdict["Yields"]["Bd2DPi"]          = {"2011": { "NonRes":374.0,   "PhiPi":6.0,     "KstK":93.0,    "KPiPi":30.0,    "PiPiPi":0.0} ,     "Fixed":True}
configdict["Yields"]["Lb2LcPi"]         = {"2011": { "NonRes":290.0,   "PhiPi":36.0,    "KstK":69.0,    "KPiPi":1.0,     "PiPiPi":0.0} ,     "Fixed":True}
configdict["Yields"]["Bs2DsK"]          = {"2011": { "NonRes":40.0,    "PhiPi":47.0,    "KstK":40.0,    "KPiPi":8.0,     "PiPiPi":21.0} ,    "Fixed":True}
configdict["Yields"]["Bs2DsDsstPiRho"]  = {"2011": { "NonRes":100.0,   "PhiPi":100.0,   "KstK":100.0,   "KPiPi":100.0,   "PiPiPi":100.0},    "Fixed":False}
configdict["Yields"]["CombBkg"]         = {"2011": { "NonRes":10000.0, "PhiPi":10000.0, "KstK":10000.0, "KPiPi":10000.0, "PiPiPi":10000.0},  "Fixed":False}
configdict["Yields"]["Signal"]          = {"2011": { "NonRes":10000.0, "PhiPi":10000.0, "KstK":10000.0, "KPiPi":10000.0, "PiPiPi":10000.0} , "Fixed":False}
```

You can:

- set yields separately for each fitted D_s final state and data year,
- fix or float yields using "Fixed" variable,
- add necessary parameters for your background description.
- specify your background description in dedicated namespace `Bs2Dsh2011TDAnaModels`, `Bs2DssthModels`, `Bs2DshDsHHHPi0Models`. Usually it is not simply adding modes together.

```
#
configdict["AdditionalParameters"] = {}
configdict["AdditionalParameters"]["g1_f1_frac"] = { "CentralValue":0.5, "Range":[0.0,1.0], "Fixed":False}
configdict["AdditionalParameters"]["g1_f2_frac"] = { "CentralValue":0.5, "Range":[0.0,1.0], "Fixed":False}
```

Fitting: setting

Special treatment of the $B_d^0 \to D_s^- X^+$ backgrounds:

```
#Bd2Dsh background
#shape for BeautyMass, for CharmMass as well as BacPIDK taken by default the same as signal
configdict["Bd2Ds(st)XShape"] = {}
configdict["Bd2Ds(st)XShape"]["type"]    = "ShiftedSignal"
configdict["Bd2Ds(st)XShape"]["name"]    = "Bd2DsPi"
configdict["Bd2Ds(st)XShape"]["scaleSigma"] = { "2011": {"frac1": 1.00808721452, "frac2":1.0386867331}}
```

- Usually the shape is taken as signal shifted by B_s - B = 86.8 MeV downwards.
- The widths are scaled by Signal/background ratio from MC.
- For now only `ShiftedSignal` available.
- Merging treatment from the $B_s^0 \to D_s^{*-} X^+$ measurement is in progress.

Fitting: example 1

- Default run condition for $B_s^0 \to D_s^{\mp} K^{\pm}$:
  `python runMDFitter.py --fileName work_bsdsk.root --merge --sample both --mode all --year 2011 --configName Bs2DsKConfigForNominalMassFit --debug --dim 3 --sweight`

  You can try:
  `--dim 1`, `--dim 2`, `--dim 3`,
  `--sample down`, `--sample up`, (without `--merge`)
  `--mode nonres`, `--mode kstk`, `--mode phipi`, `--mode kpipi`, `--mode pipipi`,
  `--mode 3modeskkpi`,
  You can try play with fixed/floated parameters in `Bs2DsKConfigForNominalMassFit`
  By default:
  fit results are saved in file: `WS_MDFit_Results.root`
  you can change name by adding `--save yourname.root`
  NTuple with sWeights and observables from data set: `sWeights_Results.root`
  you can change name by adding `--sweightName yourname.root`

Fitting: example 2

- Default run condition for $B_s^0 \to D_s^- \pi^+$:
  `python runMDFitter.py --fileName work_bsdspi.root --merge --sample both --mode all --year 2011 --configName Bs2DsPiConfigForNominalMassFit --debug --dim 3 --sweight`

  You can try:
  - `--dim 1`, `--dim 2`, `--dim 3`,
  - `--sample down`, `--sample up`, (without `--merge`)
  - `--mode nonres`, `--mode kstk`, `--mode phipi`, `--mode kpipi`, `--mode pipipi`, `--mode 3modeskkpi`,

  You can try play with fixed/floated parameters in `Bs2DsPiConfigForNominalMassFit`
  By default:
  - fit results are saved in file: `WS_MDFit_Results.root`
    - you can change name by adding `--save yourname.root`
  - NTuple with sWeights and observables from data set: `sWeights_Results.root`
    - you can change name by adding `--sweightName yourname.root`

Fitting: example 3

- Default run condition for $B_s^0 \to D_s^{*-}\pi^+$:
  `python runMDFitter.py --fileName work_bsdsstpi.root --merge --sample both --mode kkpi --year run1 --configName Bs2DsstPiConfigForNominalMassFitOld --debug --dim 1 --sweight`

  You can try:
  - `--sample down`, `--sample up`, (without `--merge`)
  - `--sample both` (without `--merge`) - simultaneous fit to magnet polarities.
  - `--year 2011`, `--year 2012`

  You can try play with fixed/floated parameters in
  - `Bs2DsstPiConfigForNominalMassFitOld`

  By default:
  - fit results are saved in file: `WS_MDFit_Results.root`
    - you can change name by adding `--save yourname.root`
  - NTuple with sWeights and observables from data set: `sWeights_Results.root`
    - you can change name by adding `--sweightName yourname.root`

Plotting obtained results using plotMDFitter.py.

Fitting: options

- `--debug` : print debug information while processing,
- `--sample both` : choose polarity of sample,
  possibilities sample ={ down, up, both },
  `--sample both` sets simultaneous fit to magnet polarities.
- `--merge` : merge magnet polarities, sets combined fit to magnet polarities,
- `--mode all` : set charm meson final states,
  possibilities mode ={ nonres, phipi, kstk, kkpi, kpipi, pipipi, all, 3modeskkpi },
- `--year 2011` : set year of data taking,
  possibilities year ={ 2011, 2012, run1 },
  `--year run1` sets simultaneous fit to year of data taking.
- `--dim` : set number of dimension of the fit
  possibilities dim ={ 1, 2, 3 },
- `--log, --logscale` : set logarithmic scale to plot,
- `--configName config` : set name of configuration file,
- `--legend` : plot legend in the canvas.
- `--suffix name` : set suffix to output name.
- `--bin 100` : set number of bins.
- `--var varNamee` : variable to plot.

Plotting: settings

You can configure plotting conditions via config file:

```
from ROOT import *
configdict["PlotSettings"] = {}
configdict["PlotSettings"]["components"] = ["Sig", "CombBkg", "Bd2DPi", "Lb2LcPi", "Bs2DsDsstPiRho", "Bs2DsK"]
configdict["PlotSettings"]["colors"] = [kRed-7, kBlue-6, kOrange, kRed, kBlue-10, kGreen+3]
```

where:

- the first has to be signal,
- then the backgrounds are plotted in order of writing,

Some more advanced example for $B_s^0 \to D_s^{\mp} K^{\pm}$ is shown below, where:

- `"EPDF"` contains extended PDFs, which are used to build full EPDF,
- `"PDF"` are contributions to plot,
- `"Legend"` are contributions written in the legend.

```
configdict["PlotSettings"] = {}
configdict["PlotSettings"]["components"] = { "EPDF": ["Sig", "CombBkg", "Lb2LcK", "Lb2LcPi", "Bd2DK", "Bd2DPi","BsLb2DsDsstPPiRho", "Bs2DsDsstKKst"],
                                             "PDF":  ["Sig", "CombBkg", "Lb2LcK", "Lb2LcPi", "Lb2DsDsstP", "Bs2DsDsstPiRho", "Bd2DK", "Bd2DPi","Bs2DsDsstKKst"],
                                             "Legend": ["Sig", "CombBkg", "Lb2LcKPi", "Lb2DsDsstP", "Bs2DsDsstPiRho", "Bd2DKPi","Bs2DsDsstKKst"]}
configdict["PlotSettings"]["colors"] = { "PDF": [kRed-7, kMagenta-2, kGreen-3, kGreen-3, kYellow-9, kBlue-6, kRed, kRed, kBlue-10],
                                         "Legend": [kRed-7, kMagenta-2, kGreen-3, kYellow-9, kBlue-6, kRed, kBlue-10]}
```

Plotting: settings

You can configure plotting conditions via config file:

```
configdict["LegendSettings"] = {}
configdict["LegendSettings"]["BeautyMass"] = {"Position":[0.53, 0.45, 0.90, 0.91], "TextSize": 0.05, "LHCbText":[0.35,0.9], "ScaleYSize":2.5}
configdict["LegendSettings"]["CharmMass"]  = {"Position":[0.20, 0.69, 0.93, 0.93], "TextSize": 0.05, "LHCbText":[0.8,0.66],
                                              "ScaleYSize":1.7, "SetLegendColumns":2, "LHCbTextSize":0.075 }
configdict["LegendSettings"]["BacPIDK"]    = {"Position":[0.53, 0.45, 0.90, 0.91], "TextSize": 0.05, "LHCbText":[0.35,0.9], "ScaleYSize":1.2}
```

where you can specify:

- position of legend for each variable separately:
  `configdict["LegendSettings"][variable]["Position"]`
- text size in the legend:
  `configdict["LegendSettings"][variable]["TextSize"]`
- position of LHCb text:
  `configdict["LegendSettings"][variable]["LHCbText"]`
- text size of LHCb text,
  `configdict["LegendSettings"][variable]["LHCbTextSize"]`
- rescale y axis of plot if needed
  `configdict["LegendSettings"][variable]["ScaleYSize"]`

Plotting: example 1

Most options have to be specified the same as over fitting.

- Fitting: $B_s^0 \to D_s^{\mp} K^{\pm}$:

```
python runMDFitter.py --fileName work_bsdsk.root --merge --sample both --mode
all --year 2011 --configName Bs2DsKConfigForNominalMassFit --debug --dim 3
--sweight
```

- It is 3 dimensional simultaneous fit to Ds final states, combined magnet polarity and one 2011 year of data taking so possible plotting options are:

```
python plotMDFitter.py WS_Mass_DsK.root --year 2011 --merge --sample both --dim
3 --configName Bs2DsKConfigForNominalMassFit --mode all --var BeautyMass
```

where:
`WS_Mass_DsK.root` is the result saved from previous step,
`--mode` can vary among subsamples i.e.
mode ={ nonres, phipi, kstk, kpipi, pipipi, all },
`--var` can vary among observables i.e. `BeautyMass`, `CharmMass`, `BacPIDK`.

Plotting: example 2

Most options have to be specified the same as over fitting.

- Fitting: $B_s^0 \to D_s^- \pi^+$:

```
python runMDFitter.py --fileName work_bsdspi.root --merge --sample both --mode
all --year 2011 --configName Bs2DsPiConfigForNominalMassFit --debug --dim 3
--sweight
```

- It is 3 dimensional simultaneous fit to Ds final states, combined magnet polarity and one 2011 year of data taking so possible plotting options are:

```
python plotMDFitter.py WS_Mass_DsPi.root --year 2011 --merge --sample both
--dim 3 --configName Bs2DsPiConfigForNominalMassFit --mode all --var
BeautyMass
```

where:
`WS_Mass_DsPi.root` is the result saved from previous step,
`--mode` can vary among subsamples i.e.
mode ={ nonres, phipi, kstk, kpipi, pipipi, all },
`--var` can vary among observables i.e. `BeautyMass`, `CharmMass`, `BacPIDK`.

Plotting: example 3

Most options have to be specified the same as over fitting.

- Fitting: $B_s^0 \to D_s^{*-}\pi^+$:

```
python runMDFitter.py --fileName work_bsdsstpi.root --merge --sample both
--mode kkpi --year run1 --configName Bs2DsstPiConfigForNominalMassFitOld
--debug --dim 1 --sweight
```

- It is 1 dimensional simultaneous fit to year of data taking, with combined magnet polarity and one Ds final state KKPi, so possible plotting options are:

```
python plotMDFitter.py WS_Mass_DsstPi.root --year run1 --merge --sample both
--dim 1 --configName Bs2DsstPiConfigForNominalMassFitOld --mode kkpi
```

where:
`WS_Mass_DsstPi.root` is the result saved from previous step,
`--year` can vary among subsamples i.e. year $=\{$ 2011, 2012, run1$\}$

Plotting: example 4

Most options have to be specified the same as over fitting.

- Fitting: $B_s^0 \to D_s^{*-}\pi^+$ simultaneously to magnet polarities and year of data taking:

```
python runMDFitter.py --fileName work_bsdsstpi.root --sample both --mode kkpi
--year run1 --configName Bs2DsstPiConfigForNominalMassFitOld --debug --dim 1
--sweight
```

- It is 1 dimensional simultaneous fit to year of data taking and magnet polarities, with one Ds final state KKPi, so possible plotting options are:

```
python plotMDFitter.py WS_Mass_DsstPi.root --year run1 --sample both --dim 1
--configName Bs2DsstPiConfigForNominalMassFitOld --mode kkpi
```

where:

`WS_Mass_DsstPi.root` is the result saved from previous step,
`--year` can vary among subsamples i.e. year $=\{$ 2011, 2012, run1$\}$
`--sample` can vary among subsamples i.e. sample $=\{$ up, down, both$\}$

Thank You