Recent improvements and coming updates to the B2DXFitters package

Agnieszka Dziurda[1]

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

09.07.2015

Motivation

This presentation is sponsored by sentence:
"A few years later you should be able to do better!".

- The package had to be cleaned up after last minute cross-checks in the process of publishing the $B_s^0 \to D_s^{\mp} K^{\pm}$ measurement.
- The review of existence code has been done.
- New version is much more general and flexible, hopefully also more user friendly.

Motivation

Decays recently included in the package:

$B_s^0 \to D_s^{\mp} K^{\pm}$,
$D_s \to \phi\pi, K^* K, NonRes$
$D_s \to K\pi\pi, \pi\pi\pi$

$B_s^0 \to D_s^{-} \pi^{+}$,
$D_s \to \phi\pi, K^* K, NonRes$
$D_s \to K\pi\pi, \pi\pi\pi$

$B_s^0 \to D_s^{\mp} K^{\pm}$
$D_s \to KK\pi\pi^0$

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

$B^0 \to D^{-} \pi^{+}$
$D^- \to K^- \pi^+ \pi^-$

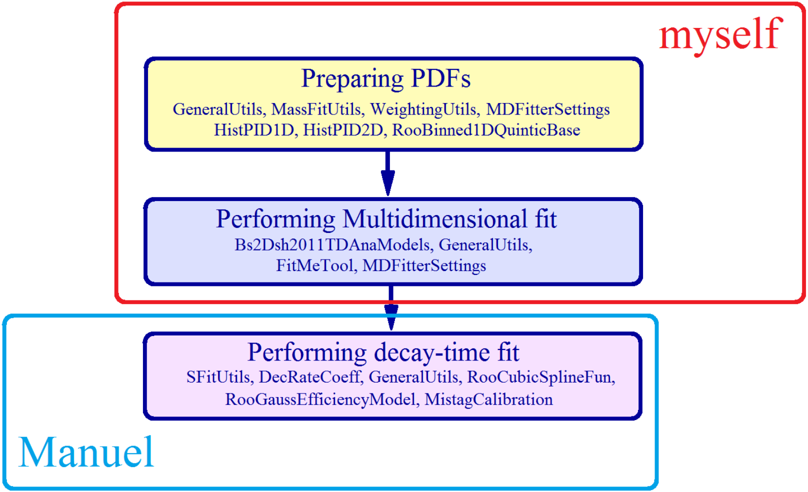
$B_s^0 \to D_s^{*\mp} K^{\pm}$
$D_s \to KK\pi$

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

B2DXFitters:
has to take into account differences between decays,
has to be flexible for different running conditions.

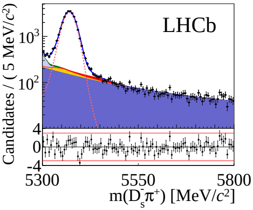
B2DXFitters package

Main fitting procedure.

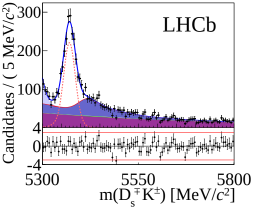

myself
Preparing PDFs
GeneralUtils, MassFitUtils, WeightingUtils, MDFitterSettings
HistPID1D, HistPID2D, RooBinned1DQuinticBase
Performing Multidimensional fit
Bs2Dsh2011TDAnaModels, GeneralUtils,
FitMeTool, MDFitterSettings
Performing decay-time fit
SFitUtils, DecRateCoeff, GeneralUtils, RooCubicSplineFun,
RooGaussEfficiencyModel, MistagCalibration
Manuel

$D_s h$ mass

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



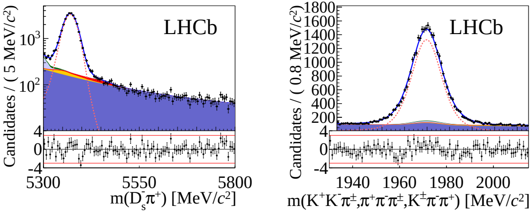
$B_s^0 \to D_s^\mp K^\pm$



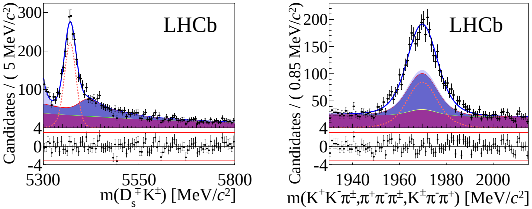
LHCb-PAPER-2014-038

$D_s h$ mass $KK\pi, K\pi\pi, \pi\pi\pi$ mass

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



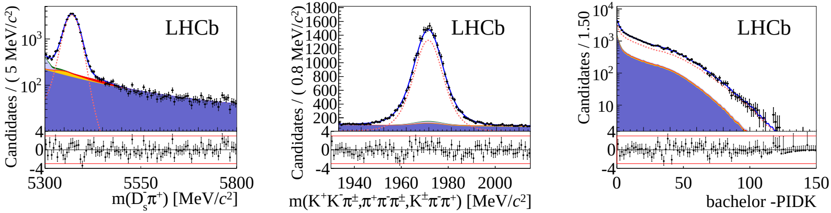
$B_s^0 \to D_s^\mp K^\pm$



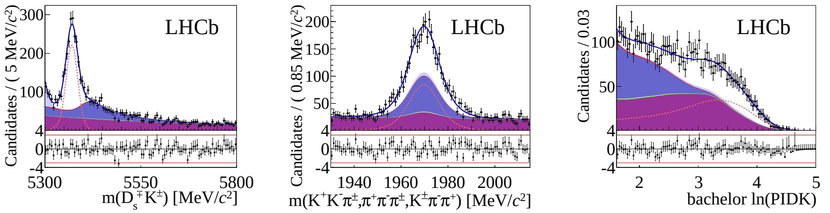
LHCb-PAPER-2014-038

$D_s h$ mass $KK\pi, K\pi\pi, \pi\pi\pi$ mass bachelor PIDK

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



$B_s^0 \to D_s^\mp K^\pm$



LHCb-PAPER-2014-038

Improvements and cleanings up

Cleaning up

We removed from the package obsolete classes/config files/scripts.

- classes :

  ```
  B2DXFitters/NonOscTaggingPdf.h
  B2DXFitters/DTAcceptanceLHCbNote2007041.h
  B2DXFitters/IfThreeWayCat.h
  B2DXFitters/BdPTAcceptance.h
  B2DXFitters/IfThreeWay.h
  B2DXFitters/RooCruijff.h
  B2DXFitters/PowLawAcceptance.h
  B2DXFitters/IfThreeWayCatPdf.h
  B2DXFitters/TagEfficiencyWeightNoCat.h
  B2DXFitters/FinalStateChargePdf.h
  B2DXFitters/IfThreeWayPdf.h
  B2DXFitters/KinHack.h
  B2DXFitters/TagEfficiencyWeight.h
  B2DXFitters/TaggingCat.h
  B2DXFitters/Dilution.h
  ```

- scripts:

  almost all config files in `B2DXFitters/data`

  all bash scripts in `B2DXFitters/scripts/Csh`

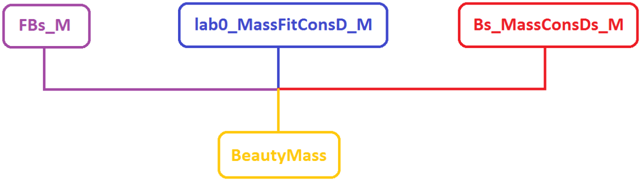
Merging

- So far decay mode had its own files for preparation workspace, running MDFit and plotting.
- This introduces $3*7 = 21$ scripts! Sounds like a mess.
- The most challenging task was merging campaign.
- I made effort to merge everything into three files:
  - `prepareWorkspace.py`
    - for preparing workspace with templates and data sets,
  - `runMDFitter.py` - for running MDFit,
  - `plotMDFitter.py` - for plotting obtained results.
- It reduces number of scripts, but needed a lot of rewriting code.
- If You have ever used old version of package... please forget it.
- In the tutorial session I will explain in the details how to use the new procedure, now I would like to only highlights the most important improvements.

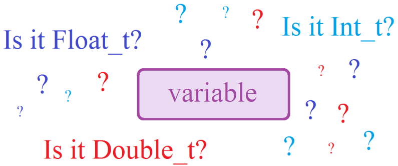
Preparing workspace using prepareWorkspace.py.

Main improvements

Challenge: analyses use different branch names for variables.

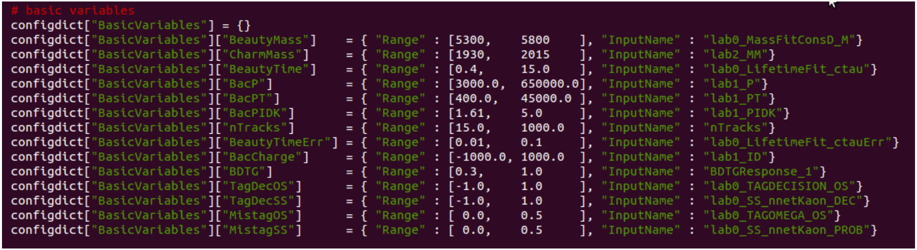


Challenge: core of the MDFitter written in C++,
analyses use different type of branches.



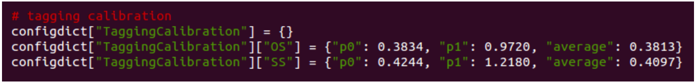
Main improvements

Solution: convert everything into common names.
Advantage: the input to sFit/cFit always the same!



```
# 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"}
```

- Not all of them have to be defined.
- Tagging decision are read as RooCategory, other variables as RooRealVar
- If specified:

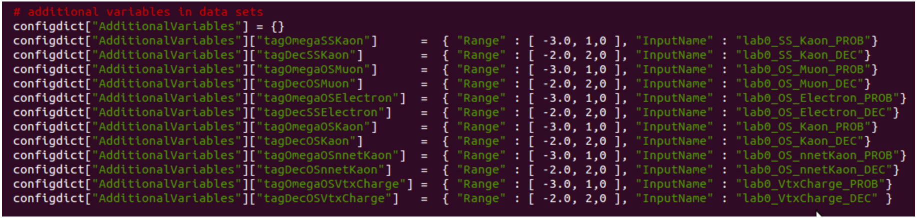


```
# tagging calibration
configdict["TaggingCalibration"] = {}
configdict["TaggingCalibration"]["OS"] = {"p0": 0.3834, "p1": 0.9720, "average": 0.3813}
configdict["TaggingCalibration"]["SS"] = {"p0": 0.4244, "p1": 1.2180, "average": 0.4097}
```

the calibrated mistags, combined tagging decision and mistags are saved to RooDataSet and computed by default.

Main improvements

- Any other “not basic” variable can be added to data set using:

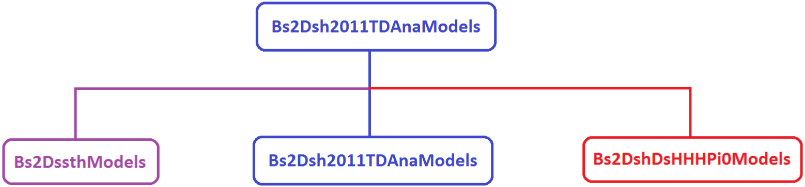


```
# additional variables in data sets
configdict["AdditionalVariables"] = {}
configdict["AdditionalVariables"]["tagOmegaSSKaon"]      =  { "Range" : [ -3.0, 1,0 ], "InputName" : "lab0_SS_Kaon_PROB"}
configdict["AdditionalVariables"]["tagDecSSKaon"]        = { "Range" : [ -2.0, 2,0 ], "InputName" : "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" }
```

- Please note: all additional variables are saved as `RooRealVar`.
- Possible improvement: add flag for saving either as `RooRealVar` or `RooCategory`.

Main improvements

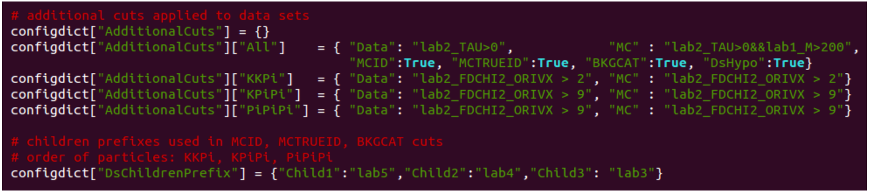
Cleaning up: splitting namespace with MDFitter PDF models.



- Signal and combinatorial background models shared among analyses in `Bs2Dsh2011TDAnaModels` (see later).
- Total specific background PDF models split according to analysis.

Main improvements

Challenge: What kind of cuts do you want apply? Depends on analysis.



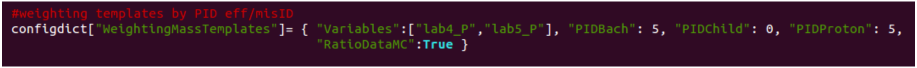
```
# additional cuts applied to data sets
configdict["AdditionalCuts"] = {}
configdict["AdditionalCuts"]["All"]    = { "Data": "lab2_TAU>0",            "MC" : "lab2_TAU>0&&lab1_M>200",
                                           "MCID":True, "MCTRUEID":True, "BKGCAT":True, "DsHypo":True}
configdict["AdditionalCuts"]["KKPi"]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

- The package recognise following $D_s^{\mp}$ final states:
  $KK\pi$ (KKPi), Non Resonant $KK\pi$ (NonRes), $\phi\pi$ (PhiPi), K^*K (KstK),
  $K\pi\pi$ (KPiPi), $\pi\pi\pi$ (PiPiPi) and $KK\pi\pi^0$ (HHHPi0).
- Option "All" sets cuts to all $D_s^{\mp}$ final states as logical AND.
- You can split cuts for either data or MC samples.
- Specific cuts for MC are supported: BKGCAT, DsHypo, ID, TRUEID.
- Specific cuts need `DsChildrenPrefix` to be specified.

By default the cuts are not applied!

Main improvements

Challenge: weighting templates.



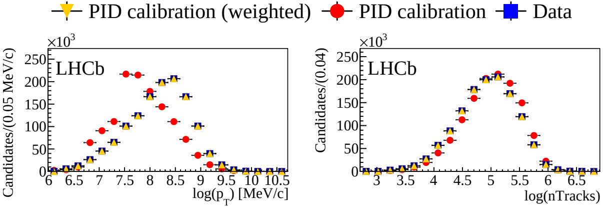
```
#weighting templates by PID eff/misID
configdict["WeightingMassTemplates"]= { "Variables":["lab4_P","lab5_P"], "PIDBach": 5, "PIDChild": 0, "PIDProton": 5,
                                        "RatioDataMC":True }
```

- The PID/mass templates can be weighted by:
  ratio data/MC obtained by $D_s^{\mp}K^{\pm}$ analysis
  misID/eff PID histograms.
- For PID misID/eff weighting have to be set:
  name of momentum variable for particle which is misID as pion: $K \rightarrow \pi$ in D_s
  value of PIDK cut on that variable: “PIDChild”: 0
  name of momentum variable for particle which is misID as proton: $K \rightarrow p$ in D_s
  value of PIDK cut on that variable: ”PIDProton”: 5
  PIDK cut on bachelor: ”PIDBach”: 5
- misID/eff histograms only provided for Stripping 17.
  If you are willing to take care of histograms for Stripping 20/21, please let me know!

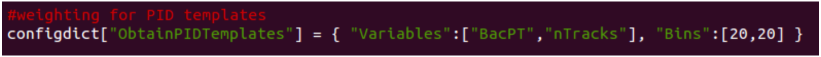
By default no weighting is applied!

Main improvements

Challenge: obtaining PIDK shapes.



- Reminder: PIDK is obtained by reweighting kinematics of calibration samples.
- For Strippings $>$ 17 the templates are obtained as `RooBinned1DQuinticBase` (it speeds up quite a lot procedure).

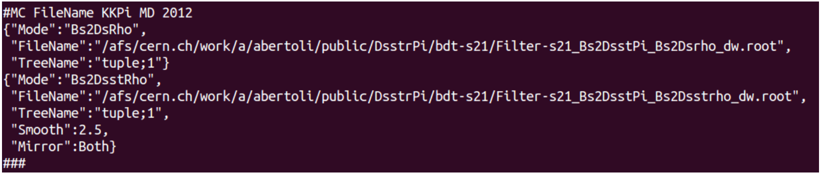


```
#weighting for PID templates
configdict["ObtainPIDTemplates"] = { "Variables":["BacPT","nTracks"], "Bins":[20,20] }
```

- Variables are OUTPUT names for variables used to weighting calibration samples
- Bins are number of bins in each direction.

Main improvements

Challenge: statistical limitation of MC samples requires more flexible `RooKeysPdf`.



```
#MC FileName KKPi MD 2012
{"Mode":"Bs2DsRho",
 "FileName":"/afs/cern.ch/work/a/abertoli/public/DsstrPi/bdt-s21/Filter-s21_Bs2DsstPi_Bs2Dsrho_dw.root",
 "TreeName":"tuple;1"}
{"Mode":"Bs2DsstRho",
 "FileName":"/afs/cern.ch/work/a/abertoli/public/DsstrPi/bdt-s21/Filter-s21_Bs2DsstPi_Bs2Dsstrho_dw.root",
 "TreeName":"tuple;1",
 "Smooth":2.5,
 "Mirror":Both}
###
```

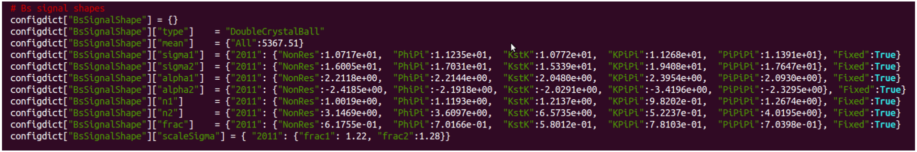
- New class: `MCBackground`
- In the config.txt file new MC background declaration.
- `Smooth` and `Mirror` are `RooKeysPdf` parameters, default value: 1.5, Both
- Still some work needs to be done.

Warning: it will be moved soon from the .txt file to python script!
It has a high priority on my to-do list.

Fitting using runMDFitter.py

Main improvements

Challenge: analyses need different signal description.



```
# 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 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.

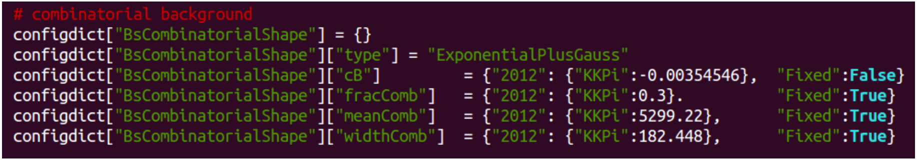
In the package so far the supported shapes are:

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

If you want to add your own parametrization, let me know and I will help you in implementing it!

Main improvements

Challenge: analyses need different combinatorial background description.



```
# 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,

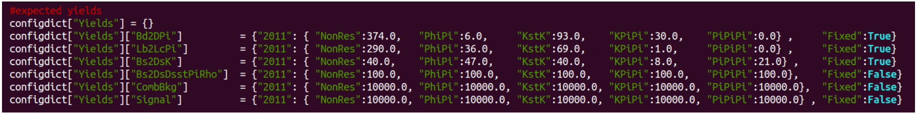
In the package so far the supported shapes are:

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

If you want to add your own parametrization, let me know and I will help you in implementing it!

Main improvements

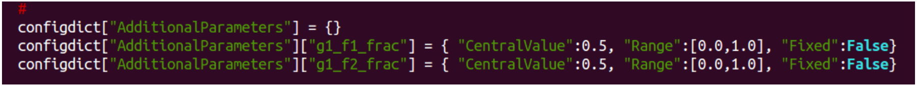
Challenge: analyses need different specific background description.



```
#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:

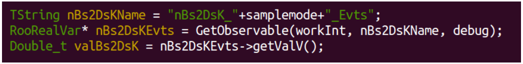
- set yields separately for each fitted D_s final state and data year,
- fix or float yields using "Fixed" variable,
- add necessary parameters for your background description.
- specify your background description in dedicated namespace `Bs2Dsh2011TDAnaModels`, `Bs2DssthModels`, `Bs2DshDsHHHPi0Models`.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



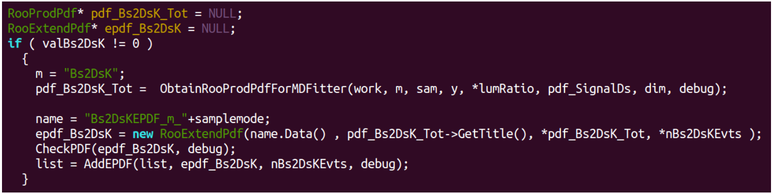
```
#
configdict["AdditionalParameters"] = {}
configdict["AdditionalParameters"]["g1_f1_frac"] = { "CentralValue":0.5, "Range":[0.0,1.0], "Fixed":False}
configdict["AdditionalParameters"]["g1_f2_frac"] = { "CentralValue":0.5, "Range":[0.0,1.0], "Fixed":False}
```

How can I build my background PDF?

- In one of the namespaces:
  `Bs2Dsh2011TDAnaModels`,
  `Bs2DssthModels`,
  `Bs2DshDsHHHPi0Models`
  you can create your specific backgrounds model.
- For simply adding PDFs it should be enough:



```
TString nBs2DsKName = "nBs2DsK_"+samplemode+"_Evts";
RooRealVar* nBs2DsKEvts = GetObservable(workInt, nBs2DsKName, debug);
Double_t valBs2DsK = nBs2DsKEvts->getValV();
```



```
RooProdPdf* pdf_Bs2DsK_Tot = NULL;
RooExtendPdf* epdf_Bs2DsK = NULL;
if ( valBs2DsK != 0 )
  {
    m = "Bs2DsK";
    pdf_Bs2DsK_Tot =  ObtainRooProdPdfForMDFitter(work, m, sam, y, *lumRatio, pdf_SignalDs, dim, debug);

    name = "Bs2DsKEPDF_m_"+samplemode;
    epdf_Bs2DsK = new RooExtendPdf(name.Data() , pdf_Bs2DsK_Tot->GetTitle(), *pdf_Bs2DsK_Tot, *nBs2DsKEvts );
    CheckPDF(epdf_Bs2DsK, debug);
    list = AddEPDF(list, epdf_Bs2DsK, nBs2DsKEvts, debug);
  }
```

- If you want to add some PDFs together you need to add `pdf_mode_Tot` and then create extended PDF.

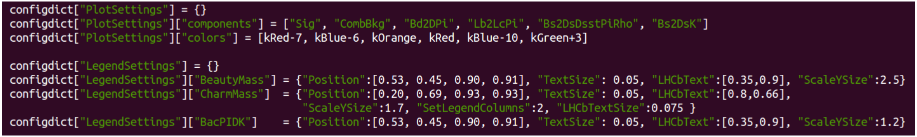
Main improvements

- Supported options for $D_s^{\mp}$ final state:
  - one final state from list [NonRes, PhiPi, KstK, KPiPi, PiPiPi, HHHPi0]
  - simultaneous fit three $KK\pi$ final states: [NonRes, PhiPi, KstK]
  - simultaneous fit five "all" final states: [NonRes, PhiPi, KstK, KPiPi, PiPiPi]
- Supported options for magnet polarity:
  - one polarity
  - simultaneous fit to both polarities
  - combined fit to both polarities
- Supported options for year of data taking:
  - one ear of data taking
  - simultaneous fit to both data years
  - combined fit to both data years (under construction, almost done)
- Supported options for dimension of fit:
  - one dimensional to beauty meson mass,
  - two dimensional to beauty and charm meson masses,
  - three dimensional to beauty and charm meson masses and PIDK of bachelor.

Plotting results using plotMDFitter.py

Main improvements

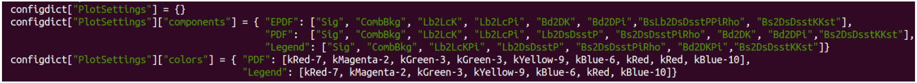
Challenge: more flexible interface.



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

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}
```

You can:

- set contributions to plot together with their colors,
- control position of legend and "LHCb" text separately for each variable,
- rescale Y axis of your plot (to make a place for legend),
- plot not only EPDF (more adavnced example for DsK below).



```
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]}
```

These are small things, but necessary if we want to keep only one plotting script for all analyses.

Conclusion

- To meet expectations from different analyses `B2DXFitters` improved its flexibility.
- The main fitting routine is merged for all analyses.
- You will learn how to use it over tutorial session.
- Still there is a lot of improvements needed.
- Please stay tuned for further developments, which will come soon!

Don't forget: only with your contributions the package makes sense. You are very welcome to help in developing it!

Thank You