EspressoPerformanceMonitor Tutorial for v0.5

Jack Wimberley

Department of Physics
University of Maryland, College Park

25 April 2016

EspressoPerformanceMonitor Tutorial

- The EspressoPerformanceMonitor is a tool designed to:
  - ease the burden of calibrating flavour tagging algorithms
  - evaluate their tagging performance
  - produce helpful LaTeX tables and uniformly styled graphs
- This tutorial will show how the tool can be used in a B2CC workflow
- Legend for the tutorial: Statements marked...
  - Warning will point out possible points of confusion
  - Annoying will point out rough edges that we hope to improve

Overview

- Downloading and compiling tool
- Basic configurable options of the tool
- Running over a $B^0 \to J/\psi K_S$ tuple
- Calibrating OS tagger on $B^+ \to J/\psi K^+$
- Calibrating SS pion tagger on $B^0 \to J/\psi K^{*0}$
- Calibrating taggers on $B^0 \to J/\psi K_S$ Monte Carlo
- Running over $B^0 \to J/\psi K_S$ tuple again

GitLab repo and dependencies

- EspressoPerformanceMonitor (EPM) is hosted on GitLab
- Can be compiled on LXPLUS (and likely CernVM; untested)
- Alternatively, the tool can be compiled on Linux / OS X with the following dependencies:
  - CMake version 3.0+
  - Boost version 1.59+ (or slightly earlier)
  - GSL version 1.13+ (or slightly earlier)
  - ROOT version 5.34+

Downloading the package

- Go to the directory where you would like the tool located
- Clone the git repository, checkout `v0.5`, and go into the new directory:

```
git clone ssh://git@gitlab.cern.ch:7999/
  lhcb-ft/EspressoPerformanceMonitor.git
cd EspressoPerformanceMonitor
git checkout v0.5
git submodule update --init
```

Compiling the package on LXPLUS

- A bash shell script defining the proper environment is provided to make compilation on LXPLUS simple:

```
#source /cvmfs/lhcb.cern.ch/group_login.sh
source setenv.sh # or source setenv.csh in tcsh shell
cmake .
make -j8 SimpleEvaluator
```

- This script only uses `LbLogin.sh` and `/cvmfs/...` paths
- Should work on CernVM, too
- Requires using bash; might need to source group login script

Compiling the package on other platforms

- Standard package management should work on Linux
- If `ROOT` is compiled without `xrootd` support, use `-DXROOTD=OFF`
- Dependencies must be met, most easily with homebrew
  - Homebrew CMake, Boost, and GSL work out-of-the-box
- CMake must be able to find FindROOT.cmake; tricky
- The following works for ROOT 6:

```
cmake -DCMAKE_MODULE_PATH=${ROOTSYS}/etc/root/cmake .
make -j8 SimpleEvaluator
```

How do you run the tool?

- The EPM is configured with a Python(-like) options file
  - Options set via

```
OptionName = OptionValue
```

  - Python style comments are supported
  - Other options files can be imported via

```
import opts1.py opts2.py opts3.py # other options
```

- After creating an options file `opts.py`, the EPM will be run via

```
/path/to/EPM/bin/SimpleEvaluator opts.py
```

Basic options for the EPM

- The basic options for the EPM are the `ROOT` file and `TTree` name
- The first is set via the option `datafile`; can be relative or absolute
- The latter is set via the option `TupleName`

```
RootFile = "ntuple.root" # same directory
RootFile = "../ntuple.root" # parent directory
RootFile = "~jwimberl/public/.../ntuple.root"
RootFile = "root://eoslhcb.cern.ch//eos/.../ntuple.root"
TupleName = "DecayTree"
TupleName = "Bd2JpsiKSDetached/DecayTree"
```

Running over multiple files

You can run over multiple files using the following:

```
NumFiles = 4
RootFile_1 = "ntuple_2011_Up.root"
RootFile_2 = "ntuple_2011_Down.root"
RootFile_3 = "ntuple_2012_Up.root"
RootFile_4 = "ntuple_2012_Down.root"
```

Selecting events

- There are two options that let you select the events from the TTree that you would like to process
- `Nmax` (integer) is the maximum number of events; `-1` means all
- `Selection` (string) is a cut using `TTree` variables

```
# all events; default
Nmax = -1
Selection = ""

# first 50k events passing trigger cut
Nmax = 50000
Selection = "Bd_Hlt2_ImaginaryTrigger_TOS"
```

Describing the signal B meson

- Several important properties of the signal B meson:
  - its species (B^+, B^0, or B_s)
  - whether it is signal or not (e.g. *sWeight*)
  - its flavour at decay time (possibly unknown)
  - its decay time (if B^0 or B_s)
- The last two are only necessary for calibrations

Describing the signal B meson species and weight

- `CalibrationMode = "Bu"` or `"Bd"` or `"Bs"`
  - Warning - Misnomer; needed even when not calibrating
- `UseWeight = 1` turns on the use of weights
- `BranchWeight` = a `TTree` branch name or formula interpreted with `TTreeFormula`
  - Annoying - A simple `Weight` string would be simpler

```
CalibrationMode = "Bs" # charged B mode
UseWeight = 1
BranchWeight = "N_sig_sw"
```

Selecting taggers

- Each tagger has its own name and options
- Most relevant for analysts: `OS_Combination`, `SS_Kaon`, `SS_nnetKaon`, `SS_Pion`, and `SS_PionBDT`, `SS_ProtonBDT`
- A particular `Tagger` out of the above can be configured via several options:
  - `<TaggerName>_Use = 1` if you want to study it
  - `<TaggerName>_BranchDec` = branch of tagger's decision $(\pm 1, 0)$
  - `<TaggerName>_TypeDec` = either `"Int_t"` (def.) or `"Short_t"`
  - `<TaggerName>_BranchProb` = branch of tagger's mistag
  - `<TaggerName>_TypeProb` = either `"Double_t"` (def.) or `"Float_t"`
- Annoying - the branch types could be determined automatically

Tutorial scenario

Imagine the following scenario:

- You are performing a study using the decay $B^0 \to J/\psi K_S$
- You will be using the `OS_Combination` and `SS_Pion` taggers
- You plan on using $B^+ \to J/\psi K^+$ to calibrate the OS tagger
- You plan on using $B^0 \to J/\psi K^{*0}$ to calibrate the SS tagger

Running over a $B^0 \to J/\psi K_S$ tuple

- You've processed a $B^0 \to J/\psi K_S$ dataset and fit the mass spectrum to obtain *sWeights*, storing your ROOT file on EOS
- Create a directory `Bd2JpsiKS` inside ‘EspressoPerformanceMonitor“
- Create an options file `Bd2JpsiKS/opts.py`
- Begin the options file as follows (a working copy can be found in the directory `/eos/lhcb/wg/FlavourTagging/tutorials/CPTT_201609`, hereafter referred to as `EOSDIR`: `EOSDIR/Bd2JpsiKS/opts.py`):

```
RootFile = "root://eoslhcb.cern.ch//eos/lhcb/wg/
    FlavourTagging/tutorials/CPTT_201609/data/Bd2JpsiKS.
    root"
TupleName = "Bd2JpsiKSDetached"
CalibrationMode = "Bd"
UseWeight = 1
BranchWeight = "SigYield_sw"
```

Adding desired taggers

Then add the OS Combination and SS Pion taggers:

```
OS_Combination_Use = 1
OS_Combination_BranchDec = "B_TAGDECISION_OS"
OS_Combination_BranchProb = "B_TAGOMEGA_OS"
SS_Pion_Use    = 1
SS_Pion_TypeDec = "Short_t"
SS_Pion_BranchDec = "B_SS_Pion_DEC"
SS_Pion_TypeProb = "Float_t"
SS_Pion_BranchProb = "B_SS_Pion_PROB"
```

Output

- cd into `B2JpsiKS`
- Run the EPM with `../bin/SimpleEvaluator opts.py`
- The tool first dumps a lot of configuration information to the screen:

```
READING CONFIGURATION FROM opts.py
 *** string CalibrationMode = Bu changed to Bd
 *** string BranchWeight =  changed to SigYield_sw
 *** double UseWeight = 0 changed to 1
 *** double SS_Pion_Use = 0 changed to 1
 ...
```

Output (cont'd)

- Then the tool prints the efficiency of the `Selection` and starts looping over the events:

```
Reading file root://eoslhcb.cern.ch//eos/lhcb/wg/FlavourTa

Cut  keeps 38425.9 out of 38425.9 events (weighted).
Corresponding efficiency: 1 (+ -1.155e-05, -0.000123).
SETING EXPECTED NUMBER OF EVENTS = 131681
Loading Weight branch
Loading SS_Pion branches
Loading OS_Combination branches
ON EVENT 0
ON EVENT 10000
...
```

Output: Tagger correlations

- The first displayed results are several correlation matrices of the taggers firing rates and tagging decisions
- First: Correlation in whether the variables have fired or not (0 or 1):

```
//////////////////////////////////////////////
Tagger Fire Correlations [%]
//////////////////////////////////////////////
                          SS_Pion  OS_Combination
         SS_Pion       100.0000%         1.8610%
  OS_Combination         1.8610%       100.0000%
//////////////////////////////////////////////
```

Output: Tagger correlations (cont'd)

Second: Correlation in the tagger decisions (-1, 0, or $+1$):

```
/////////////////////////////////////////////
Tagger decision Correlations [%]
/////////////////////////////////////////////
                        SS_Pion  OS_Combination
        SS_Pion        100.0000%        0.6906%
 OS_Combination          0.6906%      100.0000%
/////////////////////////////////////////////
```

Output: Tagger correlations (cont'd)

Third: Alternatively, the tagging decision can be weighted by its dilution $D = 1 - 2\omega$ so that marginal decisions are close to 0 ($-D$, 0, or D).

```
////////////////////////////////////////////////////
Tagger decision Correlations (dilution weighted) [%]
////////////////////////////////////////////////////
                         SS_Pion  OS_Combination
          SS_Pion      100.0000%         1.3496%
   OS_Combination        1.3496%       100.0000%
////////////////////////////////////////////////////
```

Output: Tagger correlations (cont'd)

Fourth: For each pair of taggers, you can just look at the events where both fire and find the correlation in their decision (which is now a variable that is either -1 or $+1$):

```
////////////////////////////////////////////////
Tagger decision Correlations (when both fire) [%]
////////////////////////////////////////////////
                         SS_Pion  OS_Combination
         SS_Pion       100.0000%        5.0216%
  OS_Combination         5.0216%      100.0000%
////////////////////////////////////////////////
```

Tagging performance table

- Values have statistical uncertainty (stat) and uncertainty that propagates from the calibration parameters (cal).
  - The tagging rate is $\epsilon = (R + W)/(R + W + U)$
  - The raw mistag is the fraction of mistagged events $W/(R + W)$
    - Meaningless when the B flavor isn't known
  - The effective mistag is $\omega_{\text{eff}} = (1 - \sqrt{\langle D^2 \rangle})/2$
  - The tagging power is $\epsilon_{\text{eff}} = \epsilon \langle D^2 \rangle = \epsilon\,(1 - 2\omega_{\text{eff}})^2$
- Also saved to `EspressoPerformanceTable.tex`

```
-----------------------------------------------
------------- TAGGING PERFORMANCES -------------
-----------------------------------------------
```

Tagger	Tagging Rate	Raw Mistag	Eff Mistag	Tagging Power
SS_Pion	(4.5924 ± 0.1711)%	(49.8688 ± 1.1902)%	(38.8739 ± 0.1079(stat) ± 0.0000(cal)%	(0.2274 ± 0.0096(stat) ± 0.0000(cal)%
OS_Combination	(38.2112 ± 0.3973)%	(52.5322 ± 0.4121)%	(35.4504 ± 0.0774(stat) ± 0.0000(cal)%	(3.2356 ± 0.0481(stat) ± 0.0000(cal)%

Eta percentile table

- The next table printed lists percentiles of the of η
- Useful for unit tests; not very interesting for your analysis
- Also saved to `EspressoEtaPercentilesTable.tex`

```
---------------------------------------------
---------------- ETA PERCENTILES ---------------
---------------------------------------------

         Tagger        0%       18%       50%       82%      100%
        SS_Pion     0.217     0.361     0.406     0.430      -nan
 OS_Combination     0.081     0.309     0.399     0.462     0.500
```

Histograms and graphs

- Two graphs created for each tagger:
  - `<TaggerName>_EtaDist.png` is a histogram of the η distribution
  - `<TaggerName>_IntPower.png` is less interesting to you; plots cumulative tagging power for all tags with $\eta <$ value on x-axis
- These are also saved in `EspressoHistograms.root`

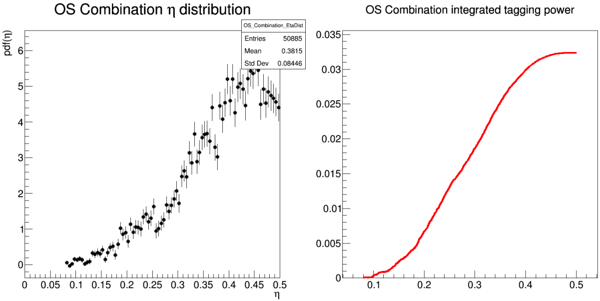


Figure: Examples for OS Combination

Adding OS+SS combination

- `PerformOfflineCombination_OSplusSS = 1` turns on the OS+SS combination
- `<TaggerName>_InComb` add tagger it to OS+SS combination
- So, you add the following to your options file:

```
PerformOfflineCombination_OSplusSS = 1
OS_Combination_InComb = 1
SS_Pion_InComb = 1
```

- `EOSDIR/Bd2JpsiKS/comb/opts.py`

OS+SS combination output

- Combination tagger added to performance table:

```
--------------------------------------------------
------------- TAGGING PERFORMANCES -------------
--------------------------------------------------
```

Tagger	Tagging Rate	Raw Mistag	Eff Mistag	Tagging Power
SS_Pion	(4.5924 ± 0.1711)%	(49.8688 ± 1.1902)%	(38.8739 ± 0.1079(stat) ± 0.0000(cal)%	(0.2274 ± 0.0096(stat) ± 0.0000(cal)%
OS_Combination	(38.2112 ± 0.3973)%	(52.5322 ± 0.4121)%	(35.4504 ± 0.0774(stat) ± 0.0000(cal)%	(3.2356 ± 0.0481(stat) ± 0.0000(cal)%
Combination	(40.8595 ± 0.4019)%	(52.3908 ± 0.3986)%	(35.4644 ± 0.0752(stat) ± 0.0000(cal)%	(3.4532 ± 0.0493(stat) ± 0.0000(cal)%

- A new table of performance in exclusive categories is printed:

```
------------------------------------------------------------
------------- OS+SS COMBINATION PERFORMANCES -------------
------------------------------------------------------------
```

Tagger	Tagging Rate	Raw Mistag	Eff Mistag	Tagging Power
SPi	(2.6483 ± 0.1313)%	(52.0147 ± 1.5661)%	(39.0883 ± 0.1335(stat) ± 0.0000(cal)%	(0.1261 ± 0.0070(stat) ± 0.0000(cal)%
OS	(36.2671 ± 0.3931)%	(52.5302 ± 0.4230)%	(35.4614 ± 0.0795(stat) ± 0.0000(cal)%	(3.0663 ± 0.0472(stat) ± 0.0000(cal)%
SPi+OS	(1.9441 ± 0.1129)%	(50.3015 ± 1.8293)%	(31.6886 ± 0.3884(stat) ± 0.0000(cal)%	(0.2608 ± 0.0188(stat) ± 0.0000(cal)%

- Also saved to `EspressoPerformanceTable_OSplusSSComb.tex`

EspressoPerformanceMonitor calibrations

- The EPM performs calibrations using binomial regression
  - Likelihood maximization via Newton-Raphson or Minuit
  - No binning procedure is used during the calibration
  - With *sWeights*, an sLikelihood function is used and the covariance matrix is corrected
- Calibrations are performed when `DoCalibrations = 1`
- Extra requirements for calibration:
  - The B flavour at decay (or true MC flavour at production)
  - The lifetime (for B^0 and B_s) and resolution (for B_s)

Describing the signal B meson flavour

- `BranchID` = branch giving B flavour ($\pm 1, 521, 531, \ldots$)
- `TypeID` = either `"Int_t"` (def.) or `"Short_t"`

```
BranchID = "B_id"
TypeID = "Int_t" # default; superfluous

BranchID = "Bs_ID"
TypeID = "Short_t"
```

Calibrating OS tagger on $B^+ \to J/\psi K^+$

- You've processed a $B^+ \to J/\psi K^+$ dataset and fit the mass spectrum to obtain *sWeights*
- Create a directory `B2JpsiK` inside 'EspressoPerformanceMonitor"
- Create an options file `B2JpsiK/opts.py`
- `EOSDIR/B2JpsiK/opts.py`

```
RootFile = "root://eoslhcb.cern.ch//eos/lhcb/wg/FlavourTagging/tutorials/CPTT_201609/data
    /B2JpsiK.root"
TupleName = "Bu2JpsiKDetached"
CalibrationMode = "Bu"
UseWeight = 1
BranchWeight = "SigYield_sw"
BranchID = "B_ID"
DoCalibrations = 1
OS_Combination_Use = 1
OS_Combination_BranchDec = "B_TAGDECISION_OS"
OS_Combination_BranchProb = "B_TAGOMEGA_OS"
```

Initial output

- Initial output similar to last step
  - Correlation matrices are boring (1x1)
  - A table shows the performance of the OS tagger:

```
-------------------------------------------------
------------- TAGGING PERFORMANCES -------------
-------------------------------------------------

        Tagger        Tagging Rate          Raw Mistag                               Eff Mistag                                Tagging Power
OS_Combination   (31.8269 ± 0.1780)%   (38.4709 ± 0.2946)%   (35.0409 ± 0.0593(stat) ± 0.0000(cal)%   (2.8488 ± 0.0276(stat) ± 0.0000(cal)%
```

  - Table of eta percentiles is printed:

```
-------------------------------------------------
---------------- ETA PERCENTILES ----------------
-------------------------------------------------

        Tagger       0%      18%      50%      82%      10
OS_Combination    0.079    0.303    0.400    0.463    0.5
```

- But now more information related to calibration is printed under the heading `TAGGER CALIBRATION METRICS`

Mis-calibration tests

- The first item printed are mis-calibration tests
- Don't worry about these; more interesting for the FTWG

```
----------------------
--- OS_Combination ---
----------------------

-- INPUT CALIBRATION G.O.F. --
DEVIANCE        = 27620.82
Brier Score     = 6676.59
PEARSON X2 test = 2.884
lCvHCH S test   = 2.909
```

The calibration itself

- Slightly different convention: $\omega = \eta + p_0 + p_1 (\eta - \langle\eta\rangle)$
  - p_0 here $=$ normal $p_0 - \langle\eta\rangle$; should be close to 0
  - p_1 here $=$ normal $p_1 - 1$; should be close to 0

```
-- MISTAG BASIS --

 -- POLYNOMIAL BASIS --
P_0(x) = 1
P_1(x) = x -0.3789

 -- PARAMETER VALUES --
p0 = 0.0055254 +-0.0033136
p1 = -0.053434 +-0.034528

 -- PARAMETER DELTA VALUES --
Δp0 = 0.018054 +-0.0066273
Δp1 = 0.032893 +-0.069055

 -- CORRELATION MATRIX --
       p0          p1         Δp0          Δp1
  p0    1     0.12781  -0.0026025   -0.0065853
  p1  ---           1  -0.0065853   -0.0055301
 Δp0  ---         ---           1      0.12781
 Δp1  ---         ---         ---            1
```

Goodness-of-fit tests

- The next item printed are more goodness-of-fit tests
- Again, don't worry about these; more interesting for the FTWG

```
-- OUTPUT CALIBRATION G.O.F. --
DEVIANCE          = 27606.87
AIC               = 27610.87
BIC               = 27629.14
Brier Score       = 6592.45
PEARSON X2 test = 6.459e-06
lCvHCH S test    = 0.7925
```

Calibrated tagging performance table

- Performance table reprinted with calibration applied
- Values now have nonzero uncertainty propagating from the calibration parameters to `EspressoCalibratedPerformanceTable.tex`

```
-------------------------------------------------
-------- CALIBRATED TAGGING PERFORMANCES ---------
-------------------------------------------------
```

Tagger	Tagging Rate	Raw Mistag	Eff Mistag	Tagging Power
OS_Combination	(31.8269 ± 0.1780)%	(38.4709 ± 0.2946)%	(35.7638 ± 0.0561(stat) ± 0.3026(cal)%	(2.5801 ± 0.0250(stat) ± 0.1097(cal)%

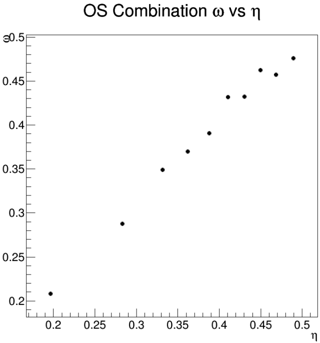
Calibration archive

- Calibration saved to `EspressoCalibrations.py`
- This will be used in later stages to add calibration to Bd2JpsiKS
- Annoying - The convention here is different again
  - p_0 here $=$ normal $p_0 - \langle\eta\rangle$; should be close to 0
  - p_1 here is the normal p_1; should be close to 1

```
OS_Combination_Link       = "MISTAG"
OS_Combination_Eta        = 0.378933
OS_Combination_P0         = 0.00552545
OS_Combination_P1         = 0.946566
OS_Combination_P0Err      = 0.00331364
OS_Combination_P1Err      = 0.0345277
OS_Combination_RhoP0P1    = 0.127808
```

Calibration plots

- `OS_Combination_BinTable.png` is a graph of the mistagged fraction ω vs predicted mistag rate η in 10 deciles of η; also saved to CSV
- `..._SmoothedBinTable.png` plots a kernel-smoothed version



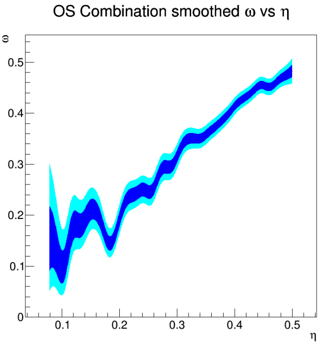


Figure: A very linear ω vs η trend is apparent in the data

Calibration plots (part II)

- `..._InputCalibration.png` and `..._InputCalibration_SmoothedData.png` additionally plot a graph of $\omega = \eta$

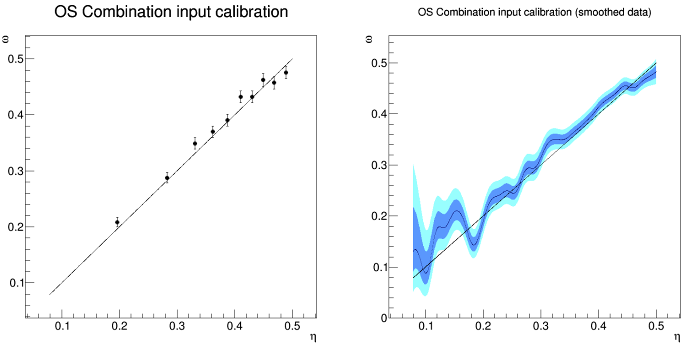


Figure: The data is close to consistent with $\omega = \eta$

Calibration plots (part III)

- `..._Calibration.png` and `..._Calibration_SmoothedData.png` additionally plot a graph of the measured calibration

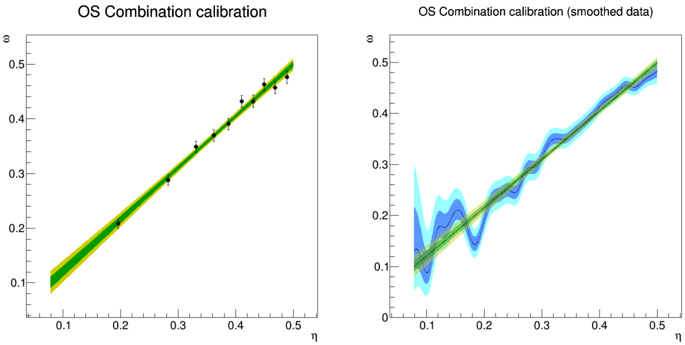


Figure: The linear calibration appears to match the data very well

Other plots

- Other less interesting graphs and histograms are produced:
  - The `...EtaDist.png` and `...IntPower.png` graphs are still made
  - `...EtaDistRight.png` and `...EtaDistWrong.png` show the η distribution for rightly and wrongly tagged events
  - `OS_Combination_ROC.png` shows the ROC curve for the tagger
- All plots saved in `EspressoHistograms.root`

Describing the signal B meson decay time

- For a neutral B meson, the EPM needs to know the decay time to account for oscillation in its flavour
  - Warning - Except for MC, where the ID is the production
- The lifetime can be configured via the options:
  - `UseTau = 1` to account for oscillation
  - `BranchTau` = the lifetime branch; `TypeTau` = either `"Double_t"` (def.) or `"Float_t"`
  - `TauUnits = "ps"` by default; other options are `"ns"` and `"fs"`
- The lifetime resolution can be configured via the options:
  - `UseTauErr = 1` to account for lifetime resolution
  - `BranchTauErr` = the lifetime branch; `TypeTauErr` = either `"Double_t"` (def.) or `"Float_t"`
  - Resolution models, important for B_s, will be discussed later

Calibrating SS pion tagger on $B^0 \to J/\psi K^{*0}$

- You've processed a $B^0 \to J/\psi K^{*0}$ dataset and fit the mass spectrum to obtain *sWeights*
- Create a directory `Bd2JpsiKst` inside "EspressoPerformanceMonitor"
- Create an options file `Bd2JpsiKst/opts.py`
- `EOSDIR/Bd2JpsiKst/opts.py`

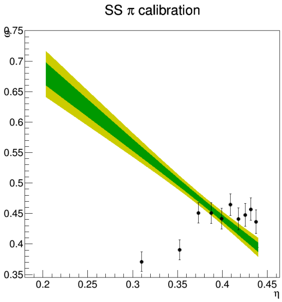
```
RootFile = "root://eoslhcb.cern.ch//eos/lhcb/wg/FlavourTagging/tutorials/CPTT_201609/data
    /Bd2JpsiKst.root"
TupleName = "DecayTree"
CalibrationMode = "Bd"
UseWeight = 1
BranchWeight = "N_sig_sw"
BranchID = "lab0_ID"
UseTau = 1
BranchTau = "lab0_TAU"
UseTauErr = 1
BranchTauErr = "lab0_TAUERR"
DoCalibrations = 1
SS_Pion_Use = 1
SS_Pion_BranchDec = "lab0_SS_Pion_DEC"
SS_Pion_BranchProb = "lab0_SS_Pion_PROB"
```

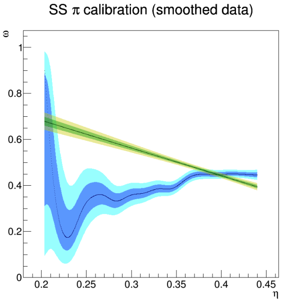
Problem!

- You immediately see that something has gone wrong in the calibration:

```
 -- PARAMETER VALUES --
p0 = 0.060614 +-0.0053238
p1 = -2.203 +-0.1016
```

- You inspect the calibration plots:





Diagnosis and solution

- Very rarely, a big statistical fluctuation leads to a bad minimum
- The `Minuit` solver is more robust in these situations
- Add `UseNewtonRaphson = 0` to the options file
- `EOSDIR/Bd2JpsiKst/minuit/opts.py`

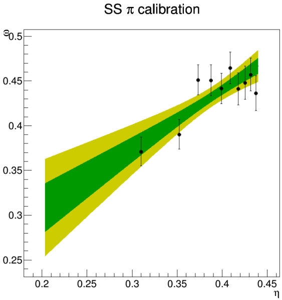
The calibration itself

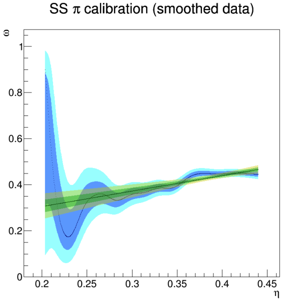
- Now the calibration looks much more sensible
- As before, ignore the miscalibration tests and goodness-of-fit tests
- The calibration is saved to `EspressoCalibrations.py`

```
 -- MISTAG BASIS --

 -- POLYNOMIAL BASIS --
P_0(x) = 1
P_1(x) = x -0.3918

 -- PARAMETER VALUES --
p0 = 0.042615 +-0.0055773
p1 = -0.33017 +-0.14216

 -- PARAMETER DELTA VALUES --
Δp0 = -0.0087545 +-0.010947
Δp1 = -0.83221 +-0.27868

 -- CORRELATION MATRIX --
        p0              p1            Δp0           Δp1
   p0    1         0.02087      -0.039186      -0.01781
   p1  ---               1      -0.017501     0.0070188
  Δp0   ---             ---              1      0.028084
  Δp1   ---             ---            ---             1
```

SS Pion calibration plots

- The calibration plots look much nicer now as well:





- The calibrated performance table is now trustworthy:

```
--------------------------------------------------
------- CALIBRATED TAGGING PERFORMANCES ---------
--------------------------------------------------
```

Tagger	Tagging Rate	Raw Mistag	Eff Mistag	Tagging Power
SS_Pion	(8.8683 ± 0.1025)%	(43.6824 ± 0.5006)%	(42.9327 ± 0.0305(stat) ± 0.5537(cal))%	(0.1772 ± 0.0026(stat) ± 0.0278(cal))%

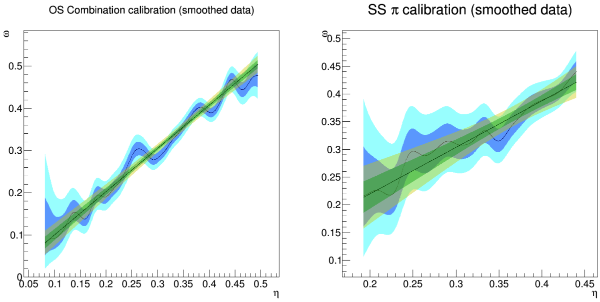
Options file for MC calibration

- You have a prepared MC sample
- The MC sample needs neither *sWeights* nor lifetime variables
- As it happens, the SS Pion branches use `Short_t` and `Float_t`
- `EOSDIR/Bd2JpsiKS_MC/opts.py`

```
RootFile = "root://eoslhcb.cern.ch//eos/lhcb/wg/FlavourTagging/tutorials/CPTT_201609/MC/
    DT_Sim09_Bd2JpsiKS_Sim09_DVv36r1p2_jwishahi_MagUp_P6.root"
TupleName = "Bd2JpsiKSDetached/DecayTree"
CalibrationMode = "Bd"
DoCalibrations = 1
BranchID = "B_TRUEID"
OS_Combination_Use = 1
OS_Combination_BranchDec = "B_TAGDECISION_OS"
OS_Combination_BranchProb = "B_TAGOMEGA_OS"
SS_Pion_Use    = 1
SS_Pion_TypeDec = "Short_t"
SS_Pion_BranchDec = "B_SS_Pion_DEC"
SS_Pion_TypeProb = "Float_t"
SS_Pion_BranchProb = "B_SS_Pion_PROB"
```

MC Calibrations

- Both calibrations are saved in `EspressoCalibrations.py`
- Both calibrations show a good linear fit:



- In an analysis, you'd want to compare these to calibrations from $B^+ \to J/\psi K^+$ MC and $B^0 \to J/\psi K^{*0}$ MC
- This would help you learn how portable the calibrations are across decay channels

Running over $B^0 \to J/\psi K_S$ tuple again

- Go back into your Bd2JpsiKS directory
- Copy B2JpsiK/EspressoCalibrations.py and Bd2JpsiKst/EspressoCalibrations.py into the new directory as files osCal.py and ssCal.py
- Add the following line to opts.py:

```
import osCal.py ssCal.py
WriteCalibratedMistagBranches = 1
OS_Combination_Write = 1
SS_Pion_Write = 1
Combination_Write = 1
```

- EOSDIR/Bd2JpsiKS/cal/opts.py

Output with calibration uncertainties

- SS Pion and OS Comb. taggers now have calibration uncertainty:

```
-----------------------------------------------------
------------- TAGGING PERFORMANCES -------------
-----------------------------------------------------

         Tagger        Tagging Rate           Raw Mistag                        Eff Mistag                          Tagging Power
        SS_Pion      (4.5924 ± 0.1711)%    (49.8688 ± 1.1902)%    (43.1904 ± 0.0732(stat) ± 0.5428(cal)%    (0.0852 ± 0.0037(stat) ± 0.0136(cal)%
 OS_Combination     (38.2112 ± 0.3973)%    (52.5322 ± 0.4121)%    (36.1509 ± 0.0733(stat) ± 0.3321(cal)%    (2.9315 ± 0.0435(stat) ± 0.1406(cal)%
    Combination     (40.8595 ± 0.4019)%    (52.3267 ± 0.3986)%    (36.4111 ± 0.0718(stat) ± 0.0000(cal)%    (3.0180 ± 0.0436(stat) ± 0.0000(cal)%
```

  - Annoying - uncertainty not propagated to combination

- The table of performance in exclusive categories is printed:

```
----------------------------------------------------------------
------------- OS+SS COMBINATION PERFORMANCES -------------
----------------------------------------------------------------

         Tagger        Tagging Rate           Raw Mistag                        Eff Mistag                          Tagging Power
            SPi      (2.6483 ± 0.1313)%    (52.0147 ± 1.5661)%    (43.3366 ± 0.0904(stat) ± 0.0000(cal)%    (0.0470 ± 0.0027(stat) ± 0.0000(cal)%
             OS     (36.2671 ± 0.3931)%    (52.5302 ± 0.4230)%    (36.1614 ± 0.0753(stat) ± 0.0000(cal)%    (2.7782 ± 0.0427(stat) ± 0.0000(cal)%
         SPi+OS      (1.9441 ± 0.1129)%    (48.9557 ± 1.8290)%    (34.2527 ± 0.3585(stat) ± 0.0000(cal)%    (0.1928 ± 0.0142(stat) ± 0.0000(cal)%
```

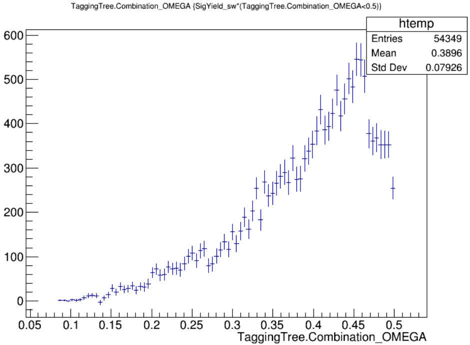
Calibrated branches file

- EspressoCalibratedOutput.root contains TaggingTree with calibrated branches for OS combination, SS pion, and overall tagger
- Branches are named

  <TaggerName>_DEC – Tagging decision; integer
  <TaggerName>_OMEGA – Calibrated mistag; double
  <TaggerName>_OMEGA – Tagging asymmetry; double
- This has the same number of entries as the input ROOT file and can be added as a friend tree
- Name of file can be changed with option CalibratedOutputFile; tree name is fixed to TaggingTree

Calibrated branches file (cont'd)

```
import ROOT
mainfile = ROOT.TFile("root://eoslhcb.cern.ch//eos/lhcb/wg/FlavourTagging/tutorials/CPTT_201
tagfile = ROOT.TFile("EspressoCalibratedOutput.root")
maintree = mainfile.Get("Bd2JpsiKSDetached")
tagtree = tagfile.Get("TaggingTree")
maintree.AddFriend(tagtree)
maintree.Draw("TaggingTree.Combination_OMEGA",
        "SigYield_sw*(TaggingTree.Combination_OMEGA<0.5)")
```



Future improvements

Ideas:

- Fix minor annoyances
- Propagate uncertainties to combination
- Support for more resolution models (next slide)
- A tool/instructions for integrating with `RooFit` in a *CP* analysis

Applying calibration to tagger in a ROOT file

Decay time resolution models

- Lifetime resolution only has an $\mathcal{O}(0.01\%)$ effect on B^0 calibrations:

$$e^{-\frac{1}{2}\Delta m_d^2 \sigma^2} = 0.999681 \text{ for } \sigma = 50\,\mathrm{fs}$$

- EPM calibrations to B_s decays (still experimental) require a precise lifetime resolution model:

$$e^{-\frac{1}{2}\Delta m_s^2 \sigma^2} = 0.679244 \text{ for } \sigma = 50\,\mathrm{fs}$$

- EPM supports a triple gaussian resolution model where width of each is a quadratic function of the "lifetime resolution" from the fitter
- *What other decay time resolution models, parameterized as a function of (t, σ_t), should the EPM support?*

Conclusion

- This tutorial has shown how to use the EPM for...
  - evaluating tagging performance on a signal sample
  - measure the characteristics of OS+SS combination
  - calibrating taggers on B^+ and B^0 data and MC
  - apply calibrations to ROOT files
- The GitLab Wiki contains more documentation
- Support for B_s calibration is a work-in-progress; needs validation
- We would appreciate your feedback regarding bugs, annoyances, and feature requests
  - Report issues on our JIRA page
  - or contact me at jack.wimberley@cern.ch