Background estimations for the lifetime

Marco Santimaria
26 Oct 2016

Bs->MuMu PID efficiencies

- Evaluated after our full selection and BDT>0.55
- runI and 2015 with ProbNNmu>0.2 (MC12Tunev2)
- 2016 with ProbNNmu>0.4 (MC15Tunev1)

```
Bs->MuMu runI PID Efficiency: 0.93922 +- 0.00035
```

```
Bs->MuMu 2015 PID Efficiency: 0.93207 +- 0.00044
```

```
Bs->MuMu 2016 PID Efficiency: 0.96887 +- 0.00022
```

B->hh

- Double misID per year

```
doumis_runI: 1.32e-05 +/- 2.77e-07 doumis_2015: 8.77e-06 +/- 3.54e-07   doumis_2016: 9.81e-06 +/- 9e-08
```

- For the BDTCalib i'm using:

```
BDTcalib_lifetime_runI = 0.5*BDTcalib_runI[4] + BDTcalib_runI[5] + BDTcalib_runI[6] + BDTcalib_runI[7] + BDTcalib_runI[8];
```

- This gives:

```
BDTcalib_runI = 0.435 +- 0.00732
BDTcalib_2015 = 0.443 +- 0.0124
BDTcalib_2016 = 0.443 +- 0.0124
```

NOTE: waiting for BDTCalib 2016 (small correction)

- Waiting for the justine factors for BDT>0.55, using our bin2 for now (expect ~ 5-10% correction):

```
// justine factors
Double_t jfact_runI[9] = {0, 0.939, 1.069, 1.079, 1.063, 1.051, 1.009, 0.945, 0.901};
Double_t jfact_Err_runI[9] = {0, 0.005, 0.008, 0.009, 0.009, 0.009, 0.008, 0.007, 0.007};
Double_t jfact_2015[9] = {0, 0.987, 1.106, 1.066, 1.019, 1.004, 0.978, 0.923, 0.886};
Double_t jfact_Err_2015[9] = {0, 0.010, 0.015, 0.018, 0.017, 0.016, 0.016, 0.014, 0.013};
Double_t jfact_2016[9] = {0, 0.966, 1.077, 1.068, 1.077, 1.029, 0.989, 0.931, 0.876};
Double_t jfact_Err_2016[9] = {0, 0.010, 0.015, 0.018, 0.018, 0.017, 0.016, 0.014, 0.013};
```

- results:

```
N(Bhh)_runI = 5.556 +/- 0.247 (4.447 %)
N(Bhh)_2015 = 0.689 +/- 0.081 (11.700 %)
N(Bhh)_2016 = 3.430 +/- 0.267 (7.785 %)
```

SL yields

```
SL = BdPiMuNu RESULTS:
SL_runI = 22.6874 +/- 1.1832 (5.21521 %)
SL_2015 = 2.94049 +/- 0.209675 (7.1306 %)
SL_2016 = 14.8758 +/- 0.791976 (5.32392 %)
```

```
SL = BsKMuNu RESULTS:
SL_runI = 4.85877 +/- 1.24796 (25.6847 %)
SL_2015 = 0.763276 +/- 0.19894 (26.0639 %)
SL_2016 = 3.50664 +/- 0.899101 (25.6399 %)
```

```
SL = LbPMuNu RESULTS:
SL_runI = 7.56639 +/- 2.08382 (27.5404 %)
SL_2015 = 1.97266 +/- 0.571722 (28.9822 %)
SL_2016 = 3.7976 +/- 0.971754 (25.5886 %)
```

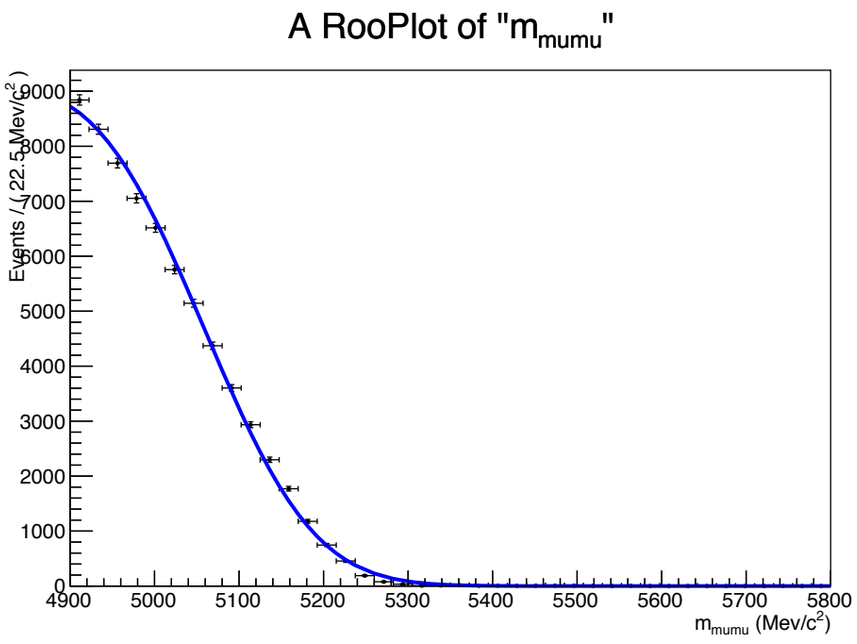
PID NOTE: runI calibration sample for protons is small, so, as we did for ProbNN4, we use ricci's proton table for runI he made with ProbNN2 (i.e. single fit in each P/PT bin)

```
SL = BuPiMuMu RESULTS:
SL_runI = 3.10473 +/- 0.442445 (14.2507 %)
SL_2015 = 0.518566 +/- 0.0763617 (14.7256 %)
SL_2016 = 2.40858 +/- 0.344934 (14.321 %)
```

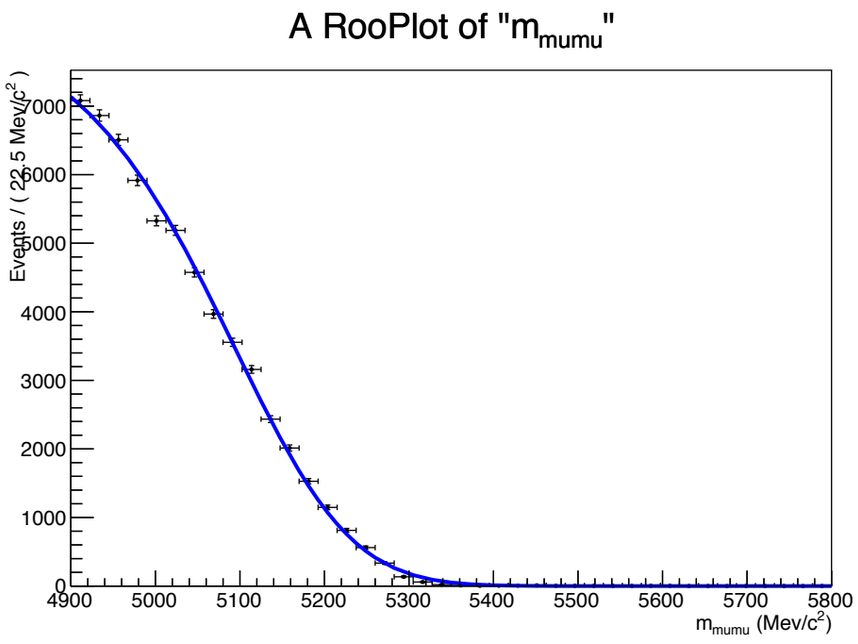
```
SL = BdPi0MuMu RESULTS:
SL_runI = 2.43196 +/- 1.02241 (42.0406 %)
SL_2015 = 0.430179 +/- 0.181548 (42.2028 %)
SL_2016 = 1.99833 +/- 0.840568 (42.0634 %)
```

```
SL = BcJpsiMuNu RESULTS:
SL_runI = 5.74713 +/- 1.29648 (22.5587 %)
SL_2015 = 0.70987 +/- 0.175003 (24.6528 %)
SL_2016 = 3.28523 +/- 0.802034 (24.4133 %)
```

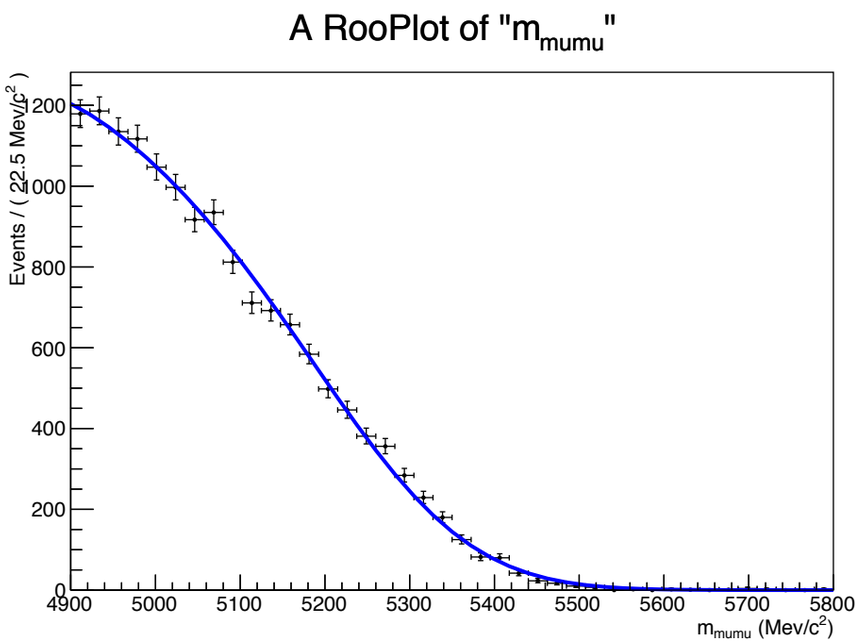
mass shapes



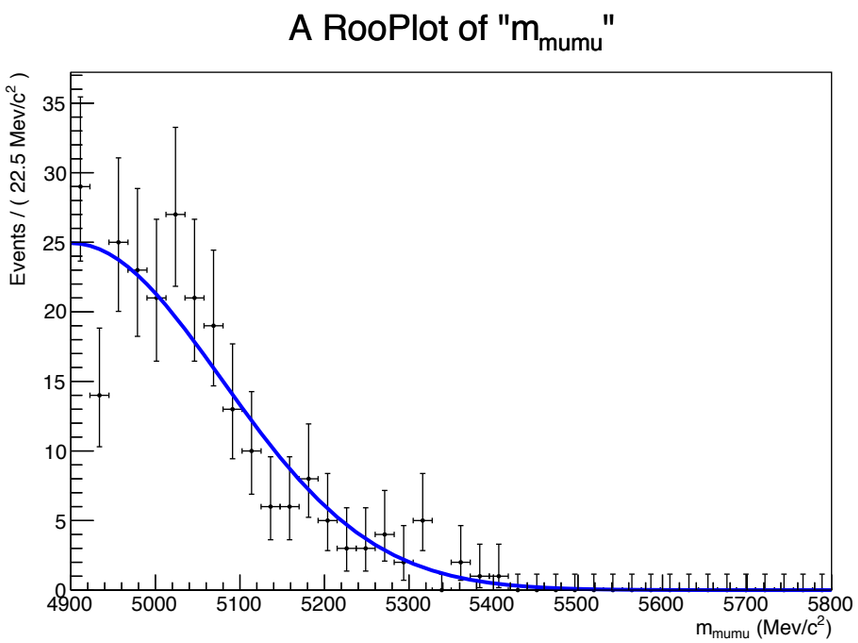
BdPiMuNu



BsKMuNu

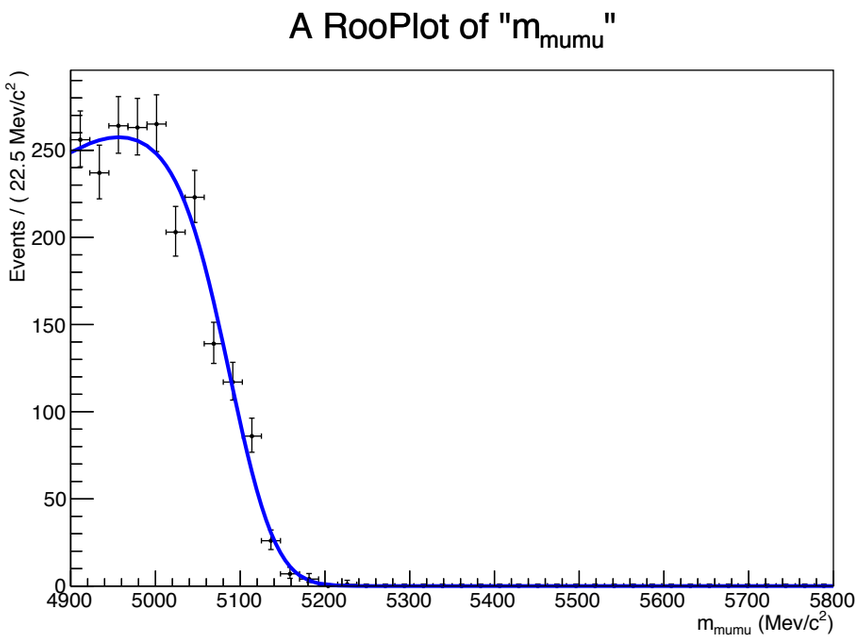


LbPMuNu

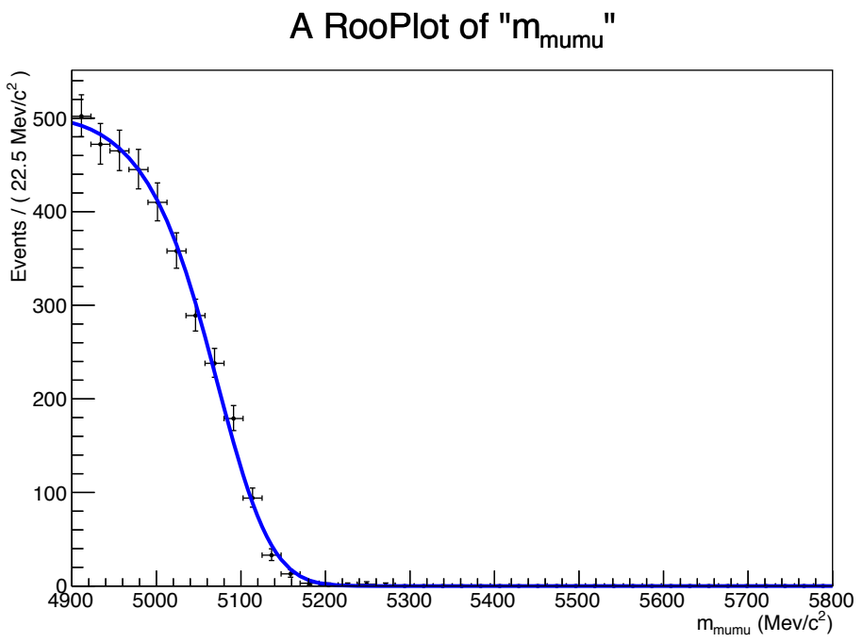


BcJpsiMuNu

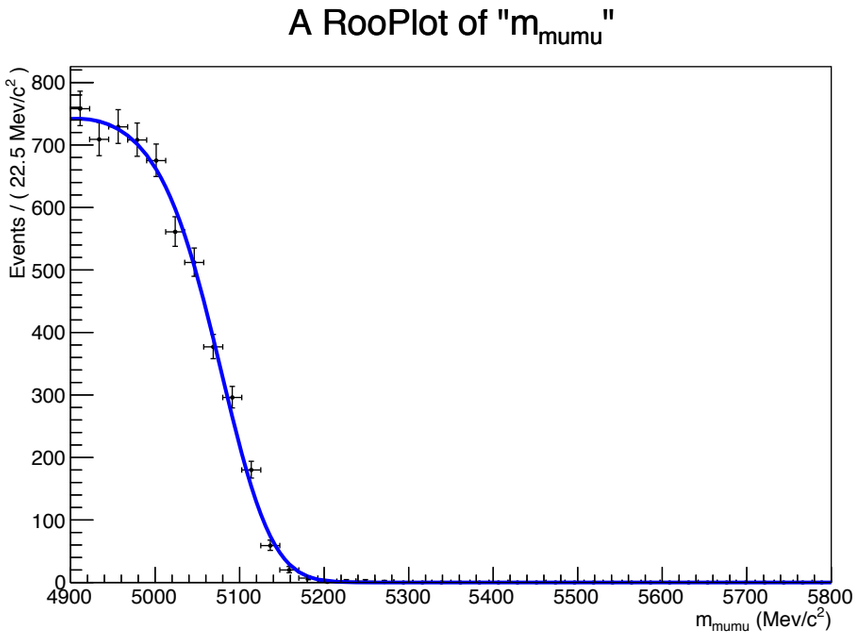
mass shapes



BuPiMuMu



BdPi0MuMu



optional: BuPiMuMu+BdPi0MuMu

mass shapes: fit model

- Fit model: argus-gaussian convolution. definition:

```
RooRealVar *m  = new RooRealVar("B_MM", "m_{mumu}", 4900, 6000, "Mev/c^{2}");

  RooRealVar *sigma = new RooRealVar("sigma", "sigma", 80, 30, 200);
  RooRealVar * delta = new RooRealVar("delta","delta",0,-10,10);
  RooGaussModel *gauss = new RooGaussModel("gauss", "gauss", *m, *delta, *sigma);

  RooRealVar *c = new RooRealVar("c","argus slope parameter",-10., -300., 0.) ;
  RooRealVar *m0 = new RooRealVar("m0","argus m0 parameter",5145, 4950, 5500);
  RooArgusBG *argus = new RooArgusBG("argus","Argus PDF", *m, *m0, *c) ;

  m->setBins(2000, "cache");
  RooFFTConvPdf *axg = new RooFFTConvPdf("axg","argus (X) gauss",*m, *argus, *gauss);
  axg->setBufferFraction(0.5) ;
```

- Parameter delta of the “gaussian resolution” is not fitted and fixed to 0

mass shapes: results

- Results: parameters of the ArgusXGaussian

```
Double_t c_BdPiMuNu_0 = -5.57693;
Double_t c_BdPiMuNu_Err_0 = 1.75339;
Double_t m0_BdPiMuNu_0 = 5092.22;
Double_t m0_BdPiMuNu_Err_0 = 7.38249;
Double_t sigma_BdPiMuNu_0 = 97.3443;
Double_t sigma_BdPiMuNu_Err_0 = 1.47615;
```

```
Double_t c_BsKMuNu_0 = -1.31986;
Double_t c_BsKMuNu_Err_0 = 0.629421;
Double_t m0_BsKMuNu_0 = 5150.5;
Double_t m0_BsKMuNu_Err_0 = 4.05387;
Double_t sigma_BsKMuNu_0 = 95.1211;
Double_t sigma_BsKMuNu_Err_0 = 1.21458;
```

```
Double_t c_LbPMuNu_0 = -0.010316;
Double_t c_LbPMuNu_Err_0 = 0.432211;
Double_t m0_LbPMuNu_0 = 5270.97;
Double_t m0_LbPMuNu_Err_0 = 10.5516;
Double_t sigma_LbPMuNu_0 = 124.668;
Double_t sigma_LbPMuNu_Err_0 = 3.49634;
```

```
Double_t c_BcJpsiMuNu_0 = -176.729;
Double_t c_BcJpsiMuNu_Err_0 = 104.366;
Double_t m0_BcJpsiMuNu_0 = 4921.57;
Double_t m0_BcJpsiMuNu_Err_0 = 241.598;
Double_t sigma_BcJpsiMuNu_0 = 177.403;
Double_t sigma_BcJpsiMuNu_Err_0 = 17.6452;
```

```
Double_t c_BuPiMuMu_0 = -8.99981;
Double_t c_BuPiMuMu_Err_0 = 2.32753;
Double_t m0_BuPiMuMu_0 = 5105.3;
Double_t m0_BuPiMuMu_Err_0 = 6.08585;
Double_t sigma_BuPiMuMu_0 = 39.2607;
Double_t sigma_BuPiMuMu_Err_0 = 3.23398;
```

```
Double_t c_BdPi0MuMu_0 = -5.26501;
Double_t c_BdPi0MuMu_Err_0 = 2.00628;
Double_t m0_BdPi0MuMu_0 = 5095.94;
Double_t m0_BdPi0MuMu_Err_0 = 5.60427;
Double_t sigma_BdPi0MuMu_0 = 45.3911;
Double_t sigma_BdPi0MuMu_Err_0 = 2.62847;
```

```
Double_t c_PiMuMu_0 = -6.64726;
Double_t c_PiMuMu_Err_0 = 1.50323;
Double_t m0_PiMuMu_0 = 5099.77;
Double_t m0_PiMuMu_Err_0 = 4.09399;
Double_t sigma_PiMuMu_0 = 43.0693;
Double_t sigma_PiMuMu_Err_0 = 2.01851;
```