Global Bandwidth Analysis

August 26, 2015

1 Introduction

This notebook is designed to visually walk through the steps to determine the optimal global bandwidth for a set of b-tagging, "MC → data" scale factors (SF) as well as the associated statistical and systematic uncertainties. It uses the local polynomial kernel estimator found in this package to "smooth" the distributions.

requirements: * `rootpy` * `ipython` * `CDIFiles` package * located here: `atlasoff/PhysicsAnalysis/JetTagging/JetTagPerformanceCalibration/CDIFiles/trunk` * `CalibrationDataInterface` package * located here: `atlasoff/PhysicsAnalysis/JetTagging/JetTagPerformanceCalibra` * needs to be compiled before running this notebook * `NPandSmoothingTools` package * located here: `atlasperf/CombPerf/FlavorTag/JetTagPerformanceCalibration/NPandSmoothingTools/trunk` * needs to be compiled before running this notebook

1.1 Common Imports

```
In [13]: import collections
         import ROOT
         import rootpy
         from rootpy.plotting import Hist, Canvas, Legend, set_style

         # now import RootCore stuff
         rootCore_import_result = ROOT.gROOT.Macro('$ROOTCOREDIR/scripts/load_packages.C')
         if rootCore_import_result != 0 and rootCore_import_result != 1:
             print "Couldn't import RootCore package libraries. Aborting..."
         else:
             from ROOT import Analysis
             from ROOT.Analysis import ROOTHistogramSmoother#, optimizeLeaveOneOutCrossValidation
             from ROOT.Analysis import CalibrationDataHistogramContainer
```

1.2 Custom Python wrappers

```
In [14]: # make Python friendly class wrapper
         class PyHistogramSmoother(ROOTHistogramSmoother):
             def __init__(self):
                 #ROOTHistogramSmoother.__init(self)__
                 super(PyHistogramSmoother, self).__init__()
                 # maps to cache the ROOT objects
                 self._call_args = {}
                 self._AddDataPoint_args = {}
                 self._SetDataPoint_args = {}

             # need this b/c ROOT has difficulty exporting the templated operator()
             def __call__(self, *args):
                 nargs = len(args)
                 c = None
                 if nargs == 1 and isinstance(args[0], collections.Iterable):
                     arg = args[0]
                     size = len(arg)
                     c = None
                     if not size in self._call_args:
                         self._call_args[size] = ROOTHistogramSmoother.Covariates_t(size)
                     c = self._call_args[size]
                     for i in range(size):
                         c[i] = arg[i]
                     return super(PyHistogramSmoother, self).__call__(c)
                 else:
                     if not nargs in self._call_args:
                         self._call_args[nargs] = ROOTHistogramSmoother.Covariates_t(nargs)
                     c = self._call_args[nargs]
                     for i in range(nargs):
                         c[i] = args[i]
                 return super(PyHistogramSmoother, self).__call__(c)
```

2 Demonstration of the Smoother and Cross Validation

2.1 Setup smoothing object

```
In [15]: smoother = PyHistogramSmoother()
         smoother.SetNumberOfDataPoints(0)
         smoother.SetDimension(1)
         smoother.SetOrder(1)
         smoother.SetBandwidth(0, 0.4)
         smoother.SetNbins(0, 100)
         smoother.SetScaleFunction(0, ROOT.TF1("ln scale", "TMath::Log(x)", -1.0, 1.0))
         smoother.SetInvScaleFunction(0, ROOT.TF1("invert ln scale", "TMath::Exp(x)", -1.0, 1.0))
         smoother.SetKernel(ROOT.TF1("gaus", "TMath::Gaus(x, 0, 1, kTRUE)"))
```

2.2 Setup plotting environment

```
In [16]: set_style('ATLAS')
         c = Canvas(width=800, height=600)
```

```
INFO:rootpy.plotting.style:using ROOT style 'ATLAS'
```

2.3 Dummy Data

```
In [20]: unsmoothed = Hist([20, 30, 50, 100, 150], title="unsmoothed p_{T}", legendstyle='pe')
         unsmoothed[1] = (1.1, 0.2)
         unsmoothed[2] = (1.2, 0.09)
         unsmoothed[3] = (1.01, 0.085)
         unsmoothed[4] = (1.09, 0.15)
```

```
In [21]: smoother.LoadData(unsmoothed)
         smoothed = smoother.MakeSmoothedTH1()
```

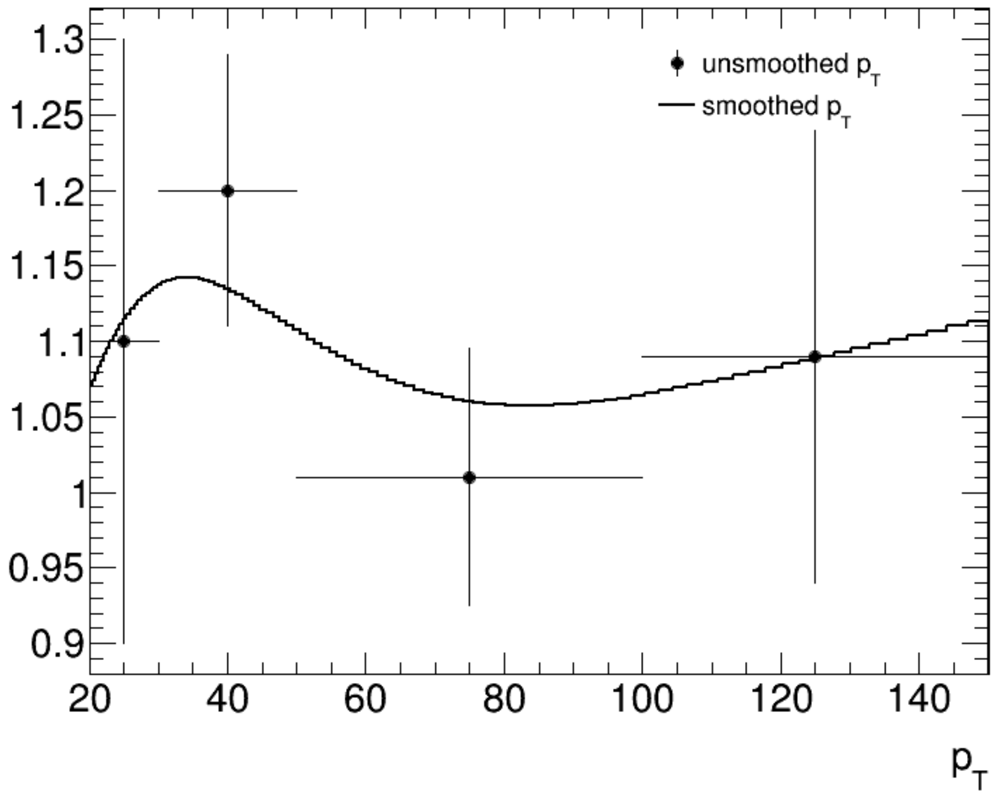
```
In [22]: unsmoothed.SetStats(False)
         unsmoothed.GetXaxis().SetTitle("p_{T}")
         unsmoothed.Draw()
```

```
smoothed.SetStats(False)
smoothed.SetTitle("smoothed p_{T}")
smoothed.Draw("same")

leg = Legend([unsmoothed], leftmargin=0.5,
             topmargin=0.04, rightmargin=0.15,
             textsize=20, entryheight=0.1)
leg.AddEntry(smoothed, label=smoothed.GetTitle(), style="l")
leg.Draw()

c
```

Out[22]:



In [ ]: