Using Goofit with Python

Himadri Pandey

Setup

```
pip install scikit-build cmake
pip install -v goofit
```

Features

- Simple installation like any Python module
- Easy setup - takes around 10-15 minutes in building files

Python Bindings with PyBind11

Python Bindings

Bind C++ PDFs in Python using PyBind 11

PyBind11

“A lightweight header-only library that exposes C++ types in Python and vice versa, mainly to create Python bindings of existing C++ code.”

Using Goofit in Python Examples

```
from goofit import *
```

Addition.py - Example for Adding Two PDFs

```
# Imports
from goofit import *
import numpy as np
import matplotlib.pyplot as plt

# Print GooFit and computer info
print_goofit_info()

# GooFit DataSet
xvar = Observable("xvar", -5, 5)
data = UnbinnedDataSet(xvar)

# Make a dataset from 90% gauss + 10% flat
dat = np.random.normal(0.2,1.1,100000)
dat[:10000] = np.random.uniform(-5,5,10000)
data.from_matrix([dat], filter=True)
```

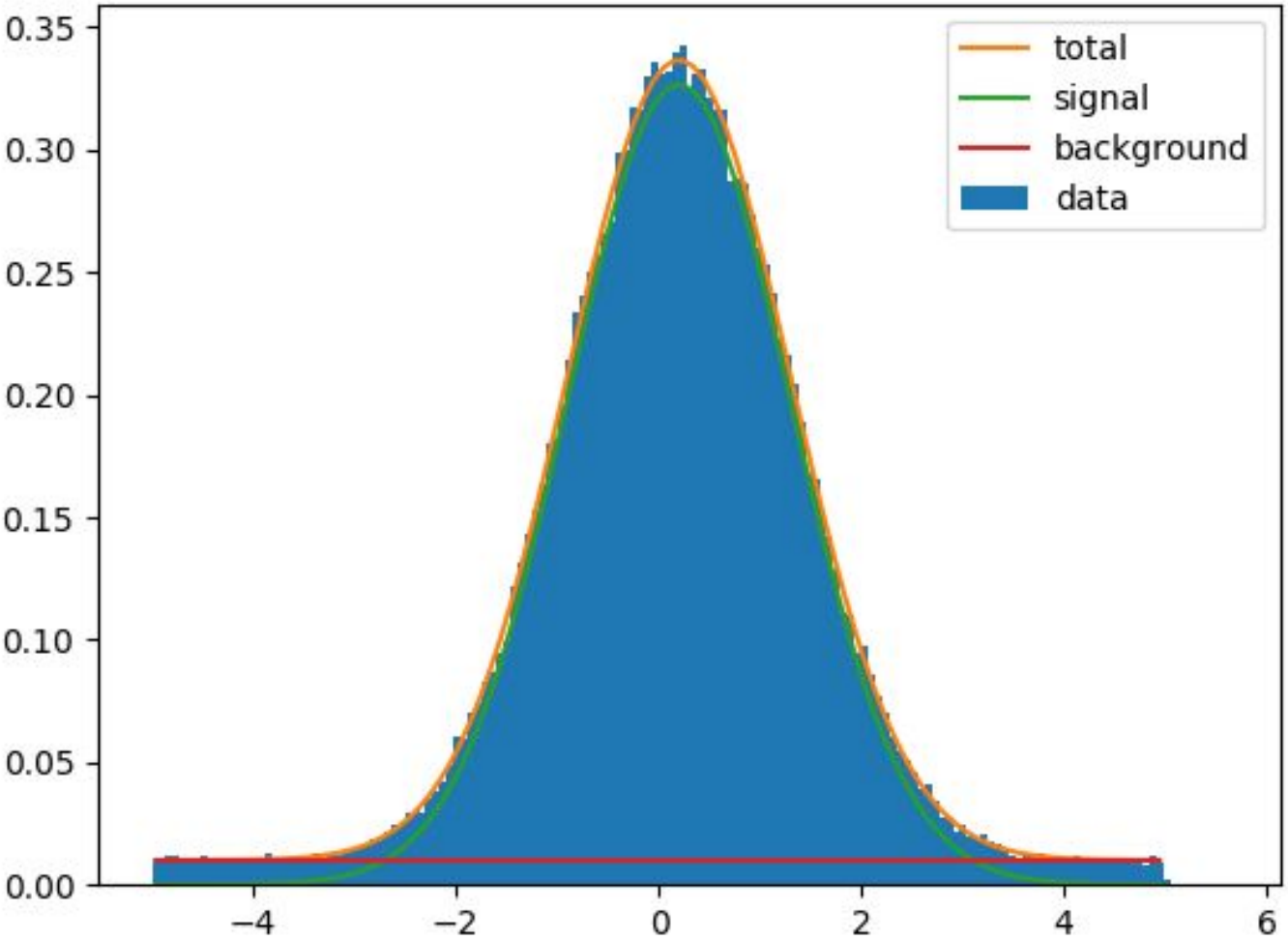
Addition.py

```
# GooFit PDFs and fitting variables
xmean = Variable("xmean", 0, 1, -10, 10)
xsigm = Variable("xsigm", 1, 0.5, 1.5)
signal = GaussianPdf("signal", xvar, xmean, xsigm)

constant = Variable("constant", 1.0)
backgr = PolynomialPdf("backgr", xvar, [constant])

sigfrac = Variable("sigFrac", 0.9, 0.75, 1.00)
total = AddPdf("total", [sigfrac], [signal, backgr])

# Do the fit
total.fitTo(data)
```

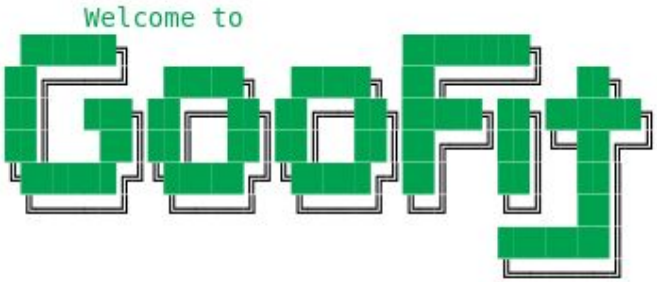
Addition.Py - Plotting Available in GooFit 2.1.1

```
# Plot data
plt.hist(dat, bins='auto', label='data', normed=True)

# Make grid and evaluate on it
grid = total.makeGrid()
total.setData(grid)
main, gauss, flat = total.getCompProbsAtDataPoints()
xvals = grid.to_matrix().flatten()

# Plotting components
plt.plot(xvals, main, label='total')
plt.plot(xvals, np.array(gauss)*sigfrac.value, label='signal')
plt.plot(xvals, np.array(flat)*(1-sigfrac.value), label='background')

# Show the plot
plt.legend()
plt.show()
```

Addition.py - Generated Plot



Result




```
MnSeedGenerator: for initial parameters FCN = 522066.7325869
MnSeedGenerator: Initial state:   - FCN =   522066.7325869 Edm =       185192 NCalls =
 17
VariableMetric: start iterating until Edm is < 0.0001
VariableMetric: Initial state    - FCN =   522066.7325869 Edm =       185192 NCalls =     17
VariableMetric: Iteration #   0 - FCN =   522066.7325869 Edm =       185192 NCalls =     17
VariableMetric: Iteration #   1 - FCN =   376648.6105226 Edm =      3943.49 NCalls =     34
VariableMetric: Iteration #   2 - FCN =   360521.5666858 Edm =      5807.81 NCalls =     47
VariableMetric: Iteration #   3 - FCN =   352534.4856178 Edm =      7510.83 NCalls =     60
VariableMetric: Iteration #   4 - FCN =   346873.7913056 Edm =      1176.23 NCalls =     71
VariableMetric: Iteration #   5 - FCN =   346087.7221632 Edm =      35.9524 NCalls =     81
VariableMetric: Iteration #   6 - FCN =   346039.8386063 Edm =      3.95918 NCalls =     91
VariableMetric: Iteration #   7 - FCN =   346035.4532991 Edm =     0.136684 NCalls =    101
VariableMetric: Iteration #   8 - FCN =   346035.2946238 Edm =  0.000453675 NCalls =    111
VariableMetric: Iteration #   9 - FCN =   346035.2941216 Edm =  1.28383e-06 NCalls =    121
VariableMetric: After Hessian    - FCN =   346035.2941216 Edm =  9.35221e-07 NCalls =    148
VariableMetric: Iteration #  10 - FCN =   346035.2941216 Edm =  9.35221e-07 NCalls =    148

Minuit did successfully converge.
# of function calls: 148
minimum function Value: 346035.2941216
minimum edm: 9.352210805897e-07
```

Result

```
minimum internal state vector: LAVector parameters:
    0.02017415908528
     0.1947783625819
    0.05009256018299
    -0.1965786116221

minimum internal covariance matrix: LASymMatrix parameters:
  1.2034046e-07  1.1078133e-09 -1.9815328e-16  3.0246173e-15
  1.1078133e-09  2.4994541e-05  3.6994703e-15  6.7913938e-14
 -1.9815328e-16  3.6994703e-15  9.0932142e-09  1.6764704e-10
  3.0246173e-15  6.7913938e-14  1.6764704e-10  7.5440973e-06


# ext. ||   Name    ||   type  ||     Value     ||  Error +/-

   0   ||     xmean || limited ||     0.2017279064378 ||0.002452462911705
   1   ||     xsigm || limited ||      1.096774546666 ||0.001734146556426
   2   ||     ymean || limited ||     0.5007161356331 ||0.0006734395298746
   3   ||     ysigm || limited ||     0.3011712534656 ||0.0004761924636327



The minimization took: 888.16 ms
Average time per call: 6.01 ms
xmean: 0.201728 +/- 0.00245246 [-10, 10] GooFit index: 16 Fitter index: 0
xsigm: 1.09677 +/- 0.00173415 [0.5, 1.5] GooFit index: 17 Fitter index: 1
ymean: 0.500716 +/- 0.00067344 [-10, 10] GooFit index: 18 Fitter index: 2
ysigm: 0.301171 +/- 0.000476192 [0.1, 0.6] GooFit index: 19 Fitter index: 3
```

Features

Available Resources

- 9 out of 13 C++ examples available in Python
- Support from Python and PyBind11 community on development in Python

Features

- Simple Setup
- Easy to use
- Adding python functionalities is trivial
- Accessibility to new new Python bindings created

Run Goofit Now on Your Own Computer

Setup

pip install scikit-build cmake
pip install -v goofit

Use it in your python code

from goofit import *

Questions?

Continued development on GooFit is supported by the U.S. National Science Foundation under grant number 1414736 and was developed under grant number 1005530. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the developers and do not necessarily reflect the views of the National Science Foundation. In addition, we thank the nVidia GPU Grant Program for donating hardware used in developing this framework.