dashi Documentation

Release 1.0

Eike Middell

February 10, 2017

Contents

Elaborate data analyses are possible with the functionality offered by the great numpy, matplotlib, and pytables libraries. However their support for HEP-typical problems (histograms, fitting routines,visualization of those,..) is limited. Dashi is intended to be a thin wrapper around these libraries to provide some useful tools for these problems without obstructing the user the access to the underlying libraries and without being a dependency sink.

CHAPTER 1

Installation

The easiest way to install dashi is with *pip*:

```
pip install https://github.com/emiddell/dashi/zipball/master
```

CHAPTER 2

API Reference

2.1 Dashi API Documentation

dashi

2.1.1 dashi.histogram

2.1.2 dashi.histfactory

2.1.3 dashi.fitting

2.1.4 dashi.objbundle

CHAPTER 3

Documentation

3.1 Object Bundles

Object bundles are intended to simplify the work with sets of similar objects. They are basically dictionaries with the additional feature, that method calls on the bundle are executed on each object in the bundle.

```
1
2  import numpy as n
3  import dashi as d
4
5  x = n.array( [1,2,3,4, 5] )
6  y = n.array( [6,7,8,9,10] )
7
8  bundle = d.bundle( x=x, y=y )
9
10 x.sum()
11 y.sum()
12
13 bundle.sum()
```

Numpy arrays are a very useful container for the data that one wants to analyse. Because of the element-wise operation of operators and functions calculations are easily formulated and fast. Hence, most of my analysis scripts hold the data in memory and stored in numpy arrays. Multiple variables or classes of measurements can be distinguished, by keeping the arrays in dictionaries

Object bundles are intended to solve two problems:

-

The element-wise operation of operators and functions on numpy arrays make them a very handy tool for calculations. For example and can add two arrays and the operation will add the respective items from each array.

One can reapply this concept one level higher and apply can add two one can e.g add two numbers

CHAPTER 4

Examples

4.1 examples for 1d histograms

```
import numpy as n
import dashi as d
import pylab as p

d.visual()

data1 = n.random.normal(2,2,1e3)
data2 = n.random.normal(5,4,1e3)
data = n.hstack((data1,data2))
h = d.factory.hist1d( data, 40, label="x")

p.figure(figsize=(8,4))

p.subplot(131)
h.line()

p.subplot(132)
h.scatter()

p.subplot(133)
h.band()
```

```
import numpy as n
import dashi as d
d.visual()

h = d.factory.hist1d( n.random.normal(2,2,1e5), n.linspace(-15,15,101), label="x")
h.line()
h.statbox()
```

```
import pylab as p
import numpy as n
import dashi as d
d.visual()

h1 = d.factory.hist1d( n.random.normal(2,2,1e5), n.linspace(-15,10,101), label=r"$\sqrt{x}$", title="
h1.line(filled=1, fc="r", alpha=.2)
h2 = d.factory.hist1d( n.random.normal(-1,2,1e5), n.linspace(-15,10,101))
h2.line(filled=1, fc="b", alpha=.2)
```

```
h1.statbox(loc=1)
h2.statbox(loc=2)
p.ylim(0,10000)
```

4.2 examples for 2d histograms

```
import numpy as n
import dashi as d
import pylab as p

d.visual()

x = n.random.normal(2, 2, 1e5)
y = n.random.normal(-1, 3, 1e5)

h = d.factory.hist2d( (x,y), 100, labels=("x", "y"))

p.figure(figsize=(8,8))

p.subplot(221)
h.imshow()
cb = p.colorbar()
cb.set_label("bin count")

p.subplot(222)
h.imshow(log=1)
cb = p.colorbar()
cb.set_label("log10(bin count)")

p.subplot(223)
h.contour(filled=1)
cb = p.colorbar()
cb.set_label("bin count")

p.subplot(224)
h.contour(filled=0, levels=[.5,1,1.5], log=1,clabels=1)
cb = p.colorbar()
cb.set_label("log(bin count)")
p.xlim(-5,10)
p.ylim(-10,10)
```

```
"""
 use of 2d histograms for correlation studies
"""
import numpy as n
import dashi as d
import pylab as p

d.visual()

npoints = 1e5
y = n.random.uniform(0,10,npoints)
x = 2 * y + 3 + n.random.normal(0,2,npoints)

h = d.factory.hist2d((x,y),
                     (n.linspace(0,30,101), n.linspace(0,10,11)),
```

```
                      labels=("x","y")
                     )

p.figure(figsize=(9,9))
p.subplots_adjust(wspace=.25)

# simple imshow of bincontent array
p.subplot(221)
h.imshow()

# profile plots. project on dimension 1 ("y")
p.subplot(222)
scatterpoints = d.histfuncs.h2profile(h,dim=1)
scatterpoints.scatter()

# stacked 1d histograms plots
# keeping the overall shape of the projected distribution
p.subplot(223)
h.stack1d()

# normalizing each bin to 1 to emphasize relative contributions
p.subplot(224)
h.stack1d(boxify=1)
```

4.3 examples of fitting arbitrary x-y data points

```
"""
   fitting a straight line
"""
import dashi as d; d.visual()
import numpy as n
import pylab as p

x = n.linspace(-10,10,21)
y = 2*x + 3 + n.random.normal(0,1,len(x))

mod = d.poly(1)
mod = d.fitting.leastsq(x,y,mod)
p.errorbar(x,y,yerr=1,fmt="ko",linestyle="none", capsize=0)
p.plot(x, mod(x), "r--")
mod.parbox(loc=2)
```

```
"""
   fitting a  parabola using dashi.scatterpoints.points2d as
   a data container
"""
import dashi as d; d.visual()
import numpy as n
import pylab as p

yerr=10.
data = d.scatterpoints.points2d()
data.x = n.linspace(-10,10,21)
data.y = 2*(data.x-2)**2 - 3 + n.random.normal(0,yerr,len(data.x))
data.yerr = yerr *  n.ones(len(data.x)) # initial guess for the error
```

```
mod = d.poly(2)
mod = d.fitting.leastsq(data.x,data.y,mod, chi2values=True)

p.figure(figsize=(9,4))
p.subplots_adjust(wspace=.25)

p.subplot(121)
data.scatter(fmt="ko", ms=3)
p.plot(data.x, mod(data.x), "r--")
mod.parbox(loc=1)

ax = p.subplot(122)
p.plot(mod.chi2values[0], mod.chi2values[1], "k-")
p.text(0.1,0.9, "$\chi^2/ndof = %.2f/%d$" % (mod.chi2, mod.ndof), transform = ax.transAxes)
p.ylabel("chi2 contribution")
```

4.4 examples of fitting histograms

```
"""
 example of fitting a more elaborated model (sum of two gaussians)
"""
import numpy as n
import dashi as d
import pylab as p
import scipy.stats
d.visual()


# create random numbers from two normal distributions
x1 = n.random.normal(2,2,1e4)
x2 = n.random.normal(0,6,1e4)

# histogram each
h1 = d.factory.hist1d(x1,n.linspace(-50,20,101))
h2 = d.factory.hist1d(x2,n.linspace(-50,20,101))

# use histogram arithmetic to combine them
h = h1 + h2

# Define the function that should be fitted to (h1+h2).
# The fist argument is interpreted as the point where
# the function should be evaluated (more dimensional
# functions are possible by using tuples).
# All other parameters are taken as fit parameters
# and should get descriptive names.

def twogauss(x, amp1, mean1, sigma1, amp2, mean2, sigma2):
    return (amp1 * scipy.stats.norm(loc=mean1, scale=sigma1).pdf(x) +
            amp2 * scipy.stats.norm(loc=mean2, scale=sigma2).pdf(x) )
def twogauss_int(x, amp1, mean1, sigma1, amp2, mean2, sigma2):
    return (amp1 * scipy.stats.norm(loc=mean1, scale=sigma1).cdf(x) +
            amp2 * scipy.stats.norm(loc=mean2, scale=sigma2).cdf(x) )

# wrap into a dashi.fitting.model and perform a least square fit
mod = d.fitting.model(twogauss,twogauss_int)
# first guesses
```

```
mod.params['amp1'], mod.params['amp2'] = (h.stats.weightsum/2.,)*2
mod.params['mean1'] = -1
mod.params['mean2'] = 1

h.leastsq(mod)


# plot histograms
h.scatter(c="k")
h1.line(c="#777777")
h2.line(c="#555555")
h1.statbox(loc=6)
h2.statbox(loc=3)

# after fitting mod behaves like a normal function with the fitted
# parameters fixed
p.plot(h.bincenters, n.diff(mod.integral(h.binedges)), "r-", linewidth=2,alpha=.5)
mod.parbox(loc=2)
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search