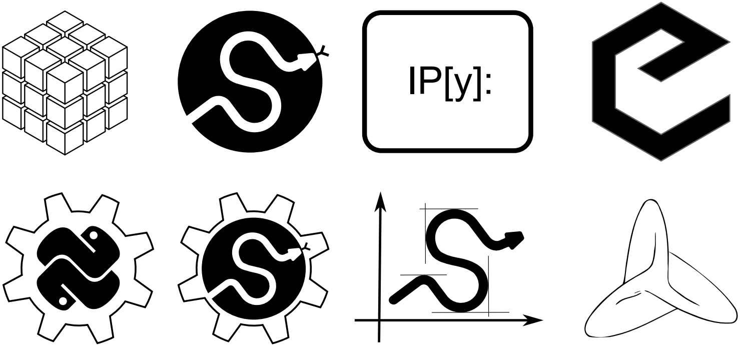
SciPy

Kai Dührkop

Lehrstuhl fuer Bioinformatik
Friedrich-Schiller-Universitaet Jena
`kai.duehrkop@uni-jena.de`

11.-15. August 2014

SciPy



- scientific libraries for python
- plotting, numerical and statistical analysis, machine learning, interactive shell, ...

Numpy

- defines high efficient matrix operations and data structures
- **numpy.array** datatype for n-dimensional numerical arrays. Operations on arrays are done componentwise (no map necessary)
- **numpy.matrix** datatype for matrices. Very similar to arrays, but some operations behave differently (e.g. multiplication is matrix product)

arrays and matrices

```
import numpy
vector = numpy.array([1,3,2,5])
vector + 2 #=> numpy.array([3,5,4,7])
twodim = numpy.array([[1,2], [4,5], [7,4]])
matrix = numpy.matrix([1,4,2])
```

- multidimensional array slicing is done in a single bracket
- slicing accepts index arrays and booleans!

arrays and matrices

```
data = numpy.array([[1,2,4], [4,0,5], [7,4,7]])

# get 2nd row and 3rd column
data[1, 2] #=> 5

# we are only interested in the first
# and third column
data[:, (0,2)]

# all rows which contain no zero
data[product(data,0)!=0,:]
```

- array operations (sum, product, mean, var, ...) can be done one the whole matrix or column- or row-wise
- optional axis parameter can be set to 1 (row-wise) or 0 (column-wise)

arrays and matrices

```
data = numpy.array([[1,2,4], [4,0,5], [7,4,7]])
# sum of all entries in the matrix
sum(data) #=> 1+2+4+4+0+5+7+4+7

# sum of all rows
sum(data, axis=0) #=> [1+2+4, 4+0+5, 7+4+7]

# sum of all columns
sum(data, axis=1)#=> [1+4+7, 2+0+4, 4+5+7]
```

arrays and matrices

```
# compute eigenvalues
eigenvalues, eigenvectors = eig(someMatrix)

# compute covariance matrix
covariance = cov(X)

# transpose matrix
A = B.transpose()

# dot product A x A^T
X = A.dot(A.transpose())

# dimension of the array/matrix
X.shape #=> (4,4)
```

SciPy

distributions

```python
from scipy.stats import norm

# create normal distribution with
# mean=5 and standard deviation=2
N = norm(loc=5, scale=2)

# get cumulative probability
N.cdf(2)

# get probability density for many points
N.pdf([1, 5, 7])
```

SciPy

distributions

```
from scipy.stats import pareto

# create pareto distribution with
# b=1 and k=2
P = pareto(b=1, scale=2)

# generate 100 random values
values = P.rvs(100)

# learn the distribution parameters from data
params = pareto.fit(values)
Plearned = pareto(*params)
```

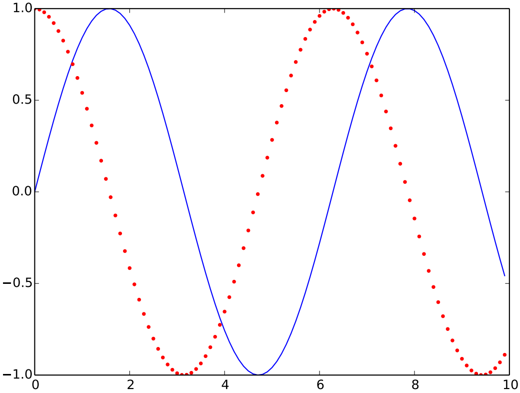
- Provides a huge set of distribution
- common interface: all distributions usually provide the same methods
- can learn distribution parameters (maximum likelihood)
- big library for statistical tests

matplotlib

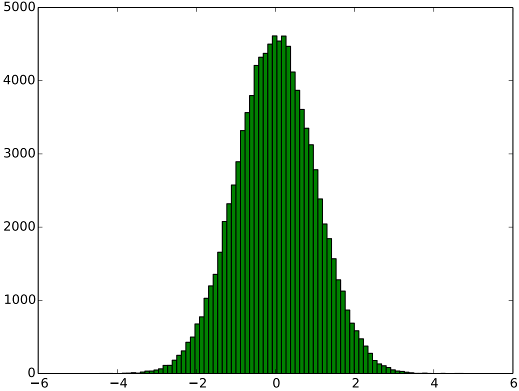
- plotting library
- designed for use in shell (but can be used also in script file)
- see examples in http://matplotlib.org/

arrays and matrices

```
import matplotlib.pyplot as plt
import numpy
xseq = numpy.array(range(100)) / 10.0
# 2D plot
plt.plot(xseq, numpy.sin(xseq), "b",
         xseq, numpy.cos(xseq), "r.")
plt.clf() # clear figure
# histogram
plt.hist(randn(100000), 80)
```



```
xseq = numpy.array(range(100)) / 10.0
plt.plot(xseq, numpy.sin(xseq), "b",
         xseq, numpy.cos(xseq), "r.")
```



```
plt.hist(randn(100000), 80)
```

other helpful functions

```
# add a title
title("Some␣Plot␣Title")

# add some text in LaTex style!
text(xpos, ypos, "$\lambda_x=\mu$"

# save graphic in arbitrary format
savefig("myfigure.pdf")
```