Python

und statistische Methoden der Datenanalyse

Python-Grundlagen

- moderne Hochsprache
  - unterstützt Skripting (Prozeduren u. Funktionen)
  - objektorientiet (Klassen)
  - Funktionale Programmierung (z.B. List-Comprehension)
- “Batteries included”
  - iPython (komfortable Interpreter)
  - NumPy/SciPy (Vektor/Matrixoperationen, Sammlung wissenschaftlicher Routinen)
  - MatPlotLib (2D Datenvisualisierung)

Dokumentation

- Python Homepage
- Python Tutorial
- Python Einführung (deutsch)
- NumPy/SciPy homepage
- SciPy Cookbook
- MatPlotLib Homepage
- Übungsmaterial

Computerpool Physik

- python, ipython, numpy, scipy and matplotlib ist berteits installiert
- start mit

```
ipython -pylab
```

(beim ersten Start werden Konfigurationsdateien angelegt)

```
In [1]: print "Hallo Welt"
Hallo Welt

In [2]: Exit
```

- verlassen mit “Exit” oder “CTRL-D”

Ein erstes Beispiel

- Fibonacci Folge

Try

```
In [1]: a, b = 0, 1

In [2]: while b < 10:
   ...:     print b,
   ...:     a, b = b, a+b
   ...:
1 1 2 3 5 8
```

- Blöcke durch Einrückung
- Mehrfachzuweisung möglich

Python Basics

```
In [1]: from math import * # define 'sin' etc.

In [2]: def f(x,y):              # function definition
   ...:     if x<0 :
   ...:         x = -x
   ...:     h = sin(x)*sin(y)
   ...:     return h
   ...:

In [3]: f(1.,1.)
Out[3]: 0.708073418274
```

- Lambda functions (one line functions)

```
In [12]: g = lambda x,y : sin(x)*sin(y)

In [13]: g(1.,1.)
Out[13]: 0.708073418274
```

Try

Listen und for-Schleifen

- Listen durch “[“ und “]”
- for-Schleifen laufen über Listen

Try

```
In [1]: liste = ['apple','grapefruit', 'banana']

In [2]: for item in liste:
   ...:     print item,
   ...:
apple grapefruit banana
```

- Integer-Listen durch “range(start,stop,spep)”

```
In [3]: range(10,20)
Out[3]: [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

List Comprehension

- for-Schleife über Integer mit “range(...)”

```
In [4]: i2 = []    # create empty list
In [5]: for i in range(10):
   ...:     i2.append(i*i)
   ...:
In [6]: print i2
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- kompakter durch List Comprehension

```
In [7]: [i*i for i in range(10)]
Out[7]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Try

Laden von Modulen / Hilfe

Laden von Modulen

```
In [8]: import pylab # names in module scope

In [9]: pylab.plot([1,2,3])

In [10]: from pylab import * # names in global scope

In [11]: plot([3,2,1])
In [11b]: show()    # not needed if ipython "-pylab"
```

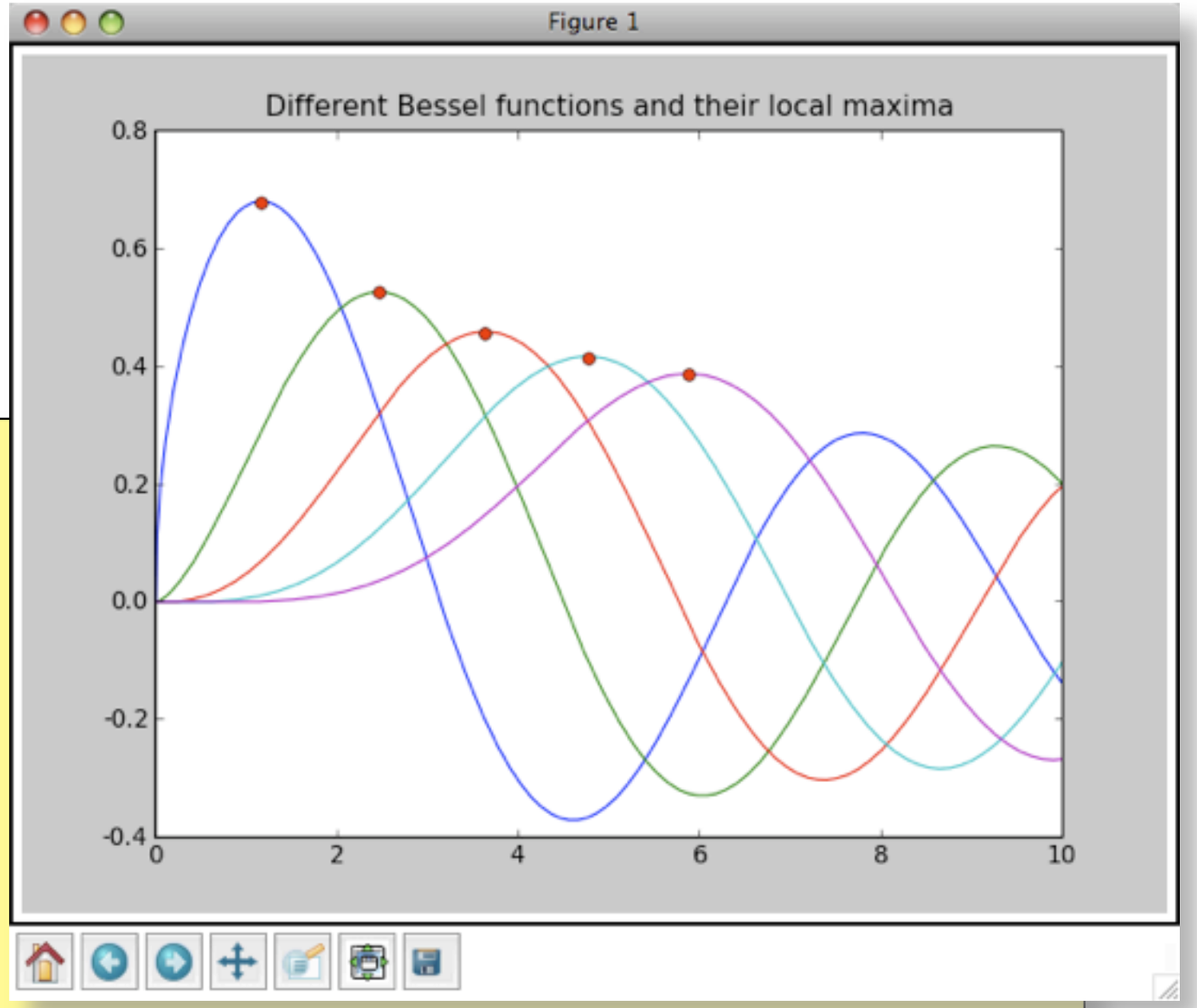
Hilfe

```
In [12]: help(plot) # python standard help system
In [13]: plot?      # enhanced help system
In [14]: plot??     # print source code if available

In [15]: %magic     # ipythons 'magic' functions
```

Py4Science

- NumPy, SciPy, PyLab

```
from scipy import *
from numpy import *
from pylab import *

x = arange(0,10,0.01)

for k in arange(0.5,5.5):
      y = special.jv(k,x)
      plot(x,y)
      f = lambda x: -special.jv(k,x)
      x_max = optimize.fminbound(f,0,6)
      plot([x_max], [special.jv(k,x_max)],'ro')

title('Different Bessel functions and their local maxima')
show()
```



NumPy

Matrix- und Array-Operationen

(array, zeros, ones, arange, linspace, random.*)

```
In [1]: from numpy import *
                                        # create arrays
In [2]: a = array([1.,3.,7.])           # - from python list
In [3]: b = arange(3.)+1                # - from range
In [4]: r = random.uniform(size=3)      # - from random numbers
In [5]: print a, b, r
[ 1.  3.  7.] [1. 2. 3.] [ 0.81470576 0.21422178 0.79054738]

In [6]: a*b                             # vector product
Out[6]: array([  1.,   6.,  21.])

In [7]: a**2                            # square
Out[7]: array([  1.,   9.,  49.])

In [22]: floor(3.*r+1)                  # truncate to integer
Out[22]: array([ 3.,  1.,  3.])
```

SciPy I - Statistik Modul

```
In [1]: from scipy import *

In [2]: f = stats.poisson(3.0)  # define poisson object
In [4]: f.pmf(arange(10))       # probability mass function
Out[4]:
array([ 0.04978707,  0.14936121, ... ])

In [14]: r = random.normal(size=10000) # inherited from numpy

In [15]: stats.mean(r)          # statistic function/test
Out[15]: 0.0119397312522        # for help call 'stats?'

In [16]: stats.median(r)
Out[16]: 0.019663695948

In [18]: stats.kurtosis(r)
Out[18]: 0.00738373607439
```

MatPlotLib - Graphik

- Easy 2D Graphik Modul
- Similar to MatLab syntax
- hist - Histogramm

Try

```
> ipython -pylab
```

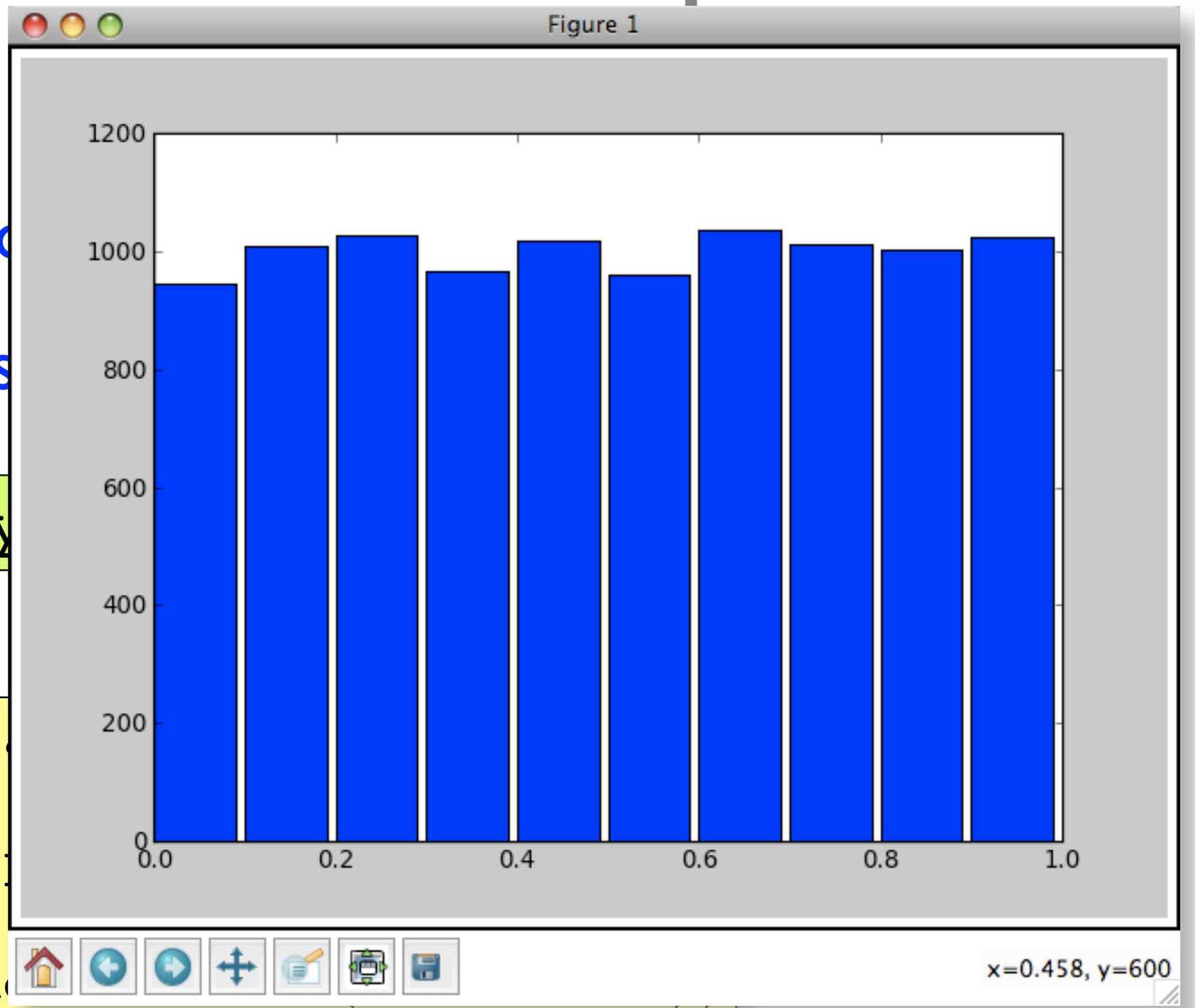
```
In [1]: from pylab import *

In [2]: from scipy import *

In [3]: hist(random.uniform(size=10000))
```

MatPlotLib - Graphik

- Easy 2D
- Similar to
- hist - His

```
> ipython -py
```

```
In [1]: from pyl

In [2]: from sci

In [3]: hist(ran
```



MatPlotLib - Graphik

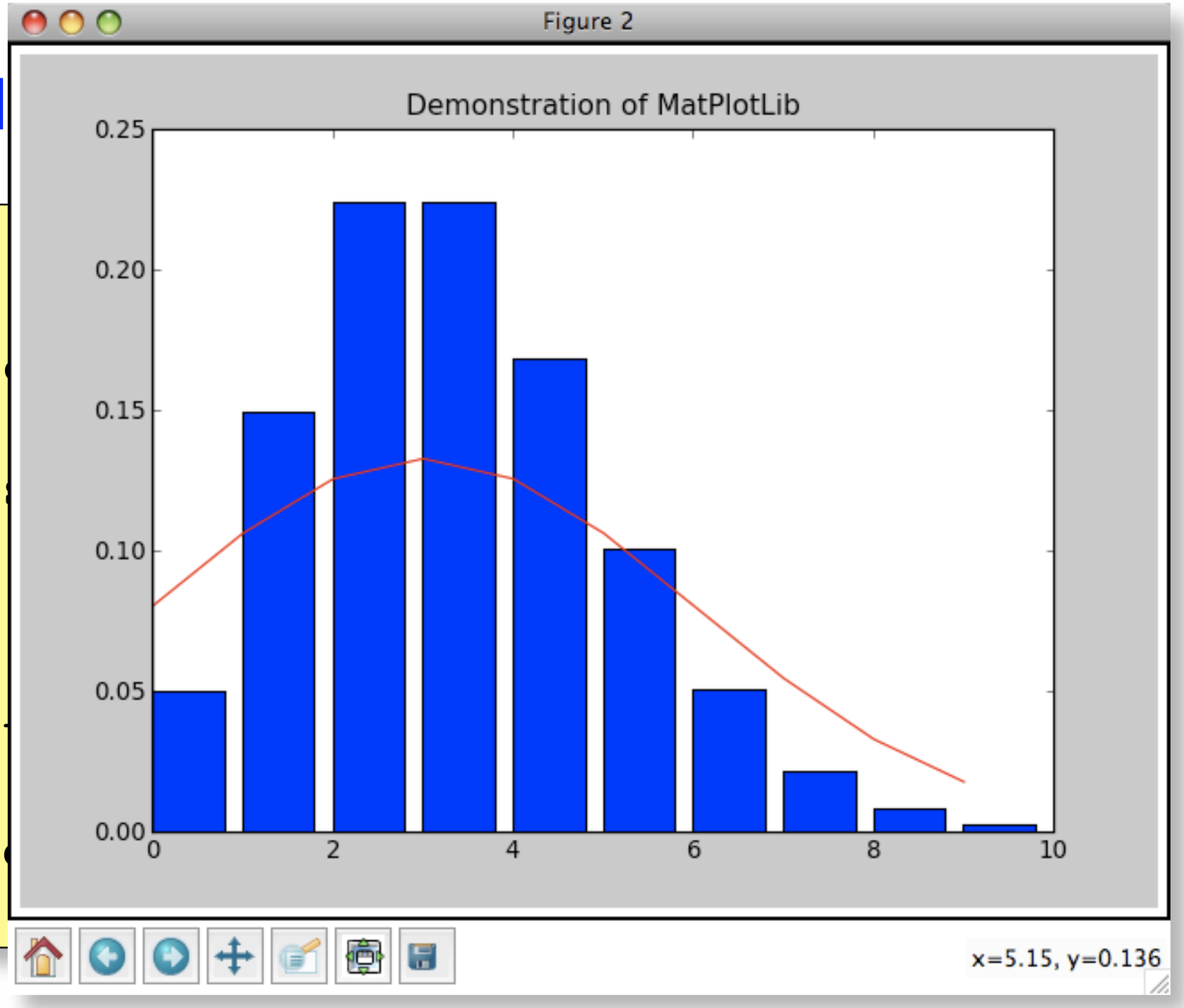
- box, plot, ... title, xlable, ylable

```
In [4]: figure()                    # create new figure

In [5]: x = arange(10)              # define x-values

In [6]: y = stats.poisson.pmf(x,3.0)
                                    # calc y-vaules
In [7]: bar(x,y)                    # make bar plot

In [8]: plot(x,stats.norm.pdf(x,3.0,3.0),color='red')
                                    # draw Gauss function
In [9]: title('Demonstration of MatPlotLib')
                                    # set title
```

MatPlotLib - Graphik

- box, plot, ... titl

```
In [4]: figure()

In [5]: x = aran

In [6]: y = stat

In [7]: bar(x,y)

In [8]: plot(x,s

In [9]: title('D
```



Additional Moduls

- Viele, viele ..... http://pypi.python.org
  - PyROOT - einfach zu verwendener ROOT-Wrapper
  - PyMinuit - Interface zum C++ Minuit
  - PyTables - IO im HDF5-Dateiformat
  - .....

Advanced Information

- running scripts from python

```
> python myScript.py
```

- running scripts from ipython
  - on the command line

```
> ipython myScript.py
```

  - from the ipython prompt

```
In [1]: run myScript.py
```

Advanced Tips

- for-Schleifen über mehrere Variablen
  - mit `zip`

```
a = []
xVal, yVal = [1, 2, 4] , [1,3,9]
for x,y in zip(xVal,yVal):
    a.append(x*y)
```

  - mittels indexing

```
for i in range(3):
    a.append(xVal[i]*yVal[i])
```