
https://blog.laimoon.com/wp-content/uploads/2018/09/technology2.jpg

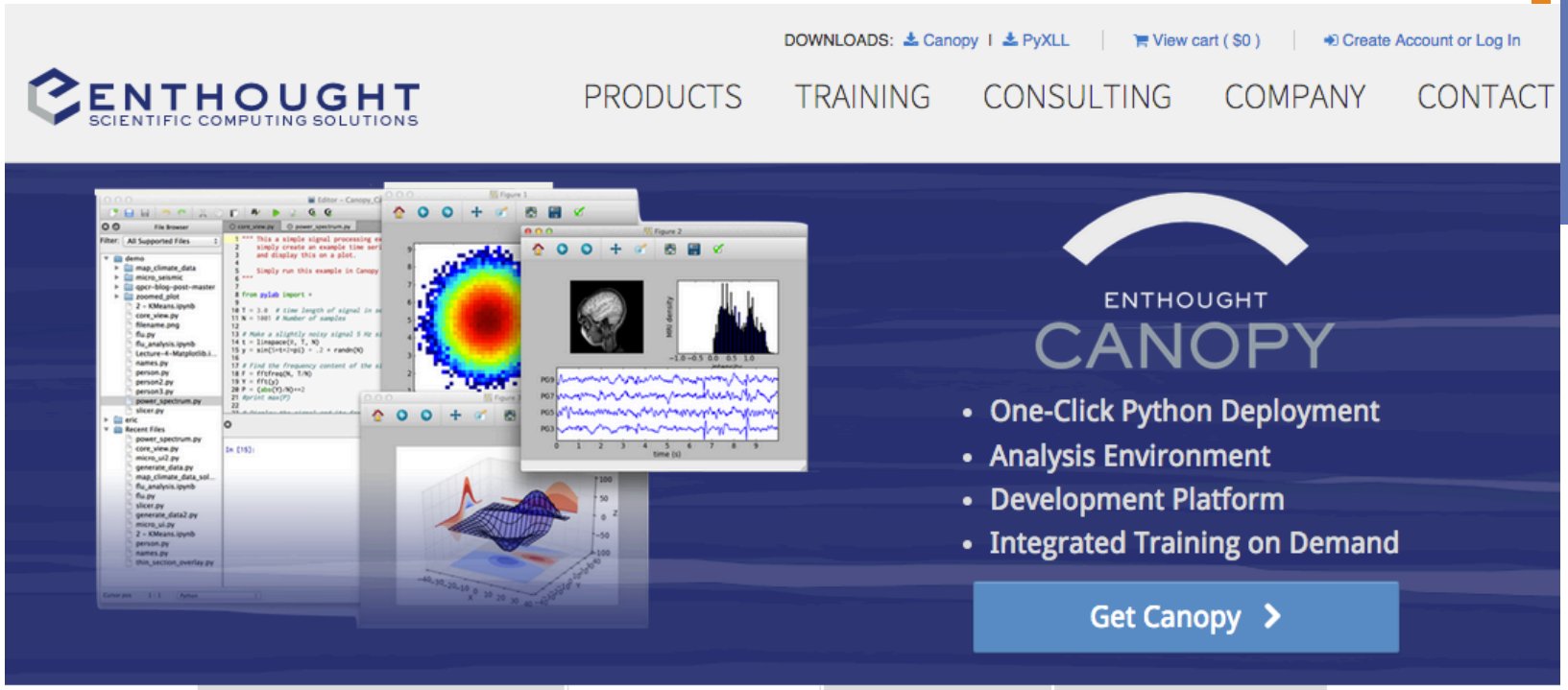
Tópicos Especiales I

Clase 7: Numerical Python, Matplotlib, SciPy

Javier Sanchez Galan, PhD
Facultad de Ingeniería de Sistemas Computacionales
3/octubre/2019

De la Clase Anterior

Python Compiler

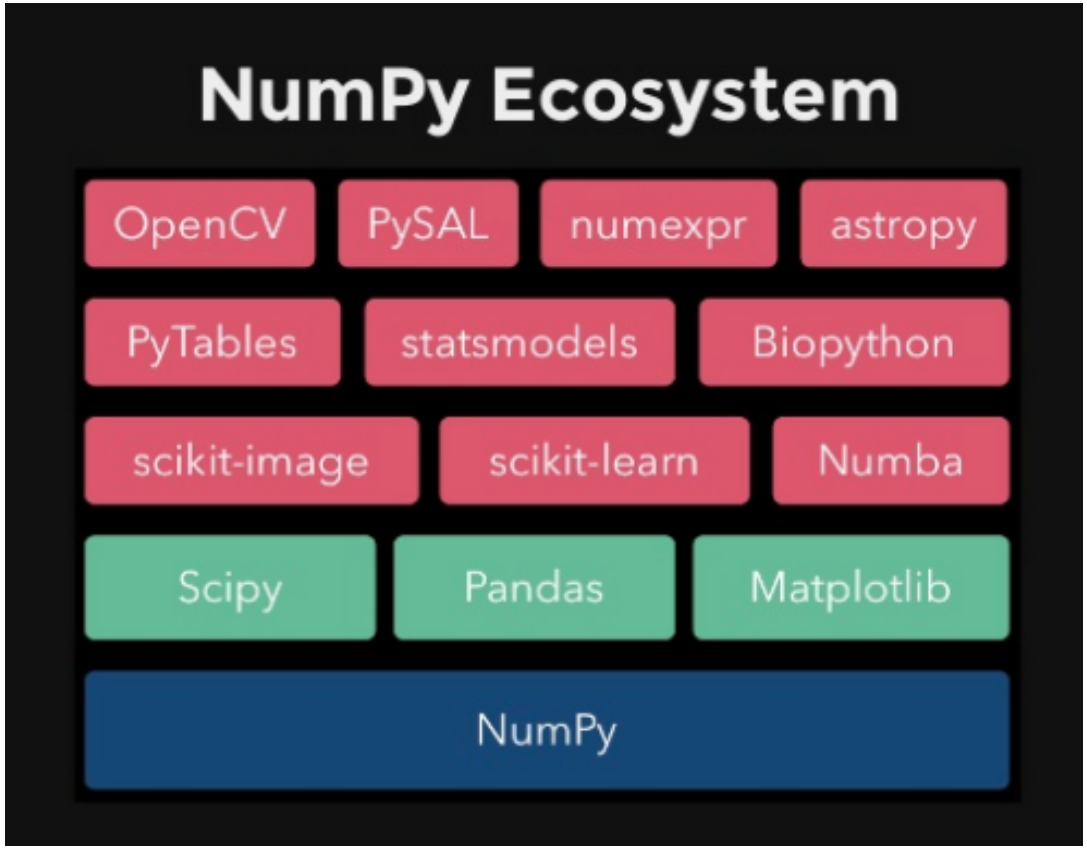


NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

NumPy

NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted[citation needed], and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.



+ Numpy

Making a Matrix

```
import numpy as np
A = np.zeros((5,5))
```

A 5x5 matrix full of zeros.

```
import numpy as np
A = np.random.random((10,10))
```

A 10x10 matrix full of random numbers.

Making a Matrix

```
 1 import numpy as np
 2
 3 a=np.array([])
 4 print a
 5
 6 a=np.array([1,2,3,4,5,6,7,8,9])
 7 print a
 8
 9 b=a.reshape((9,))
10 print b
11
12 b=a.reshape((3,3))
13 print b
14
15 print b*10+4
```

Making a Matrix

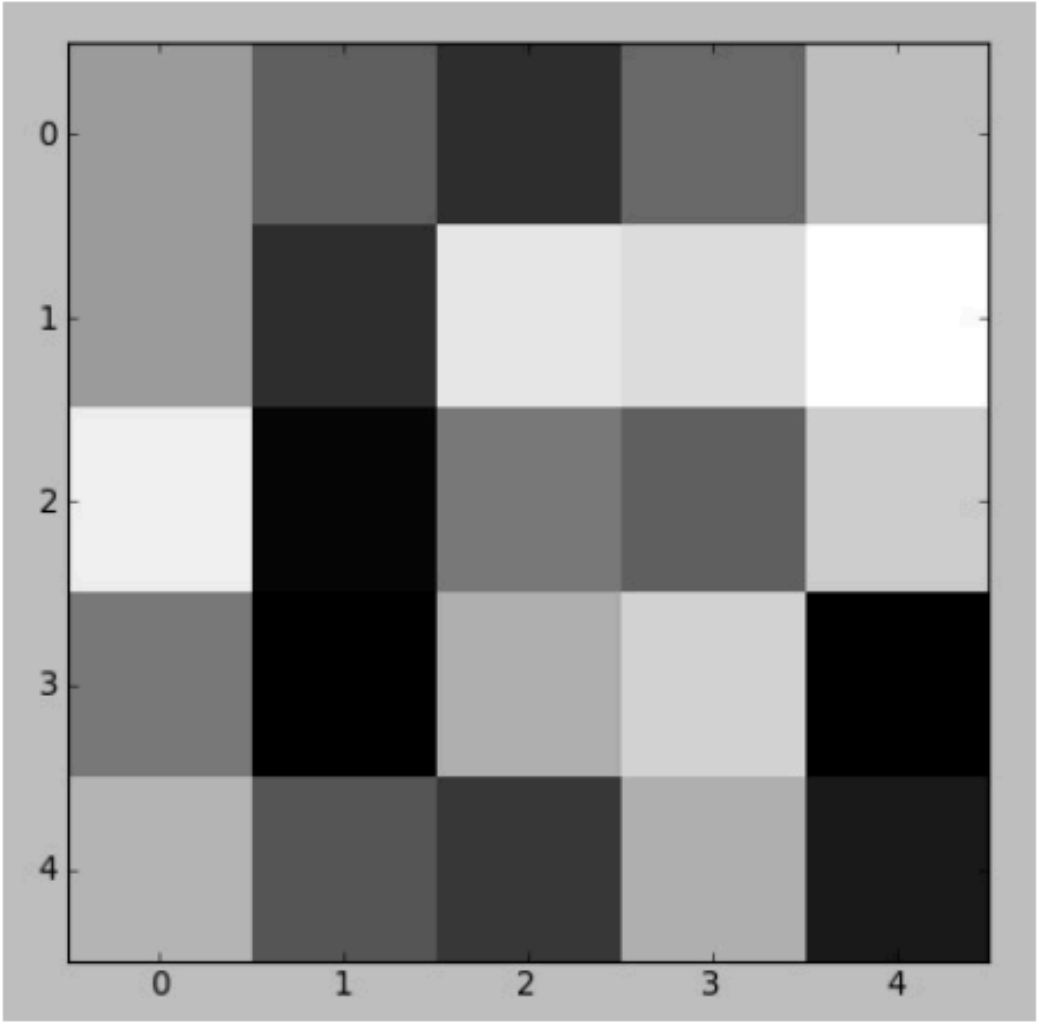
```
1  import numpy as np
2
3
4  a=np.array([1,2,3])
5
6  print a.dtype
7
8  b=np.array([1,2,3,4.56788])
9  print b.dtype
10
11 a=np.array([1,2,3], dtype=np.float32)
12 print a.dtype
13
14 print a
15
```

Making a Matrix

```
1 import numpy as np
2
3 x = np.arange(4)
4 print x
5
6 xnuevo=x.reshape( (2,2) )
7 print xnuevo
8
9 print xnuevo.T
```

+ HeatMaps (Mapas de Calor)

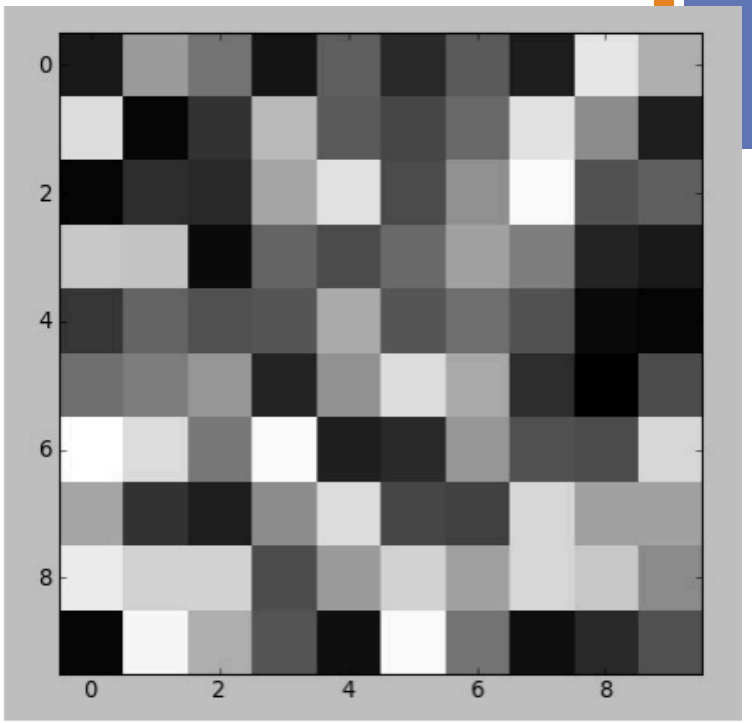
Heatmaps



A Heatmap is just a graphical way to render or visualize a matrix

A Simple Heatmap

imshow(A) - shows matrix A. Values in A must be between 0 and 1




```
import numpy as np
import matplotlib.pyplot as plt

A = np.random.random( (10,10) )
plt.gray()
plt.imshow(A,interpolation='nearest')
plt.show()
```

Setting Values in a Matrix

```
print A[i,j]
A[i,j] = 0.3
print A[i,j]
```

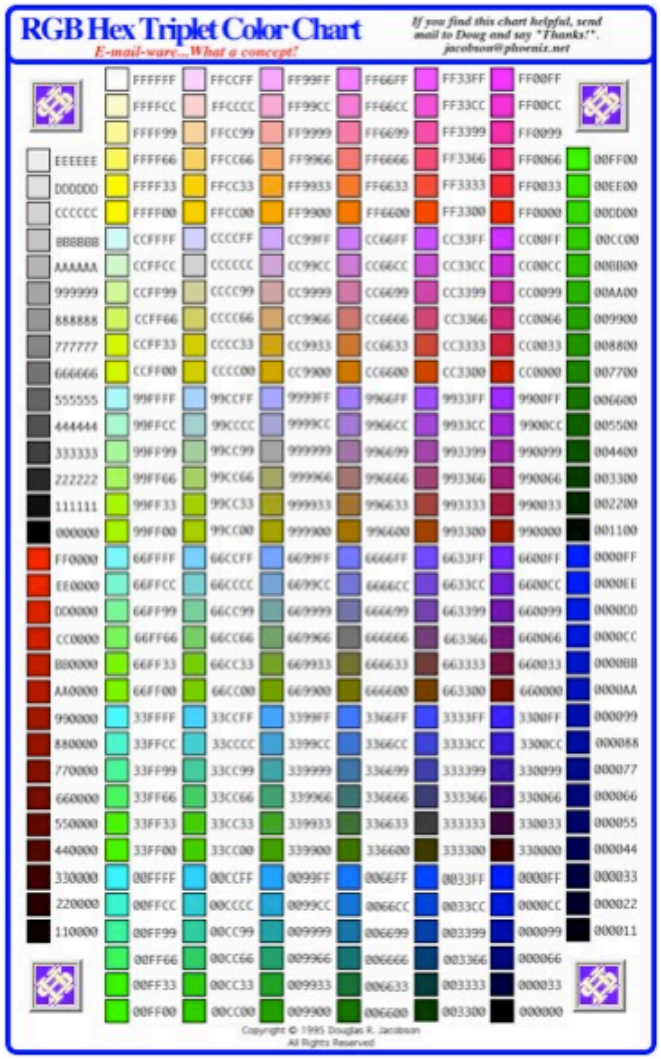
```
# -*- coding: utf-8 -*-
import  numpy   as  np
import  matplotlib.pyplot as plt

A=np.zeros((5,5))

print A

for i in range(0,5):
    print i
    A[i,i]=i*30

plt.gray()
plt.imshow(A,interpolation='nearest')
plt.show()
```

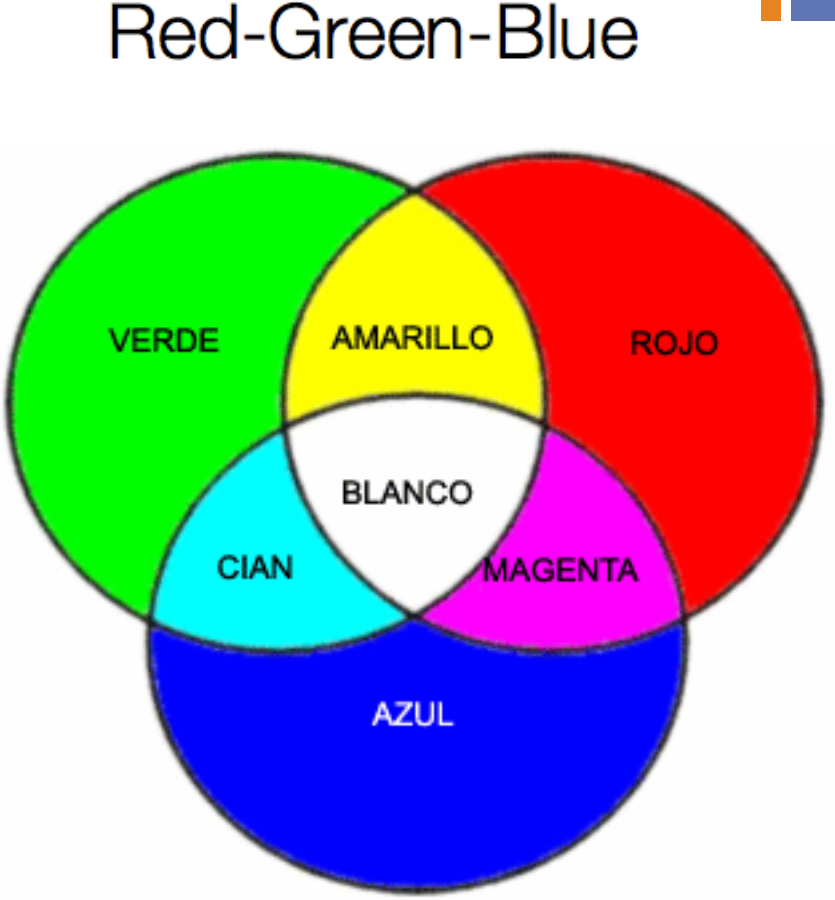
An Example

imshow(X) - if X is MxMx3, shows color (X[i,j,0],X[i,j,1],X[i,j,2]) at position X[i,j]

```
import matplotlib.pyplot as plt
import numpy as np
import sys

# Red gradient
A = np.random.random( (100,100,3) )
# i is the rows, j is the columns
for i in range(100):
  for j in range(100):
    A[i,j,0] = 1-(i)*0.01 #Red
    A[i,j,1] = 0          #Green
    A[i,j,2] = 0          #Blue
print A
plt.gray()
plt.imshow(A,interpolation='nearest')
plt.show()
```

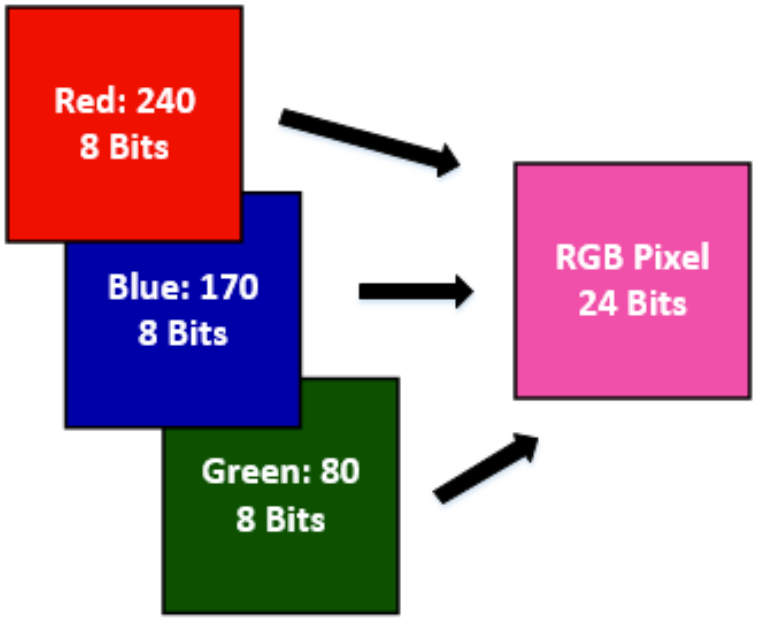
Representacion de Colores en Imagenes



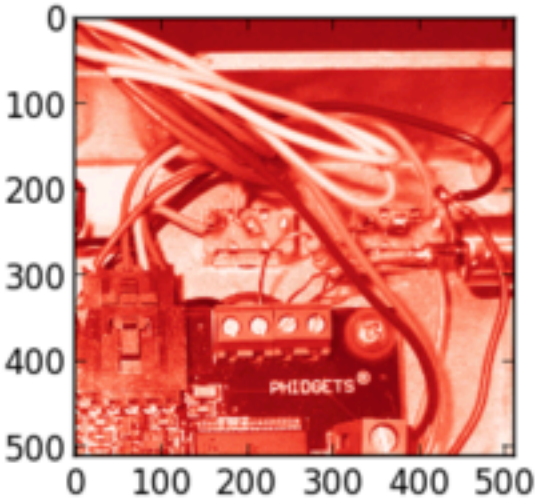


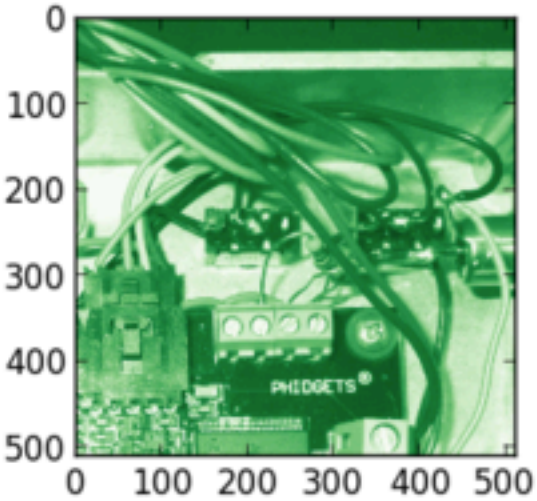
Representacion de Colores en Imagenes

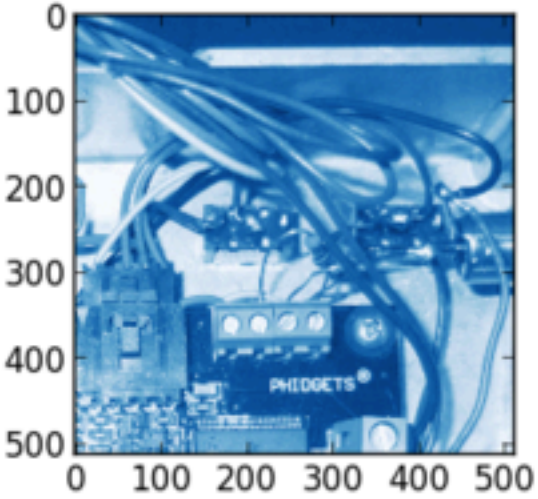
An ***RGB*** image is essentially three images layered on top of one another; a red scale image, a green scale image, and a blue scale image, with each pixel in them being 8 bits (intensity value ranging 0 - 255). To store a single pixel of an RGB image, you need to store 8 bits for all three colors, so a total of 24 bits per pixel.

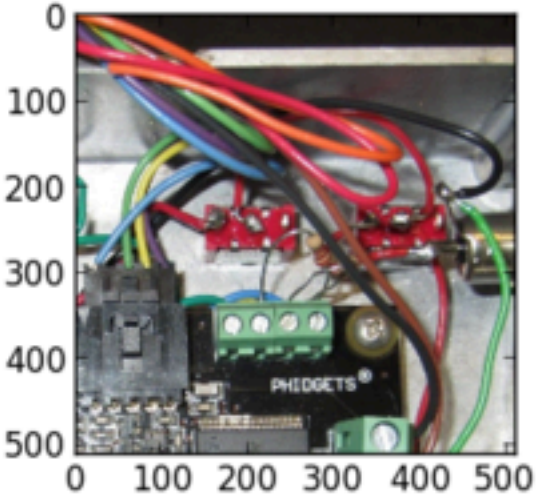


Color Representation in Images





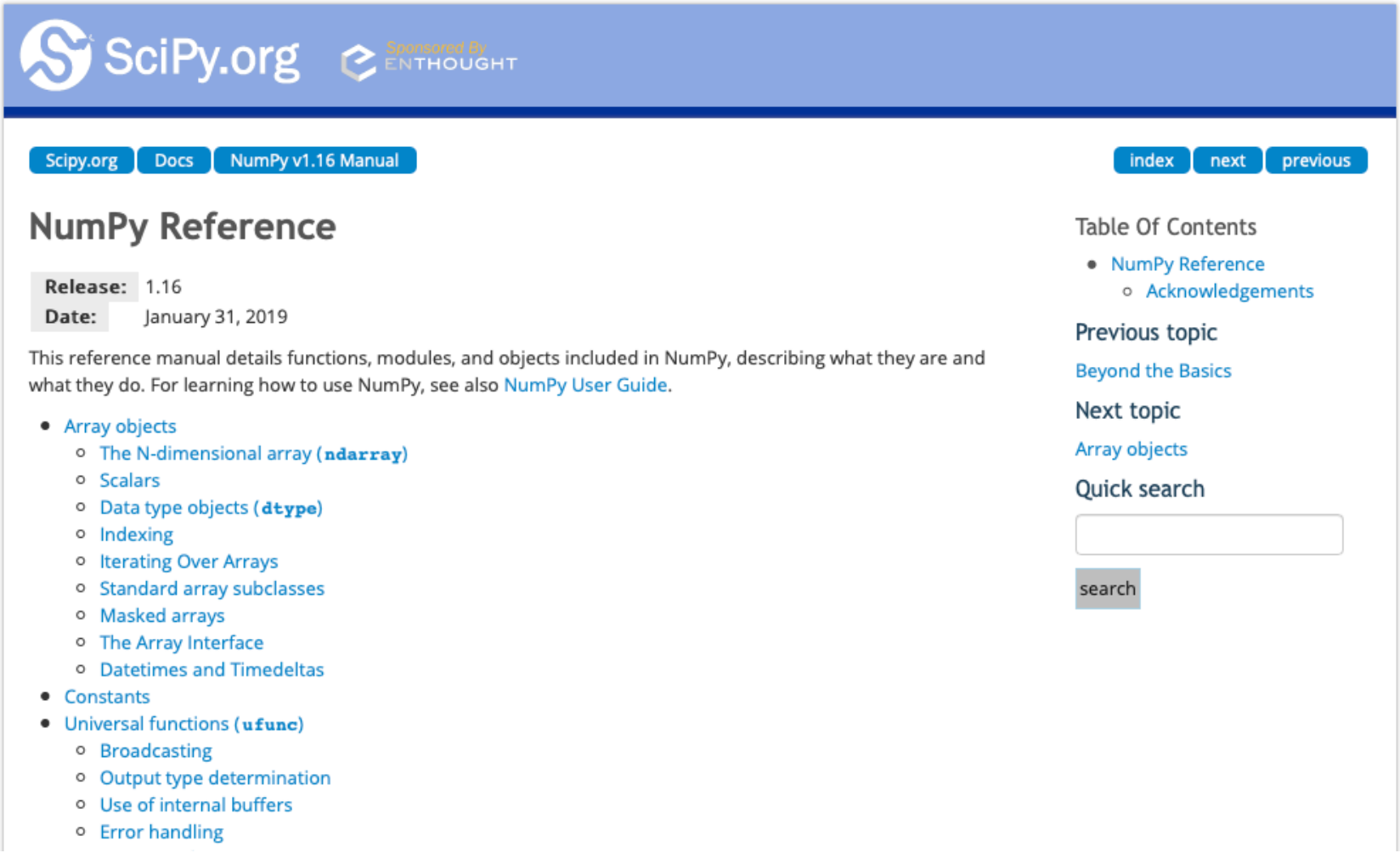




Color Representation in Images

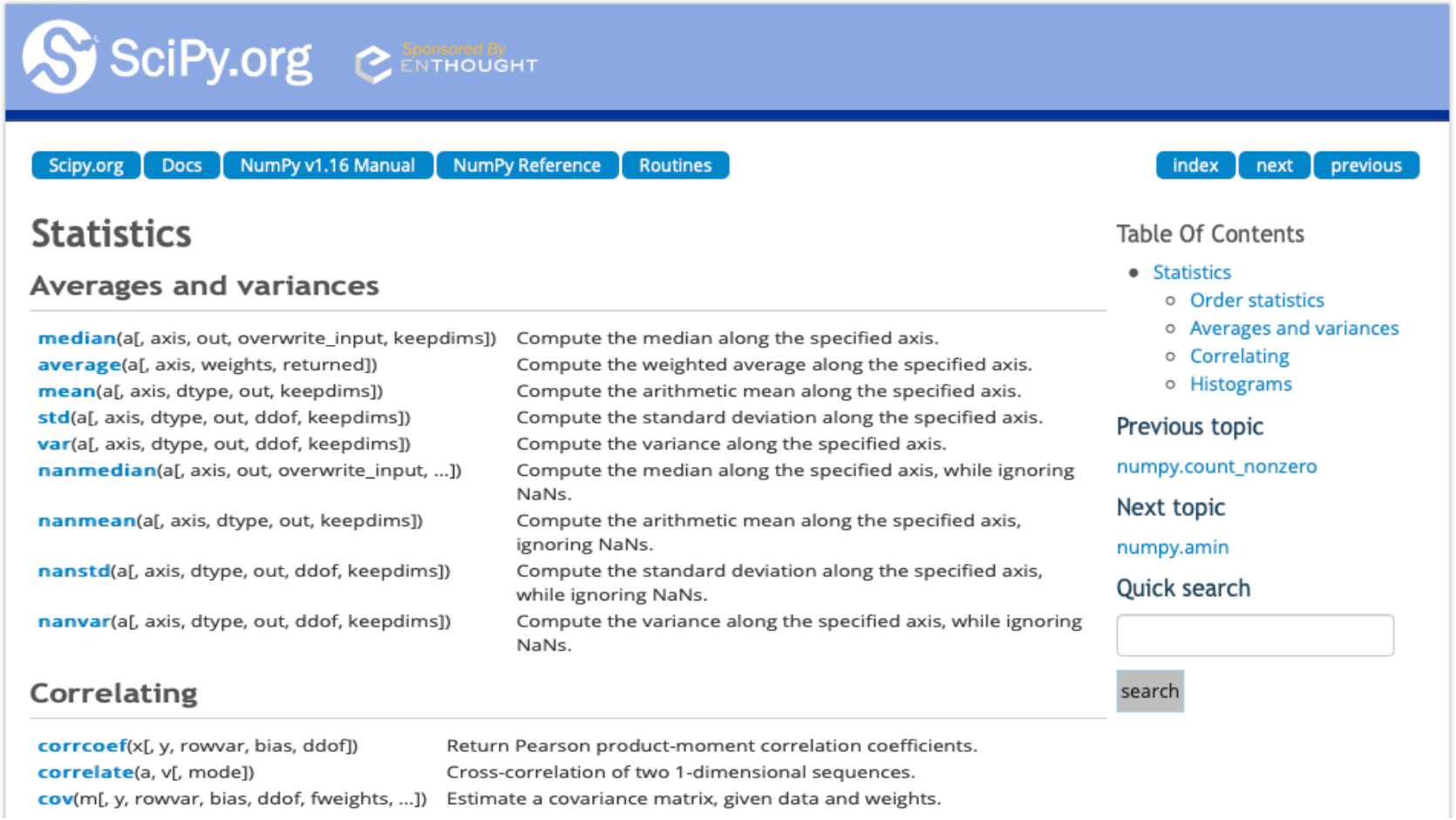
+ Otras Funciones

Referencias



https://docs.scipy.org/doc/numpy/reference/

Referencias



SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | NumPy v1.16 Manual | NumPy Reference | Routines | index | next | previous

Statistics

Averages and variances

median(a[, axis, out, overwrite_input, keepdims])	Compute the median along the specified axis.
average(a[, axis, weights, returned])	Compute the weighted average along the specified axis.
mean(a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis.
std(a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis.
var(a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis.
nanmedian(a[, axis, out, overwrite_input, ...])	Compute the median along the specified axis, while ignoring NaNs.
nanmean(a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis, ignoring NaNs.
nanstd(a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis, while ignoring NaNs.
nanvar(a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis, while ignoring NaNs.

Correlating

corrcoef(x[, y, rowvar, bias, ddof])	Return Pearson product-moment correlation coefficients.
correlate(a, v[, mode])	Cross-correlation of two 1-dimensional sequences.
cov(m[, y, rowvar, bias, ddof, fweights, ...])	Estimate a covariance matrix, given data and weights.

Table Of Contents

- Statistics
  - Order statistics
  - Averages and variances
  - Correlating
  - Histograms

Previous topic

numpy.count_nonzero

Next topic

numpy.amin

Quick search

search

https://docs.scipy.org/doc/numpy/reference/routines.statistics.html

Standard Deviation

Measure the variability of your dataset with respect to the mean. ***numpy*** can help you calculate this:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Also available in numpy: var, average, median, mean

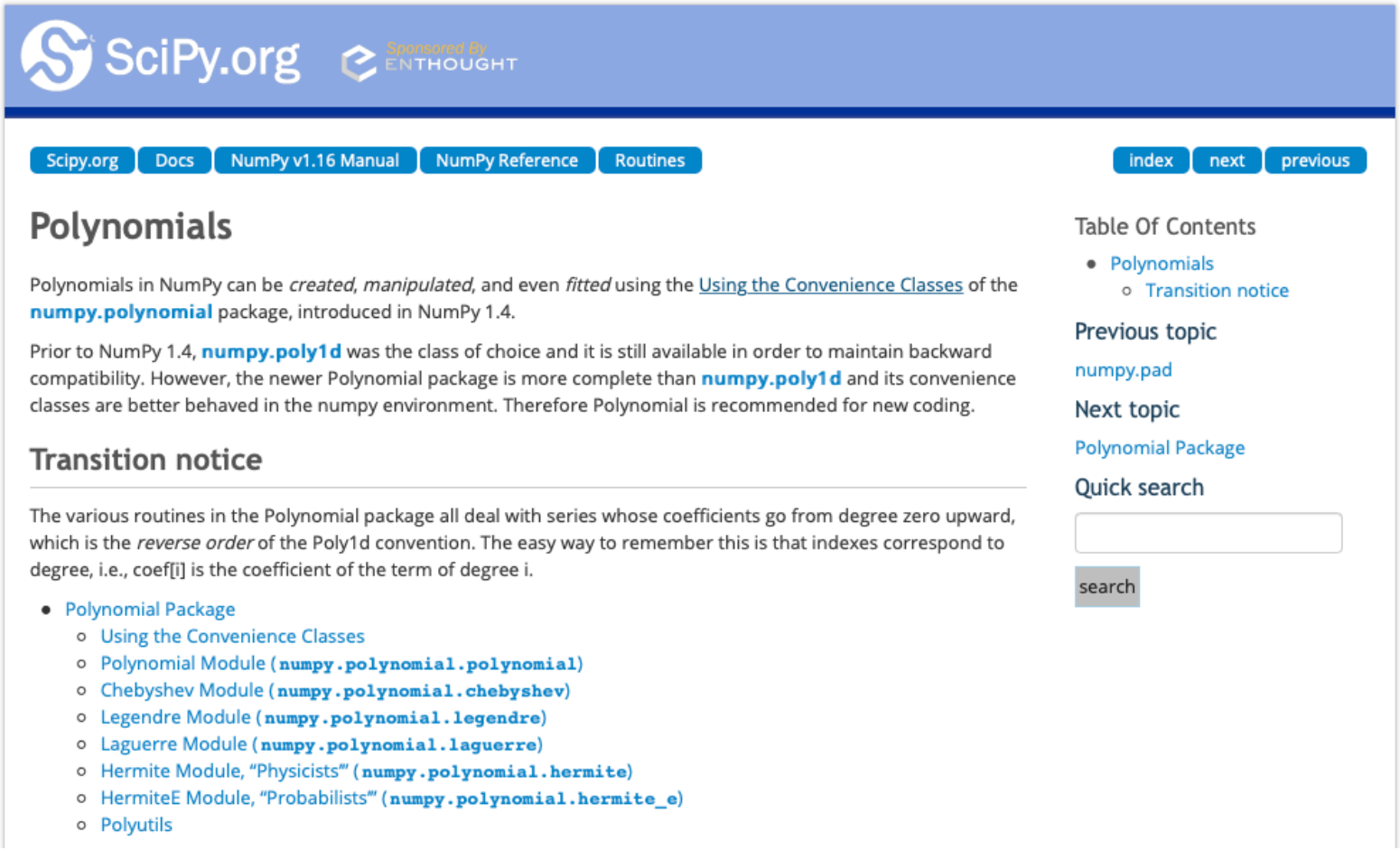
```
import numpy as np

stddev = np.std(y)
```

Funciones Estadisticas

Histograms

histogram(a[, bins, range, normed, weights, ...])	Compute the histogram of a set of data.
histogram2d(x, y[, bins, range, normed, ...])	Compute the bi-dimensional histogram of two data samples.
histogramdd(sample[, bins, range, normed, ...])	Compute the multidimensional histogram of some data.
bincount(x[, weights, minlength])	Count number of occurrences of each value in array of non-negative ints.
histogram_bin_edges(a[, bins, range, weights])	Function to calculate only the edges of the bins used by the **histogram** function.
digitize(x, bins[, right])	Return the indices of the bins to which each value in input array belongs.

Polynomiales



SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | NumPy v1.16 Manual | NumPy Reference | Routines | index | next | previous

Polynomials

Polynomials in NumPy can be *created*, *manipulated*, and even *fitted* using the Using the Convenience Classes of the **numpy.polynomial** package, introduced in NumPy 1.4.

Prior to NumPy 1.4, **numpy.poly1d** was the class of choice and it is still available in order to maintain backward compatibility. However, the newer Polynomial package is more complete than **numpy.poly1d** and its convenience classes are better behaved in the numpy environment. Therefore Polynomial is recommended for new coding.

Transition notice

The various routines in the Polynomial package all deal with series whose coefficients go from degree zero upward, which is the *reverse order* of the Poly1d convention. The easy way to remember this is that indexes correspond to degree, i.e., coef[i] is the coefficient of the term of degree i.

- Polynomial Package
  - Using the Convenience Classes
  - Polynomial Module (**numpy.polynomial.polynomial**)
  - Chebyshev Module (**numpy.polynomial.chebyshev**)
  - Legendre Module (**numpy.polynomial.legendre**)
  - Laguerre Module (**numpy.polynomial.laguerre**)
  - Hermite Module, "Physicists"' (**numpy.polynomial.hermite**)
  - HermiteE Module, "Probabilists"' (**numpy.polynomial.hermite_e**)
  - Polyutils

Table Of Contents

- Polynomials
  - Transition notice

Previous topic

numpy.pad

Next topic

Polynomial Package

Quick search

search

https://docs.scipy.org/doc/numpy/reference/routines.polynomials.html

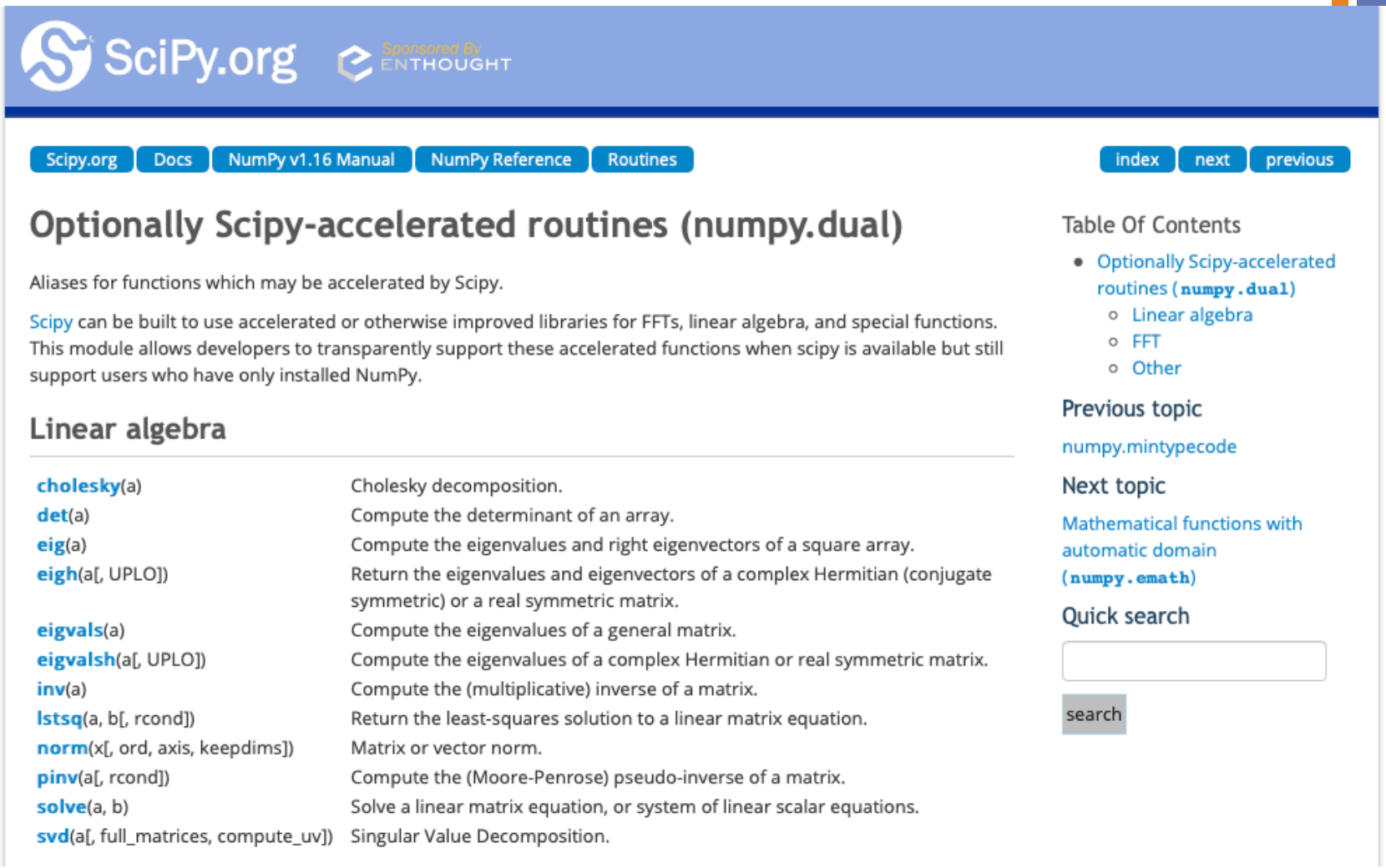
Line of best fit

Polynomial curve that best fits the dataset. Depending on the experiment being performed, different degree polynomials apply.

Be careful of over fitting!

```
import numpy as np
import matplotlib.pyplot as plt

# Linear regression
params = np.polyfit(x, y, 1) # Pick your degree
yp = np.polyval(params, x)

plt.plot(x,y,'g.') # Plot your real data
plt.plot(x,yp)     # Plot the best fit
```

Calculo Numerico



SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | NumPy v1.16 Manual | NumPy Reference | Routines

index | next | previous

Optionally Scipy-accelerated routines (numpy.dual)

Aliases for functions which may be accelerated by Scipy.

Scipy can be built to use accelerated or otherwise improved libraries for FFTs, linear algebra, and special functions. This module allows developers to transparently support these accelerated functions when scipy is available but still support users who have only installed NumPy.

Linear algebra

cholesky(a)	Cholesky decomposition.
det(a)	Compute the determinant of an array.
eig(a)	Compute the eigenvalues and right eigenvectors of a square array.
eigh(a[, UPLO])	Return the eigenvalues and eigenvectors of a complex Hermitian (conjugate symmetric) or a real symmetric matrix.
eigvals(a)	Compute the eigenvalues of a general matrix.
eigvalsh(a[, UPLO])	Compute the eigenvalues of a complex Hermitian or real symmetric matrix.
inv(a)	Compute the (multiplicative) inverse of a matrix.
lstsq(a, b[, rcond])	Return the least-squares solution to a linear matrix equation.
norm(x[, ord, axis, keepdims])	Matrix or vector norm.
pinv(a[, rcond])	Compute the (Moore-Penrose) pseudo-inverse of a matrix.
solve(a, b)	Solve a linear matrix equation, or system of linear scalar equations.
svd(a[, full_matrices, compute_uv])	Singular Value Decomposition.

Table Of Contents

- Optionally Scipy-accelerated routines (**numpy.dual**)
  - Linear algebra
  - FFT
  - Other

Previous topic

numpy.mintypecode

Next topic

Mathematical functions with automatic domain (**numpy.emath**)

Quick search

search

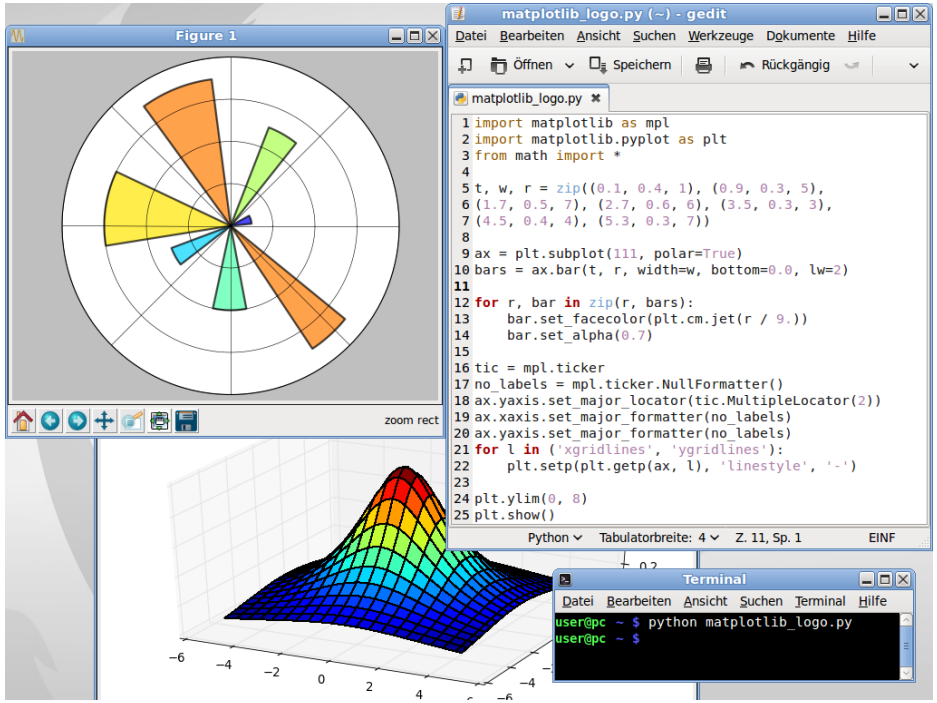
https://docs.scipy.org/doc/numpy/reference/routines.dual.html

+ Matplotlib

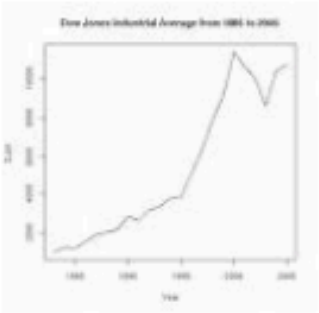
matplotlib

matplotlib is a plotting library for the Python programming language and its numerical mathematics extension ***NumPy***. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like wxPython, Qt, or GTK+.

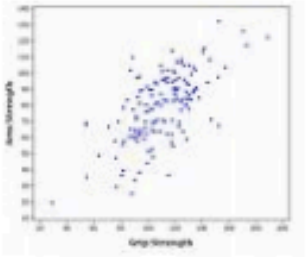
There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB. SciPy makes use of matplotlib.



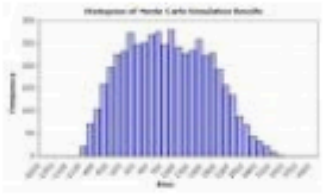
Different Kinds of Plots



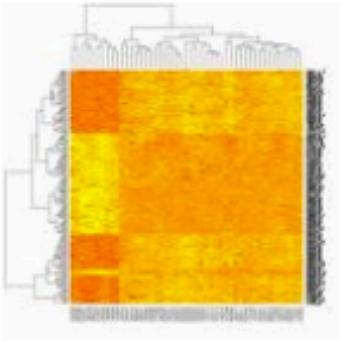
Line plot



Scatter plot



Histogram



Heatmap

Line Plots

- matplotlib is a 3rd party python library that provides MANY plotting functions (http://matplotlib.sourceforge.net)
- *matplotlib.pyplot.figure*() - creates a new blank figure
- *matplotlib.pyplot.plot*(X,Y) - draws a line plot using data points X,Y on the current figure
- *matplotlib.pyplot.show*() - displays the current figure on the screen

Stylizing our plot

matplotlib.pyplot.plot(X,Y,fmt) - fmt is a string that tells matplotlib how our points should be drawn and connected.

- *plot(X,Y,'r')* - draw in red
- *plot(X,Y,'b')* - draw in blue
- *plot(X,Y,'--b')* - draw a dashed blue line
- *plot(X,Y,'g.')* - draw a scatterplot with green points

matplotlib.pyplot.hold(True) - tells matplotlib to combine future plots onto thecurrent plot (rather than replacing it)

Annotating a plot

matplotlib.pyplot.title(s) - set the title of the current plot to s

matplotlib.pyplot.xlabel(s) - set the label of the x axis to s

matplotlib.pyplot.ylabel(s) - set the label of the y axis to s

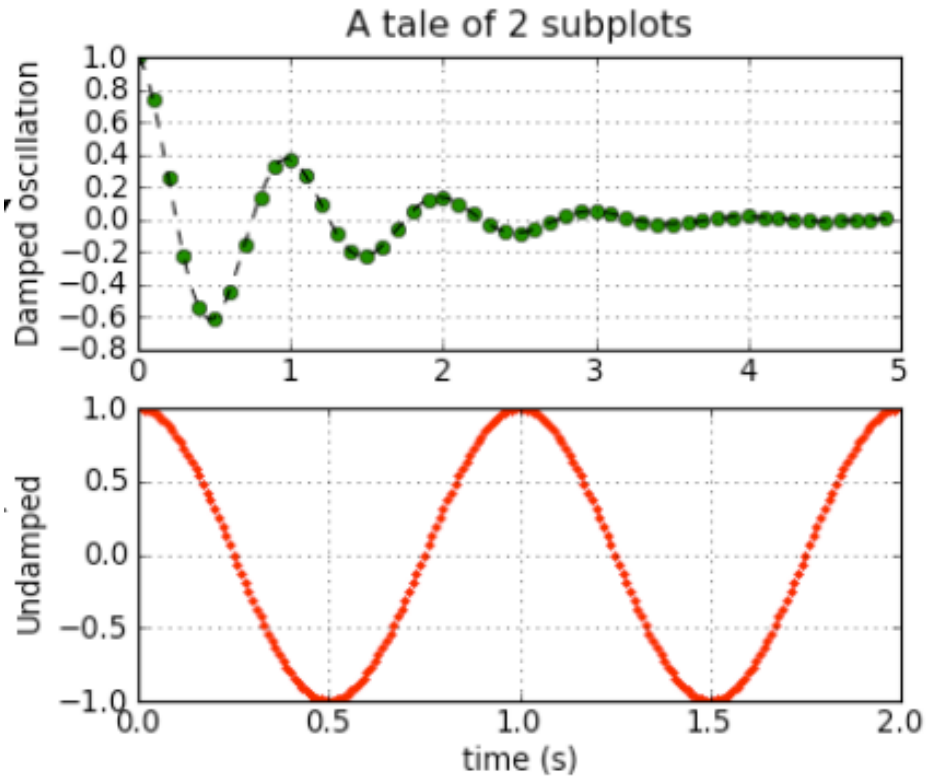
matplotlib.pyplot.legend([c1,c2,...]) - draw a legend on the figure labeling each curve

Subplots

matplotlib.pyplot.subplot(# rows, # cols, plot #)

matplotlib.pyplot.subplot(2,1,1)

matplotlib.pyplot.subplot(2,1,2))



A Simple Example

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[2,4,6,8,10]

plt.figure()
plt.plot(x,y,'g--')
plt.xlabel("Valores en X")
plt.ylabel("Valores en Y")
plt.title("Mi primera grafica en Matplotlib")
plt.show()
```

A Simple Example

```
import matplotlib.pyplot as plt
import sys, math


x=[1,2,3,4,5]
y=[2,4,6,8,10]

plt.figure()
plt.plot(x,y, 'g.')
plt.show()
```

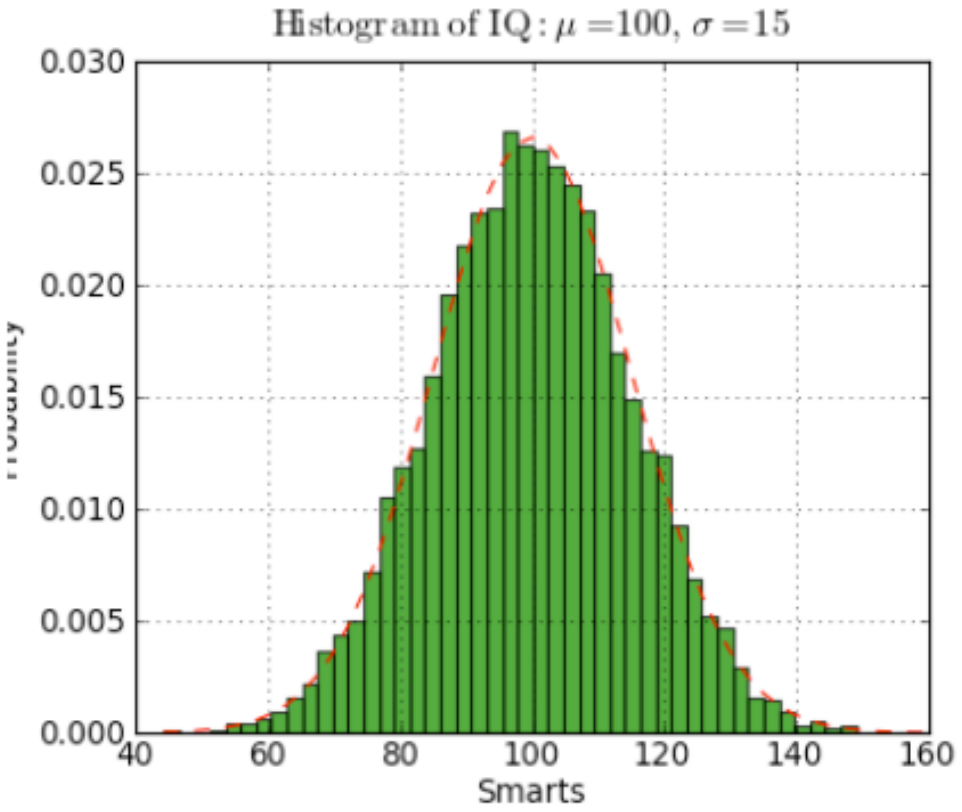
```
import matplotlib.pyplot as plt
import sys, math


x=[1,2,3,4,5]
y=[2,4,6,8,10]
xb=[sin(r) for r in x ]


plt.figure()
plt.plot(x,y,'b.')
plt.hold(True)
plt.plot(x,xb,'g.')
plt.title('Sin(x) function')
plt.xlabel('x values')
plt.ylabel('y values')
plt.show()
```

```
import matplotlib.pyplot as plt
import math

A = 3.0
OMEGA = 2*math.pi/0.75

x1 = [x*0.02 for x in range(1,100)]
sinc = [math.sin(OMEGA*x)/(OMEGA*x) for x in x1]

x2 = [x*0.01 for x in range(0,200)]
normalsin = [A*math.sin(OMEGA*x) for x in x2]


plt.figure()

plt.subplot(2,1,1)
plt.ylabel("Damped oscillation")
plt.plot(x1, sinc, 'r.')
plt.plot(x1, sinc, 'k--')
plt.ylim((-1.0,1.0))
plt.grid(color='k', linestyle=':', linewidth=1)
plt.title("The world of oscillations")
plt.subplot(2,1,2)
plt.grid(color='k', linestyle=':', linewidth=1)
plt.ylabel("Undamped")
plt.xlabel("Time (s)")
plt.plot(x2, normalsin, 'g')
plt.show()
```

Histograms

hist(...)

hist(x,bins=10)



All available options.

http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.hist

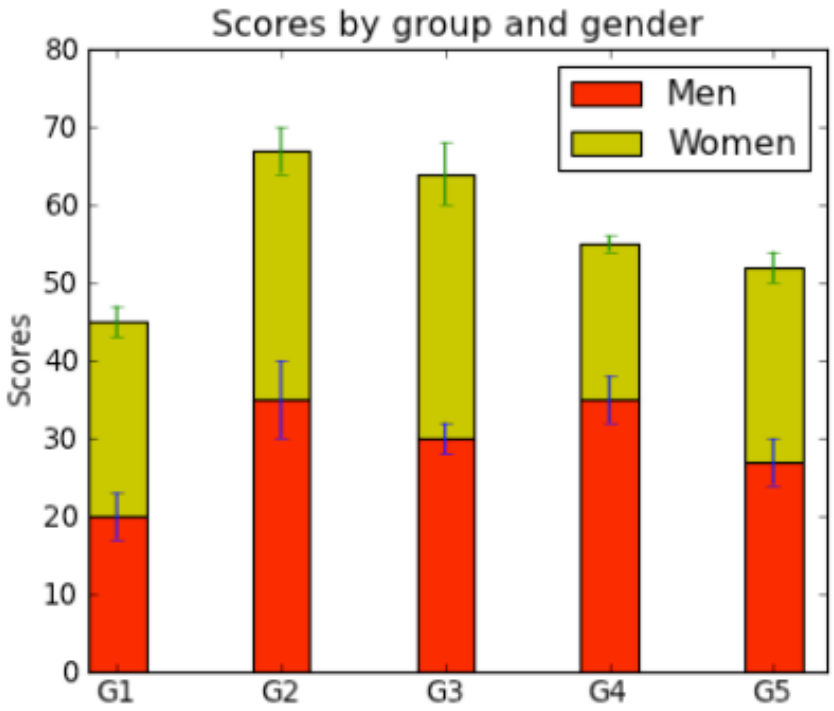
An Example

```
import matplotlib.pyplot as plt
import random, sys

# Simple demo first. Data is generated randomly but could be something else.
data = [random.random()*50. for i in range(50)]
print data
plt.figure()
plt.title('Distribution of randomly generated numbers')
plt.ylabel('Number of occurrences')
plt.xlabel('Bins')
plt.hist(data, bins=10,color='g')
plt.show()
```

Barcharts

bar(left,height)

Argument Description
left the x coordinates of the left sides of the bars
height the heights of the bars
Bar charts are useful to print frequencies or scores. More generally, data that is categorized.



All available options:
http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.bar

An Example

```
import matplotlib.pyplot as plt
import sys

# Bar chart demo

plt.figure()
plt.bar([0, 1, 2, 3], [80,56,44,32], width=0.5)
plt.ylabel('Number of occurrences')
plt.xlabel('Examples')
plt.show
```

```python
import numpy as np
import matplotlib.pyplot as plt

N = 5
menMeans = (20, 35, 30, 35, 27)
menStd =   (2, 3, 4, 1, 2)

ind = np.arange(N)  # the x locations for the groups
width = 0.35       # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(ind, menMeans, width, color='r', yerr=menStd)

womenMeans = (25, 32, 34, 20, 25)
womenStd =   (3, 5, 2, 3, 3)
rects2 = ax.bar(ind+width, womenMeans, width, color='y', yerr=womenStd)

# add some text for labels, title and axes ticks
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(ind+width)
ax.set_xticklabels( ('G1', 'G2', 'G3', 'G4', 'G5') )

ax.legend( (rects1[0], rects2[0]), ('Men', 'Women') )

def autolabel(rects):
    # attach some text labels
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.05*height, '%d'%int(height),
                ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)

plt.show()
```

Advanced Plotting

- Changing the limits of your axes

matplotlib allows you to change the limits of your plot

```
import matplotlib.pyplot as plt

plt.xlim(-1.0, 1.0)
plt.ylim(0, 50.0)

plt.xlim(xmax=1.0) # Also possible
```

http://matplotlib.sourceforge.net/apipyplot_api.html#matplotlib.pyplot.xlim

Advanced Plotting

Legend vs. Text: matplotlib allows you to put up a legend associated with a line

```
import matplotlib.pyplot as plt

plt.plot(x,y)
plt.legend(['sin(x)'])
```

But you can also put up a text label, if appropriate. Choose location based on the scale of your plot, and available whitespace (don't put over data!)

```
plt.plot(x,y)
plt.text(1.0, 5.0, 'Correlation: 0.3')
# Example at x=1.0, y=5.0
```

Advanced Plotting

- Horizontal and vertical lines: matplotlib allows you to add vertical and horizontal lines, pretty easily.

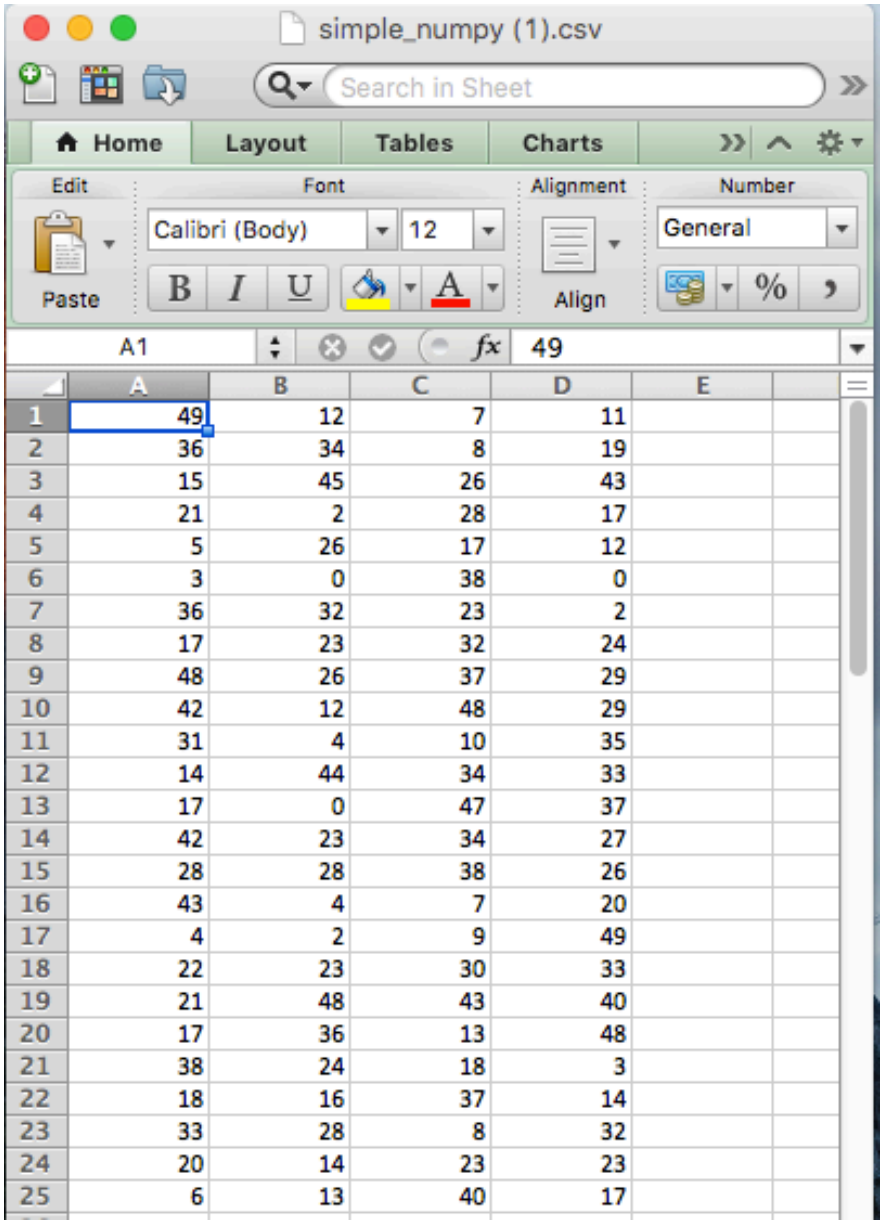
```
matplotlib.pyplot.axhline(y=0, xmin=0, xmax=1, hold=None, **kwargs)
matplotlib.pyplot.axvline(x=0, ymin=0, ymax=1, hold=None, **kwargs)
```

- The min and max bounds are fractions of the axis. For example, setting ymax=0.5 on a vertical line will make it go up the the middle of the screen. matplotlib allows you to change the limits of your plot

http://matplotlib.sourceforge.net/apipyplot_api.html#matplotlib.pyplot.xlim

+ Numpy Avanzado

File Processing with NumPy



	A	B	C	D	E
1	49	12	7	11	
2	36	34	8	19	
3	15	45	26	43	
4	21	2	28	17	
5	5	26	17	12	
6	3	0	38	0	
7	36	32	23	2	
8	17	23	32	24	
9	48	26	37	29	
10	42	12	48	29	
11	31	4	10	35	
12	14	44	34	33	
13	17	0	47	37	
14	42	23	34	27	
15	28	28	38	26	
16	43	4	7	20	
17	4	2	9	49	
18	22	23	30	33	
19	21	48	43	40	
20	17	36	13	48	
21	38	24	18	3	
22	18	16	37	14	
23	33	28	8	32	
24	20	14	23	23	
25	6	13	40	17	

File Processing with NumPy

```
import numpy as np

"""
Demonstrate some array calculations using NumPy.
"""

## Read a csv file as a matrix
X = np.loadtxt("simple_numpy.csv", delimiter=",")

## Print the dimensions of the array X
print "The dimensions of X are:"
print X.shape

## Print the number of values in X that are greater than 10
print "\nThe number of entries of X that exceed 10:"
print (X > 10).sum()

## Print the proportion of values in X that are greater than 10
print "\nThe proportion of all entries of X that exceed 10:"
print (X > 10).mean()

## Print the proportion of values in each column of X that are
## greater than 10
print "\nThe proportion of entries in each column of X that exceed 10:"
print (X > 10).mean(0)
```

Tópicos Especiales I

Clase 8: Numerical Python, Matplotlib, SciPy II

Javier Sanchez Galan, PhD
Facultad de Ingeniería de Sistemas Computacionales
10/octubre/2019