Data Analysis and Machine Learning Using Python

Course Contents

- **Python Refresher (4 hours)**
  - Introduction
  - IDEs and Tools for data analysis
  - Data types and strings
  - Control structures
  - Functions and functional programming
  - Collections
  - Object Oriented
  - Modules and Packages
  - Lab 1– Python Refresher

- **Introduction to Data Analysis (1 hour)**
  - Introduction
  - Data Analysis and ML Packages overview

- **Introduction to NumPy (6 hours)**
  - Python arrays and NumPy Arrays
  - Multi-dimensional Arrays
  - Array slicing
  - Fancy Indexing
  - Data types
  - Array calculation methods
  - Statistics methods
  - Universal functions
  - Broadcasting
  - Universal function methods
  - Lab 2– Using Numpy

- **Matplotlib (4 hours)**
  - Line plots
  - Scalar plots
  - Bar plots
  - Histograms
  - Multiple plots
  - Lab 3– Data visualization with Matplotlib

- **SciPy (2 hours)**
  - Overview
  - Interpolation
  - Integration
  - FFT
  - Signal and image processing
  - Optimizations
  - Statistics
  - Linear algebra
  - Matrix objects
  - Interactive Plotting
  - Scikits
  - Lab 4– image processing

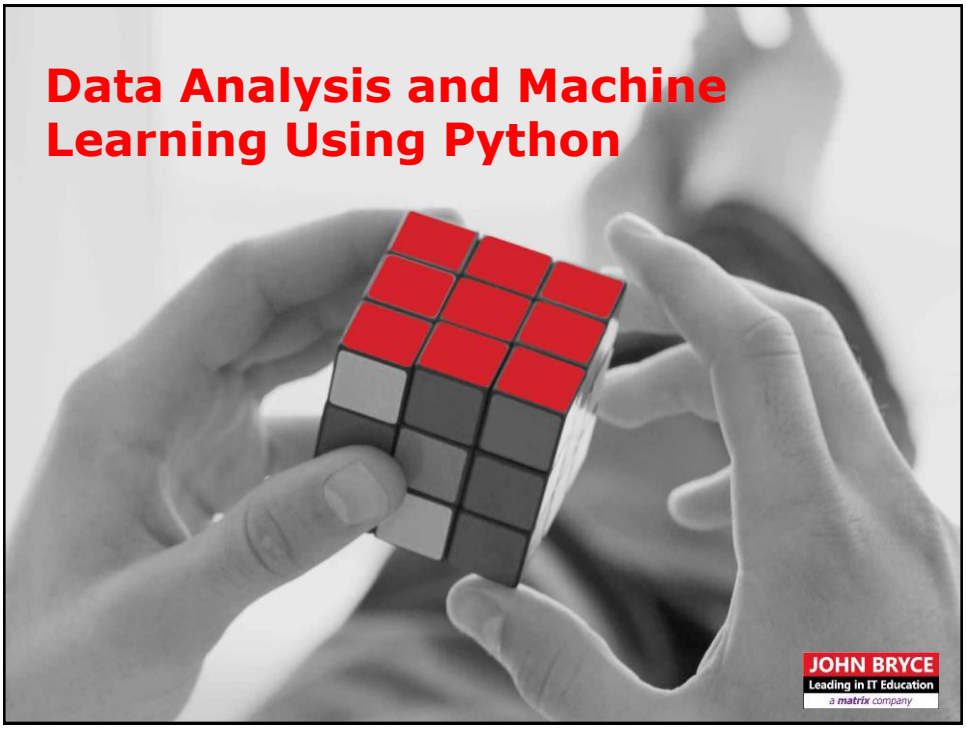
- **Pandas (4 hours)**
  - Overview
  - Data analysis with pandas
  - Pandas basics
  - Data structures – series, dataframes
  - Visualization
  - Working with data
  - Grouping, serialization and more
  - Lab 5– manipulating log files

- **Seaborn (3 hours)**
  - Overview
  - Statistics graphs
  - Grids
  - Lab 6– visualization with seaborm

- **Machine Learning (16 hours)**
  - Overview
  - Why Learn
  - Applications
  - Machine Learning Process
  - Learning Types:
    - Supervised Learning
    - Unsupervised Learning
    - Semi-Supervised Learning
    - Active Learning
    - Reinforcement learning
  - Batch vs Online Learning
  - Instance vs model based learing
  - Variables and features
  - Classification

- Regressions
- Types of Data
- Data preparation and cleaning
- Data visualization
- Missing values
- Scikit Learn
- Linear Regression
- Lab 7– using linear regression
- Logistic Regression
- Lab 8– using Logistic regression
- Support vector machines
- Decision trees and random forests
- Naïve Bayes
- KNN
- NLP
- Lab 9– Using NLP
- Deep Learning
- Neural networks
- Lab 10 – Neural network
- Other tools and libraries

Data Analysis and Machine Learning Using Python



Python and Data Analysis

JOHN BRYCE
Leading in IT Education
a matrix company

- Python is a great programming language
  - ✓ Open source
  - ✓ Cross platform
  - ✓ Easy to learn and use
  - ✓ General purpose
    - Procedural
    - Functional
    - Object Oriented
- With tons of available packages you can write almost any program with python
- Many packages to manage data (local and remote) and analyze data

Agenda

JOHN BRYCE
Leading in IT Education
a matrix company

- Python Refresher
- Numpy
- Matplotlib
- Scipy and Scikits
- Pandas
- Seaborn
- Machine Learning and Scikit-Learn

Prerequisites

JOHN BRYCE
Leading in IT Education
a matrix company

- Python programming experience
- Basic Math and Statistics

JOHN BRYCE
Leading in IT Education
a matrix company

Python Refresher

What is Python?

JOHN BRYCE
Leading in IT Education
a matrix company

- General purpose computer programming language
  - Has math and science add-ins
  - Has GUI add-ins
  - Many more
- Multi-platform
  - Same program can run on MS Windows, Mac OS X, Linux, etc.

What can I do with Python?

JOHN BRYCE
Leading in IT Education
a matrix company

- Administrative scripts
- Work with data
- Web development
- Mobile development
- Desktop applications
- Embedded and IOT
- Just about anything else you can think of

Why Python?

JOHN BRYCE
Leading in IT Education
a matrix company

- Python is
  - easy to learn
  - relatively fast
  - procedural
  - object-oriented
  - functional
  - strongly typed
  - widely used
  - portable
- C is much faster but much harder to use.
- Java is about as fast and slightly harder to use.
- Perl is slower, is as easy to use, but is not strongly typed.

How to get and install Python

JOHN BRYCE
Leading in IT Education
a matrix company

- We're going to install several components:
  - Install Python – the main language
  - Useful packages
  - IDE
- Versions
  - 2.x
    - Legacy
    - More Libraries
  - 3.x
    - The future

Scripts

JOHN BRYCE
Leading in IT Education
a matrix company

- A script is a file that contains a program in a high-level language for an interpreter
- Python scripts are text files ending in .py

Objects and types

JOHN BRYCE
Leading in IT Education
a matrix company

- We use the term object to refer to any entity in a python program.
- Every object has an associated type, which determines the properties of the object.
- Python defines six types of built-in objects:

Number	10
String	"hello"
List	[1, 17, 44]
Tuple	(4, 5)
Dictionary	{'food' : 'something you eat', 'lobster' : 'an edible, undersea arthropod'}
Files	

- Each type of object has its own properties
- It is also possible to define your own types, comprised of combinations of the six base types.

Data types

JOHN BRYCE
Leading in IT Education
a matrix company

- Python has multiple data types
  - Built-in
  - User defined
- Numeric types:
  - int, float, long, complex
- string
- Boolean

Data types

JOHN BRYCE
Leading in IT Education
a matrix company

- Use the type() function to find the type for a value or variable
- Data can be converted using *cast* commands
  - Ex.
```
a = str(17)
type(a)
```


Data Structures

JOHN BRYCE
Leading in IT Education
a matrix company

- Lists
- Tuples
- Dictionaries
- Sets
- Many more
  - Standard library
  - External packages

```
for x in [1,4,5,10]:
    print (x);


for x in (1,4,5,10):
    print (x);

prices = { 'GOOG' : 490.10,
           'AAPL' : 145.23,
           'YHOO' : 21.71 }
for key in prices:
    print (key);
```

Functions

JOHN BRYCE
Leading in IT Education
a matrix company

- Function
  - A named sequence of statements that performs a computation or action
- Functions are *called* by name
  - Most functions accept inputs (*arguments*)
  - Some functions return results (*return value*)

Example

JOHN BRYCE
Leading in IT Education
a matrix company

```
def power(base, exponent):
    val = base
    for x in range(1,exponent):
        val = val * base
    print val

power(3,4)
```

Modules

JOHN BRYCE
Leading in IT Education
a matrix company

- Module
  - A file that contains a collection of related functions
- Python has hundreds of standard modules
  - These are known as the Python Standard Library (http://docs.python.org/library/)
- You can also create and use add-in modules

Procedural programming

JOHN BRYCE
Leading in IT Education
a matrix company

```
# create empty list
target = []
# iterate over each thing in source
for item in source_list:
    trans1 = G(item)
    trans2 = F(trans1)
    # add transformed item to target
    target.append(trans2)
```

Functional programming

JOHN BRYCE
Leading in IT Education
a matrix company

- Based on the lambda calculus
- Programs are based on the systematic evaluation of mathematical functions
- Programs describe what to do, not how to do it

Functional programming in Python

JOHN BRYCE
Leading in IT Education
a matrix company

```
foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
print (filter(lambda x: x % 3 == 0, foo))
```

OOP in Python

JOHN BRYCE
Leading in IT Education
a matrix company

- Python is inherently OOP
  - Everything in Python is an object
- All OOP features included
  - ADT
  - Inheritance
  - Polymorphism
  - Operator overloading

Example

JOHN BRYCE
Leading in IT Education
a matrix company

```
class Rectangle:
    def __init__(self,width,height):
        self.width = width
        self.height = height
    def Print(self):
        print("Recangle Width:" + str(self.width) +
                    " Height:" + str(self.height))


a = Rectangle(10,20)
a.Print()
```

Example

JOHN BRYCE
Leading in IT Education
a matrix company

```
class Rectangle:
    def __init__(self,width,height):
        self.__width = width
        self.__height = height
    def __PrintHeader(self):
        print("****************************************")
    def Print(self):
        self.__PrintHeader()
        print("Recangle Width:" + str(self.__width) +
                    " Height:" + str(self.__height))


a = Rectangle(10,20)
a.Print()


a.__PrintHeader() # error
a.__height = -80  # new attribute
a.Print()
```


JOHN BRYCE
Leading in IT Education
a matrix company

```
class Rectangle:
    def __init__(self,width,height):
        self.__width = width
        self.__height = height
    def SetSize(self,new_width,new_height):
        if new_width > 0:
            self.__width = new_width
        if new_height > 0:
            self.__height = new_height
    def Print(self):
        print("Recangle Width:" + str(self.__width) +
                    " Height:" + str(self.__height))

class RoundedRectangle(Rectangle):
    def __init__(self,width,height,angle):
        Rectangle.__init__(self,width,height)
        self.__angle = angle
    def SetAngle(self,angle):
        self.__angle = angle
    def Print(self):
        Rectangle.Print(self)
        print("RecangleX angle:" + str(self.__angle))

a = Rectangle(10,20)
b = RoundedRectangle(8,9,7)
c = Rectangle(40,50)

mylist = [a,b,c]

for x in enumerate(mylist):
    x[1].Print()
```

Composition Example

JOHN BRYCE
Leading in IT Education
a matrix company

```
class A(object):
    def __init__(self,a):
        self.__a = a
        print "Constructor A was called"
    def Print(self):
        print("val=" + str(self.__a))

class B(object):
    def __init__(self,b):
        self.__innerA = A(b+10)
        print "Constructor B was called"
    def fn(self):
        print("B fn")
        self.__innerA.Print()

b = B(100)
b.fn()
```


Closures

JOHN BRYCE
Leading in IT Education
a matrix company

```
def mulby(num):
    def gn(val):
        return num*val
    return gn

zw=mulby(7)
print( zw(9));
```

```
def genfilterby(mod):
    def gn1(lst):
        x=[]
        for i in lst:
            if i % mod == 0:
                x+=[i]
        return x
    return gn1

fil=genfilterby(3)
z=[2,3,4,5,6,7,8] #hjhkjh


print( fil(z));
```

Python Packages

JOHN BRYCE
Leading in IT Education
a matrix company

- Package is a collection of modules
- Python has hundreds of modules covering almost all areas
  - GUI
  - Scientific
  - Big data
  - Gaming
  - Tools
  - ….

Threads

JOHN BRYCE
Leading in IT Education
a matrix company

```
import thread
import time


def print_time( threadName, delay):
   count = 0
   while count < 5:
      time.sleep(delay)
      count += 1
      print ("%s: %s" % ( threadName, time.ctime(time.time()) ))

# Create two threads as follows
try:
    thread.start_new_thread( print_time, ("Thread-1", 2, ) )
    thread.start_new_thread( print_time, ("Thread-2", 4, ) )
except:
    print( "Error: unable to start thread");

while 1:
    pass
```

Regular Expressions

JOHN BRYCE
Leading in IT Education
a matrix company

```
import re
x="[a-z]*@[a-z]*.[a-z]*"

s=raw_input("enter:")

c=re.match(x,s)
if c:
    print 'ok'
else:
    print 'no'
```


GUI

JOHN BRYCE
Leading in IT Education
a matrix company

- PyGUI
- PyGtk
- PyQT
- PyKDE
- …

Web Development

JOHN BRYCE
Leading in IT Education
a matrix company

- Python Web Frameworks
  - Tools, Libraries and everything you need to write, test, deploy and run a web app
- The world of Python web frameworks is full of choices:
  - Django
  - Flask
  - Pyramid
  - Tornado
  - Bottle
  - Diesel
  - Pecan
  - Falcon
  - …

Django

JOHN BRYCE
Leading in IT Education
a matrix company

- https://www.djangoproject.com
- The most popular
- Templating, forms, routing, authentication, basic database administration, and more

```
def a_view(request):

    return render_to_response(
        "view.html",
        {"user": cur_user}
    )
```

```
<!-- view.html -->
<div class="top-bar row">
  <div class="col-md-10">
  <!-- more top bar things go here -->
  </div>
  {% if user %}
  <div class="col-md-2 whoami">
    You are logged in as {{ user.fullname }}
  </div>
  {% endif %}
</div>
```

Flask

JOHN BRYCE
Leading in IT Education
a matrix company

- http://flask.pocoo.org
- Microframework

Flask is Fun

Latest Version: 0.11

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

And Easy to Setup

```
$ pip install Flask
$ python hello.py
 * Running on http://localhost:5000/
```


Pyramid

JOHN BRYCE
Leading in IT Education
a matrix company

- https://trypyramid.com

```
@view_config(renderer='templates/home.pt')
def my_view(request):
    # do stuff...
    return {'user': user}
```

```
<div class="top-bar row">
  <div class="col-md-10">
  <!-- more top bar things go here -->
  </div>
  <div tal:condition="user"
       tal:content="string:You are logged in as ${user.fullname}"
       class="col-md-2 whoami">
  </div>
</div>
```

Mobile Applications

JOHN BRYCE
Leading in IT Education
a matrix company

- https://kivy.org/#home
- https://www.blender.org
- http://pyzia.com
- …

Embedded and IOT

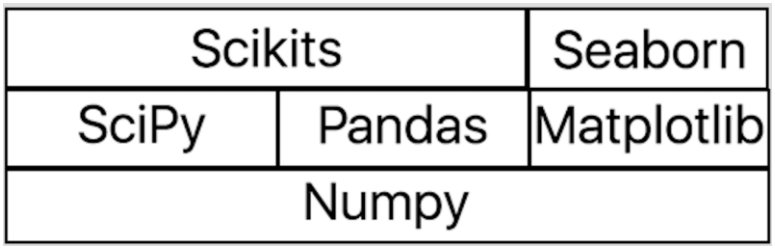
JOHN BRYCE
Leading in IT Education
a matrix company

- It is very easy to integrate python interpreter on OS based Embedded Systems
  - Embedded Linux
  - Android
- IOT
  - Zerynth
  - MicroPython
  - PyMCU

Data Analysis Packages

Overview

Packages



Numpy

JOHN BRYCE
Leading in IT Education
a matrix company

- Python is a fabulous language
  - Easy to extend
  - Great syntax which encourages easy to write and maintain code
  - Incredibly large standard-library and third-party tools
- No built-in multi-dimensional array (but it supports the needed syntax for extracting elements from one)
- NumPy provides a **fast** built-in object (ndarray) which is a multi-dimensional array of a homogeneous data-type.
- https://www.scipy.org

SciPy

JOHN BRYCE
Leading in IT Education
a matrix company

- Scientific Packages
  - Special Functions (scipy.special)
  - Signal Processing (scipy.signal)
  - Image Processing (scipy.ndimage)
  - Fourier Transforms (scipy.fftpack)
  - Optimization (scipy.optimize)
  - Numerical Integration (scipy.integrate)
  - Linear Algebra (scipy.linalg)
  - ….
- Available at www.scipy.org
- Open Source BSD Style License

Pandas

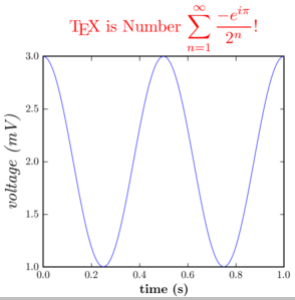
JOHN BRYCE
Leading in IT Education
a matrix company

- Python Library to provide data analysis features similar to : R, MATLB, SAS
- Rich data structures and functions to make working with data structure fast, easy and expressive.
- It is built on top of NumPy which provides it agility
- Key components provided by Pandas : Two new data structures to Python
  - Series
  - DataFrame

Matplotlib

JOHN BRYCE
Leading in IT Education
a matrix company

- Requires NumPy extension. Provides powerful plotting commands.
- http://matplotlib.sourceforge.net



Seaborn

- Statistical plotting library
- Built on Matplotlib
- Great styles
- Works great with NumPy arrays and Pandas Dataframes

Scikits

- add-on toolkits that complement Scipy
- The SciKits cover a broad spectrum of application domains, including:
  - Financial computation
  - Audio processing
  - Geosciences
  - Computer vision
  - Engineering
  - Machine learning
  - Medical computing
  - Bioinformatics
- https://scikits.appspot.com/scikits

JOHN BRYCE
Leading in IT Education
a matrix company

Numpy

Overview

JOHN BRYCE
Leading in IT Education
a matrix company

- Python is a fabulous language
  - Easy to extend
  - Great syntax which encourages easy to write and maintain code
  - Incredibly large standard-library and third-party tools
- No built-in multi-dimensional array (but it supports the needed syntax for extracting elements from one)
- NumPy provides a **fast** built-in object (ndarray) which is a multi-dimensional array of a homogeneous data-type.

N-D Array

JOHN BRYCE
Leading in IT Education
a matrix company

- N-dimensional array of rectangular data
- Element of the array can be C-structure or simple data-type.
- Fast algorithms on machine data-types (int, float, etc.)

```
import numpy as np
from pylab import *

a = np.array([1, 4, 5, 8], float)
b = np.array([1, 2, 3, 4], float)
a=a*b
print(a[1])
plot(a,b)
```


Universal Functions

JOHN BRYCE
Leading in IT Education
a matrix company

- Functions that operate element-by-element and return result
- Fast-loops registered for each fundamental data-type

$$\sin(x) = [\sin(x_i)\ i=0..N]\bullet$$

$$x+y = [x_i + y_i\ i=0..N]\bullet$$

```
import numpy as np
from pylab import *

a = np.array([1, 4, 5, 8], float)
b = np.array([1, 2, 3, 4], float)

c=np.add(a,b)

print(c)
```

Performance

JOHN BRYCE
Leading in IT Education
a matrix company

```
In [20]: %%timeit
    ...: l=range(100000)
    ...: total = 0
    ...: for n in l:
    ...:     total +=n*n
    ...:
100 loops, best of 3: 7.38 ms per loop

In [21]: %%timeit
    ...: n=numpy.arange(100000)
    ...: numpy.sum(n*n)
    ...:
The slowest run took 24.88 times longer than the fastest. This could mean
that an intermediate result is being cached.
10000 loops, best of 3: 174 µs per loop
```


NumPy Array

JOHN BRYCE
Leading in IT Education
a matrix company

- A NumPy array is a homogeneous collection of "items" of the same "data-type" (dtype)

```
>>> import numpy as N
>>> a = N.array([[1,2,3],[4,5,6]],float)
>>> print a
[[1. 2. 3.]
 [4. 5. 6.]]

>>> print a.shape, "\n", a.itemsize
(2, 3)
8

>>> print a.dtype, a.dtype.type
'<f8' <type 'float64scalar'>

>>> type(a[0,0])
<type 'float64scalar'>

>>> type(a[0,0]) is type(a[1,2])
True
```

Introducing NumPy Arrays

JOHN BRYCE
Leading in IT Education
a matrix company

SIMPLE ARRAY CREATION

```
>>> a = array([0,1,2,3])
>>> a
array([0, 1, 2, 3])
```

CHECKING THE TYPE

```
>>> type(a)
<type 'array'>
```

NUMERIC 'TYPE' OF ELEMENTS

```
>>> a.dtype
dtype('int32')
```

BYTES PER ELEMENT

```
>>> a.itemsize # per element
4
```

ARRAY SHAPE

```
# shape returns a tuple
# listing the length of the
# array along each dimension.
>>> a.shape
(4,)
>>> shape(a)
(4,)
```

ARRAY SIZE

```
# size reports the entire
# number of elements in an
# array.
>>> a.size
4
>>> size(a)
4
```


Introducing NumPy Arrays

JOHN BRYCE
Leading in IT Education
a matrix company

BYTES OF MEMORY USED

```
# returns the number of bytes
# used by the data portion of
# the array.
 >>> a.nbytes
12
```

NUMBER OF DIMENSIONS

```
>>> a.ndim
1
```

ARRAY COPY

```
# create a copy of the array
>>> b = a.copy()
>>> b
array([0, 1, 2, 3])
```

CONVERSION TO LIST

```
# convert a numpy array to a
# python list.
>>> a.tolist()
[0, 1, 2, 3]

# For 1D arrays, list also
# works equivalently, but
# is slower.
>>> list(a)
[0, 1, 2, 3]
```

Setting Array Elements

JOHN BRYCE Leading in IT Education a matrix company

ARRAY INDEXING

```
>>> a[0]
0
>>> a[0] = 10
>>> a
[10, 1, 2, 3]
```

FILL

```
# set all values in an array.
>>> a.fill(0)
>>> a
[0, 0, 0, 0]

# This also works, but may
# be slower.
>>> a[:] = 1
>>> a
[1, 1, 1, 1]
```

BEWARE OF TYPE COERSION

```
>>> a.dtype
dtype('int32')

# assigning a float to into # an int32
array will
# truncate decimal part.
>>> a[0] = 10.6
>>> a
[10, 1, 2, 3]

# fill has the same behavior
>>> a.fill(-4.8)
>>> a
[-4, -4, -4, -4]
```


Multi-Dimensional Arrays

JOHN BRYCE Leading in IT Education a matrix company

MULTI-DIMENSIONAL ARRAYS

```
>>> a = array([[ 0, 1, 2, 3],
               [10,11,12,13]])
>>> a
array([[ 0, 1, 2, 3],
       [10,11,12,13]])
```

(ROWS,COLUMNS)

```
>>> a.shape
(2, 4)
>>> shape(a)
(2, 4)
```

ELEMENT COUNT

```
>>> a.size
8
>>> size(a)
8
```

NUMBER OF DIMENSIONS

```
>>> a.ndims
2
```

GET/SET ELEMENTS

```
>>> a[1,3]
13
```

column
row

```
>>> a[1,3] = -1
>>> a
array([[ 0, 1, 2, 3],
       [10,11,12,-1]])
```

ADDRESS FIRST ROW USING SINGLE INDEX

```
>>> a[1]
array([10, 11, 12, -1])
```

Array Slicing

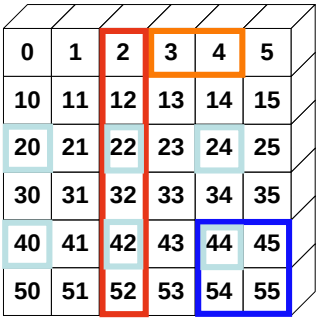
JOHN BRYCE
Leading in IT Education
a matrix company

SLICING WORKS MUCH LIKE STANDARD PYTHON SLICING

```
>>> a[0,3:5]
array([3, 4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])
```

STRIDES ARE ALSO POSSIBLE

```
>>> a[2::2,::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



Memory Model

JOHN BRYCE
Leading in IT Education
a matrix company

```
>>> print a.strides
(24, 8)
>>> print a.flags.fortran, a.flags.contiguous
False True
>>> print a.T.strides
(8, 24)
>>> print a.T.flags.fortran, a.T.flags.contiguous
True False
```

- Every dimension of an ndarray is accessed by stepping (striding) a fixed number of bytes through memory.
- If memory is contiguous, then the strides are "pre-computed" indexing-formulas for either Fortran-order (first-dimension varies the fastest), or C-order (last-dimension varies the fastest) arrays.

Array slicing (Views)

JOHN BRYCE
Leading in IT Education
a matrix company

Memory model allows "simple indexing" (integers and slices) into the array to be a **view** of the same data.

```
>>> b = a[:,::2]
>>> b[0,1] = 100
>>> print a
[[   1.    2.  100.]]
 [   4.    5.    6.]]
>>> c = a[:,::2].copy()
>>> c[1,0] = 500
>>> print a
[[   1.    2.  100.]]
 [   4.    5.    6.]]
```

Other uses of view

```
>>> b = a.view('i8')
>>> [hex(val.item()) for
val in b.flat]
['0x3FF0000000000000L',
 '0x4000000000000000L',
 '0x4059000000000000L',
 '0x4010000000000000L',
 '0x4014000000000000L',
 '0x4018000000000000L']
```


Slices Are References

JOHN BRYCE
Leading in IT Education
a matrix company

Slices are references to memory in original array. Changing values in a slice also changes the original array.

```
>>> a = array((0,1,2,3,4))

# create a slice containing only the
# last element of a
>>> b = a[2:4]
>>> b[0] = 10

# changing b changed a!
>>> a
array([ 1,  2, 10, 3, 4])
```

Fancy Indexing

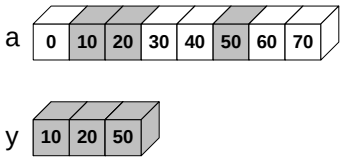
JOHN BRYCE
Leading in IT Education
a matrix company

INDEXING BY POSITION

```
>>> a = arange(0,80,10)


# fancy indexing
>>> y = a[[1, 2, -3]]
>>> print y
[10 20 50]

# using take
>>> y = take(a,[1,2,-3])
>>> print y
[10 20 50]
```

INDEXING WITH BOOLEANS

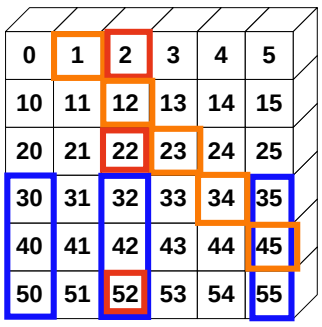
```
>>> mask = array([0,1,1,0,0,1,0,0],
...             dtype=bool)

# fancy indexing
>>> y = a[mask]
>>> print y
[10,20,50]

# using compress
>>> y = compress(mask, a)
>>> print y
[10,20,50]
```



Fancy Indexing in 2D

JOHN BRYCE
Leading in IT Education
a matrix company

```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35],
       [40, 42, 45]])
       [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                 dtype=bool)
>>> a[mask,2]
array([2,22,52])
```



Unlike slicing, fancy indexing creates copies instead of views into original arrays.

Data-types

JOHN BRYCE
Leading in IT Education
a matrix company

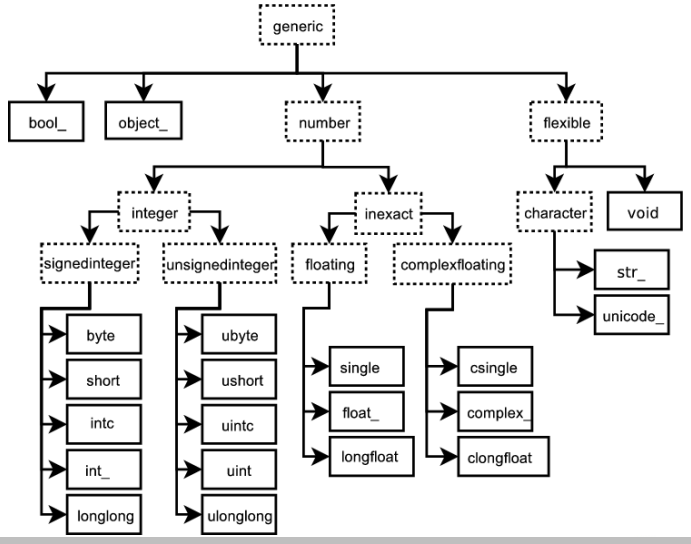
- There are two related concepts of "type"
  - The data-type object (dtype)
  - The Python "type" of the object created from a single array item (hierarchy of scalar types)
- The **dtype** object provides the details of how to interpret the memory for an item. It's an instance of a single dtype class.
- The "type" of the extracted elements are true Python classes that exist in a hierarchy of Python classes
- Every dtype object has a type attribute which provides the Python object returned when an element is selected from the array

NumPy dtypes

JOHN BRYCE
Leading in IT Education
a matrix company

Basic Type	Available NumPy types	Comments
Boolean	`bool`	`Elements are 1 byte in size`
Integer	`int8, int16, int32, int64, int128, int`	`int defaults to the size of int in C for the platform`
Unsigned Integer	`uint8, uint16, uint32, uint64, uint128, uint`	`uint defaults to the size of unsigned int in C for the platform`
Float	`float32, float64, float, longfloat,`	`Float is always a double precision floating point value (64 bits). longfloat represents large precision floats. Its size is platform dependent.`
Complex	`complex64, complex128, complex`	`The real and complex elements of a complex64 are each represented by a single precision (32 bit) value for a total size of 64 bits.`
Strings	`str, unicode`	`Unicode is always UTF32 (UCS4)`
Object	`object`	`Represent items in array as Python objects.`
Records	`void`	`Used for arbitrary data structures in record arrays.`

Built-in "scalar" types



Data-type object (dtype)

- There are 21 "built-in" (static) data-type objects
- New (dynamic) data-type objects are created to handle
  - Alteration of the byteorder
  - Change in the element size (for string, unicode, and void built-ins)
  - Addition of fields
  - Change of the type object (C-structure arrays)
- Creation of data-types is quite flexible.
- New user-defined "built-in" data-types can also be added (but must be done in C and involves filling a function-pointer table)

Data-type fields

JOHN BRYCE
Leading in IT Education
a matrix company

- An item can include fields of different data-types.
- A field is described by a data-type object and a byte offset --- this definition allows nested records.
- The array construction command interprets tuple elements as field entries.

```
>>> dt = N.dtype("i4,f8,a5")

>>> print dt.fields
{'f1': (dtype('<i4'), 0), 'f2': (dtype('<f8'), 4), 'f3':
(dtype('|S5'), 12)}

>>> a = N.array([(1,2.0,"Hello"), (2,3.0,"World")], dtype=dt)

>>> print a['f2']
[Hello World]
```


Array Calculation Methods

JOHN BRYCE
Leading in IT Education
a matrix company

SUM FUNCTION

```
>>> a = array([[1,2,3],
              [4,5,6]], float)

# Sum defaults to summing all
# *all* array values.
>>> sum(a)
21.

# supply the keyword axis to
# sum along the 0th axis.
>>> sum(a, axis=0)
array([5., 7., 9.])

# supply the keyword axis to
# sum along the last axis.
>>> sum(a, axis=-1)
array([6., 15.])
```

SUM ARRAY METHOD

```
# The a.sum() defaults to
# summing *all* array values
>>> a.sum()
21.

# Supply an axis argument to
# sum along a specific axis.
>>> a.sum(axis=0)
array([5., 7., 9.])
```

PRODUCT

```
# product along columns.
>>> a.prod(axis=0)
array([ 4., 10., 18.])

# functional form.
>>> prod(a, axis=0)
array([ 4., 10., 18.])
```

Min/Max

JOHN BRYCE
Leading in IT Education
a matrix company

MIN

```
>>> a = array([2.,3.,0.,1.])
>>> a.min(axis=0)
0.
# use Numpy's amin() instead
# of Python's builtin min()
# for speed operations on
# multi-dimensional arrays.
>>> amin(a, axis=0)
0.
```

MAX

```
>>> a = array([2.,1.,0.,3.]) >>>
a.max(axis=0)
3.


# functional form
>>> amax(a, axis=0)
3.
```

ARGMIN

```
# Find index of minimum value.
>>> a.argmin(axis=0)
2
# functional form
>>> argmin(a, axis=0)
2
```

ARGMAX

```
# Find index of maximum value.
>>> a.argmax(axis=0)
1
# functional form
>>> argmax(a, axis=0)
1
```


Statistics Array Methods

JOHN BRYCE
Leading in IT Education
a matrix company

MEAN

```
>>> a = array([[1,2,3],
              [4,5,6]], float)

# mean value of each column
>>> a.mean(axis=0)
array([ 2.5,  3.5,  4.5])
>>> mean(a, axis=0)
array([ 2.5,  3.5,  4.5])
>>> average(a, axis=0)
array([ 2.5,  3.5,  4.5])

# average can also calculate
# a weighted average
>>> average(a, weights=[1,2],
...         axis=0)
array([ 3., 4., 5.])
```

STANDARD DEV./VARIANCE

```
# Standard Deviation
>>> a.std(axis=0)
array([ 1.5,  1.5,  1.5])

# Variance
>>> a.var(axis=0)
array([2.25, 2.25, 2.25])
>>> var(a, axis=0)
array([2.25, 2.25, 2.25])
```

Other Array Methods

JOHN BRYCE
Leading in IT Education
a matrix company

CLIP

```
# Limit values to a range

>>> a = array([[1,2,3],
               [4,5,6]], float)

# Set values < 3 equal to 3.
# Set values > 5 equal to 5.
>>> a.clip(3,5)
>>> a
array([[ 3.,  3.,  3.],
       [ 4.,  5.,  5.]])
```

ROUND

```
# Round values in an array.
# Numpy rounds to even, so
# 1.5 and 2.5 both round to 2.
>>> a = array([1.35, 2.5, 1.5])
>>> a.round()
array([ 1.,  2.,  2.])

# Round to first decimal place.
>>> a.round(decimals=1)
array([ 1.4,  2.5,  1.5])
```

POINT TO POINT

```
# Calculate max – min for
# array along columns
>>> a.ptp(axis=0)
array([ 3.0,  3.0,  3.0])
# max – min for entire array.
>>> a.ptp(axis=None)
5.0
```


Array attributes/methods

JOHN BRYCE
Leading in IT Education
a matrix company

BASIC ATTRIBUTES

```
a.dtype – Numerical type of array elements. float32, uint8, etc.
a.shape – Shape of the array. (m,n,o,...)
a.size – Number of elements in entire array.
a.itemsize – Number of bytes used by a single element in the array.
a.nbytes – Number of bytes used by entire array (data only).
a.ndim – Number of dimensions in the array.
```

SHAPE OPERATIONS

```
a.flat – An iterator to step through array as if it is 1D.
a.flatten() – Returns a 1D copy of a multi-dimensional array.
a.ravel() – Same as flatten(), but returns a 'view' if possible.
a.resize(new_size) – Change the size/shape of an array in-place.
a.swapaxes(axis1, axis2) – Swap the order of two axes in an array.
a.transpose(*axes) – Swap the order of any number of array axes.
a.T – Shorthand for a.transpose()
a.squeeze() – Remove any length=1 dimensions from an array.
```

Array attributes/methods

JOHN BRYCE
Leading in IT Education
a matrix company

FILL AND COPY

```
a.copy() – Return a copy of the array.
a.fill(value) – Fill array with a scalar value.
```

CONVERSION / COERSION

```
a.tolist() – Convert array into nested lists of values.
a.tostring() – raw copy of array memory into a python string.
a.astype(dtype) – Return array coerced to given dtype.
a.byteswap(False) – Convert byte order (big <-> little endian).
```

COMPLEX NUMBERS

```
a.real – Return the real part of the array.
a.imag – Return the imaginary part of the array.
a.conjugate() – Return the complex conjugate of the array.
a.conj()– Return the complex conjugate of an array.(same as conjugate)
```


Array attributes/methods

JOHN BRYCE
Leading in IT Education
a matrix company

SAVING

```
a.dump(file) – Store a binary array data out to the given file.
a.dumps() – returns the binary pickle of the array as a string.
a.tofile(fid, sep="", format="%s") Formatted ascii output to file.
```

SEARCH / SORT

```
a.nonzero() – Return indices for all non-zero elements in a.
a.sort(axis=-1) – Inplace sort of array elements along axis.
a.argsort(axis=-1) – Return indices for element sort order along axis.
a.searchsorted(b) – Return index where elements from b would go in a.
```

ELEMENT MATH OPERATIONS

```
a.clip(low, high) – Limit values in array to the specified range.
a.round(decimals=0) – Round to the specified number of digits.
a.cumsum(axis=None) – Cumulative sum of elements along axis.
a.cumprod(axis=None) – Cumulative product of elements along axis.
```

Array attributes/methods

JOHN BRYCE
Leading in IT Education
a matrix company

REDUCTION METHODS

All the following methods "reduce" the size of the array by 1 dimension by carrying out an operation along the specified axis. If axis is None, the operation is carried out across the entire array.

```
a.sum(axis=None) - Sum up values along axis.
a.prod(axis=None) - Find the product of all values along axis.
a.min(axis=None)- Find the minimum value along axis.
a.max(axis=None) - Find the maximum value along axis.
a.argmin(axis=None) - Find the index of the minimum value along axis.
a.argmax(axis=None) - Find the index of the maximum value along axis.
a.ptp(axis=None) - Calculate a.max(axis) - a.min(axis)
a.mean(axis=None) - Find the mean (average) value along axis.
a.std(axis=None) - Find the standard deviation along axis.
a.var(axis=None) - Find the variance along axis.

a.any(axis=None) - True if any value along axis is non-zero.   (or)
a.all(axis=None) - True if all values along axis are non-zero. (and)
```


Array Operations

JOHN BRYCE
Leading in IT Education
a matrix company

SIMPLE ARRAY MATH

```
>>> a = array([1,2,3,4])
>>> b = array([2,3,4,5])
>>> a + b
array([3, 5, 7, 9])
```

Numpy defines the following constants:
pi = 3.14159265359
e = 2.71828182846

MATH FUNCTIONS

```
# Create array from 0 to 10
>>> x = arange(11.)

# multiply entire array by
# scalar value
>>> a = (2*pi)/10.
>>> a
0.62831853071795862
>>> a*x
array([ 0.,0.628,…,6.283])

# inplace operations
>>> x *= a
>>> x
array([ 0.,0.628,…,6.283])

# apply functions to array.
>>> y = sin(x)
```

Universal Functions

JOHN BRYCE Leading in IT Education a matrix company

- ufuncs are objects that rapidly evaluate a function element-by-element over an array.
- Core piece is a 1-d loop written in C that performs the operation over the largest dimension of the array
- For 1-d arrays it is equivalent to but much faster than list comprehension

```
>>> type(N.exp)
<type 'numpy.ufunc'>
>>> x = array([1,2,3,4,5])
>>> print N.exp(x)
[   2.71828183    7.3890561    20.08553692
54.59815003  148.4131591 ]
>>> print [math.exp(val) for val in x]
[2.7182818284590451,
7.3890560989306504,20.085536923187668,
 54.598150033144236,148.4131591025766]
```


Mathematic Binary Operators

JOHN BRYCE Leading in IT Education a matrix company

a + b → add(a,b)
a - b → subtract(a,b)
a % b → remainder(a,b)

a * b → multiply(a,b)
a / b → divide(a,b)
a ** b → power(a,b)

MULTIPLY BY A SCALAR

```
>>> a = array((1,2))
>>> a*3.
array([3., 6.])
```

ELEMENT BY ELEMENT ADDITION

```
>>> a = array([1,2])
>>> b = array([3,4])
>>> a + b
array([4, 6])
```

ADDITION USING AN OPERATOR FUNCTION

```
>>> add(a,b)
array([4, 6])
```

IN PLACE OPERATION

```
# Overwrite contents of a.
# Saves array creation
# overhead
>>> add(a,b,a) # a += b
array([4, 6])
>>> a
array([4, 6])
```

Comparison and Logical Operators

JOHN BRYCE
Leading in IT Education
a matrix company

equal (==)	not_equal (!=)	greater (>)
greater_equal (>=)	less (<)	less_equal (<=)
logical_and	logical_or	logical_xor
logical_not		

2D EXAMPLE

```
>>> a = array(((1,2,3,4),(2,3,4,5)))
>>> b = array(((1,2,5,4),(1,3,4,5)))
>>> a == b
array([[True, True, False, True], [False, True, True, True]])
# functional equivalent
>>> equal(a,b)
array([[True, True, False, True], [False, True, True, True]])
```


Bitwise Operators

JOHN BRYCE
Leading in IT Education
a matrix company

bitwise_and (&)	invert (~)	right_shift(a,shifts)
bitwise_or (|)	bitwise_xor	left_shift (a,shifts)

BITWISE EXAMPLES

```
>>> a = array((1,2,4,8))
>>> b = array((16,32,64,128))
>>> bitwise_or(a,b)
array([ 17,  34,  68, 136])

# bit inversion
>>> a = array((1,2,3,4), uint8)
>>> invert(a)
array([254, 253, 252, 251], dtype=uint8)

# left shift operation
>>> left_shift(a,3)
array([ 8, 16, 24, 32], dtype=uint8)
```

Trig and Other Functions

JOHN BRYCE
Leading in IT Education
a matrix company

TRIGONOMETRIC

```
sin(x)        sinh(x)
cos(x)        cosh(x)
arccos(x)     arccosh(x)

arctan(x)     arctanh(x)
arcsin(x)     arcsinh(x)
arctan2(x,y)
```

OTHERS

```
exp(x)        log(x)
log10(x)      sqrt(x)
absolute(x)   conjugate(x)
negative(x)   ceil(x)
floor(x)      fabs(x)
hypot(x,y)    fmod(x,y)
maximum(x,y)  minimum(x,y)
```

hypot(x,y)

Element by element distance calculation using $\sqrt{x^2 + y^2}$

Broadcasting

JOHN BRYCE
Leading in IT Education
a matrix company

When there are multiple inputs, then they all must be "broadcastable" to the same shape.

- All arrays are promoted to the same number of dimensions (by pre-prending 1's to the shape)
- All dimensions of length 1 are expanded as determined by other inputs with non-unit lengths in that dimension.

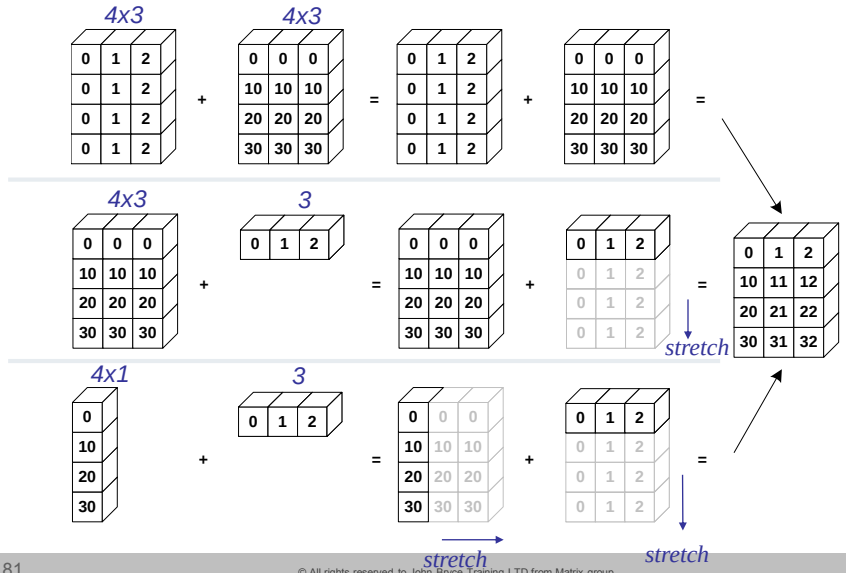
```
>>> x = [1,2,3,4];
>>> y =
[[10],[20],[30]]
>>> print N.add(x,y)
[[11 12 13 14]
 [21 22 23 24]
 [31 32 33 34]]
>>> x = array(x)
>>> y = array(y)
>>> print x+y
[[11 12 13 14]
 [21 22 23 24]
 [31 32 33 34]]
```

x has shape (4,) the ufunc sees it as having shape (1,4)

y has shape (3,1)

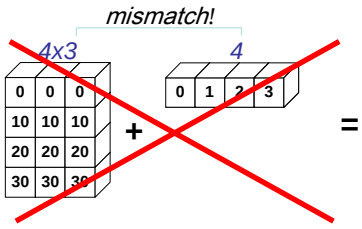
The ufunc result has shape (3,4)

Array Broadcasting



Broadcasting Rules

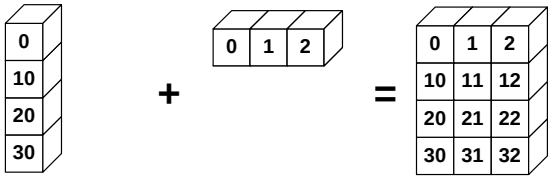
The *trailing* axes of both arrays must either be 1 or have the same size for broadcasting to occur. Otherwise, a "`ValueError: frames are not aligned`" exception is thrown.



Broadcasting in Action

JOHN BRYCE
Leading in IT Education
a matrix company

```
>>> a = array((0,10,20,30))
>>> b = array((0,1,2))
>>> y = a[:, None] + b
```



Universal Function Methods

JOHN BRYCE
Leading in IT Education
a matrix company

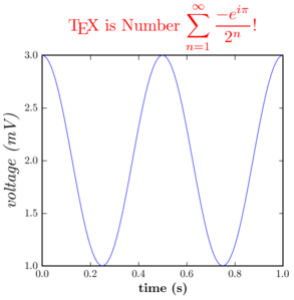
The mathematic, comparative, logical, and bitwise operators that take two arguments (binary operators) have special methods that operate on arrays:

```
op.reduce(a,axis=0)
op.accumulate(a,axis=0)
op.outer(a,b)
op.reduceat(a,indices)
```

Matplotlib

Matplotlib

- Requires NumPy extension. Provides powerful plotting commands.
- http://matplotlib.sourceforge.net



Recommendations

JOHN BRYCE
Leading in IT Education
a matrix company

- Matplotlib for day-to-day data exploration.

  Matplotlib has a large community, tons of plot types, and is well integrated into ipython. It is the de-facto standard for 'command line' plotting from ipython.

- Chaco for building interactive plotting applications

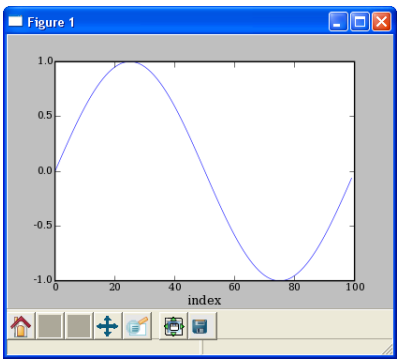
  Chaco is architected for building highly interactive and configurable plots in python. It is more useful as plotting toolkit than for making one-off plots.

Line Plots

JOHN BRYCE
Leading in IT Education
a matrix company

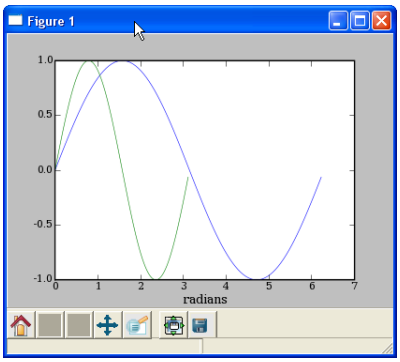
PLOT AGAINST INDICES

```
>>> x = arange(50)*2*pi/50.
>>> y = sin(x)
>>> plot(y)
>>> xlabel('index')
```



MULTIPLE DATA SETS

```
>>> plot(x,y,x2,y2)
>>> xlabel('radians')
```

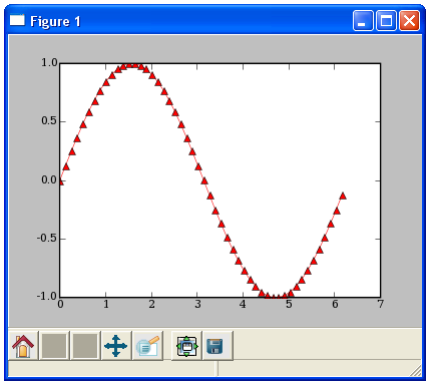


Line Plots

JOHN BRYCE
Leading in IT Education
a matrix company

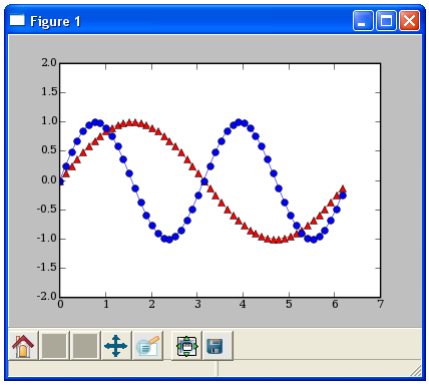
LINE FORMATTING

```
# red, dot-dash, triangles
>>> plot(x,sin(x),'r-^')
```



MULTIPLE PLOT GROUPS

```
>>> plot(x,y1,'b-o', x,y2), r-^')
>>> axis([0,7,-2,2])
```

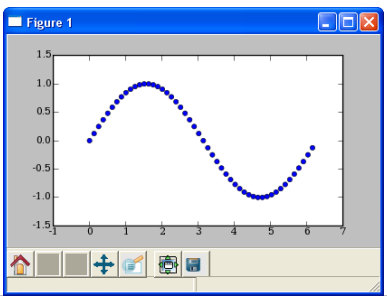


Scatter Plots

JOHN BRYCE
Leading in IT Education
a matrix company

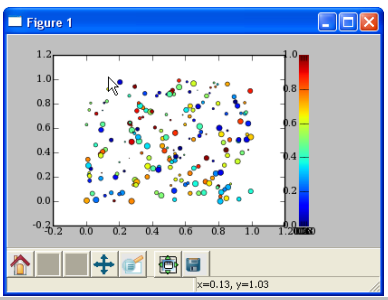
SIMPLE SCATTER PLOT

```
>>> x = arange(50)*2*pi/50.
>>> y = sin(x)
>>> scatter(x,y)
```



COLORMAPPED SCATTER

```
# marker size/color set with data
>>> x = rand(200)
>>> y = rand(200)
>>> size = rand(200)*30
>>> color = rand(200)
>>> scatter(x, y, size, color)
>>> colorbar()
```

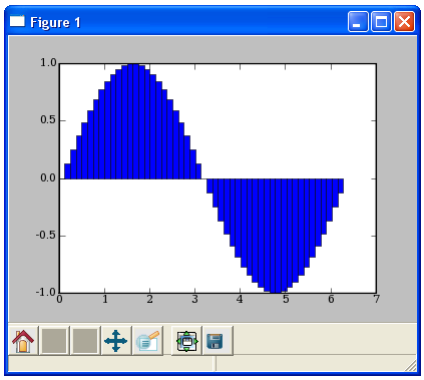


Bar Plots

JOHN BRYCE
Leading in IT Education
a matrix company

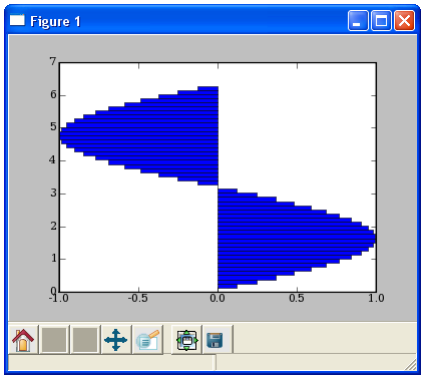
BAR PLOT

```
>>> bar(x,sin(x),
...     width=x[1]-x[0])
```



HORIZONTAL BAR PLOT

```
>>> barh(x,sin(x),
...     height=x[1]-x[0])
```



Bar Plots

JOHN BRYCE
Leading in IT Education
a matrix company

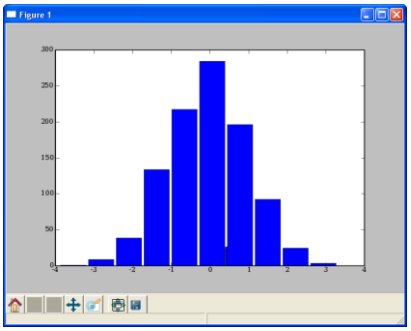
DEMO/MATPLOTLIB_PLOTTING/BARCHART_DEMO.PY

```
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()
```


Histograms

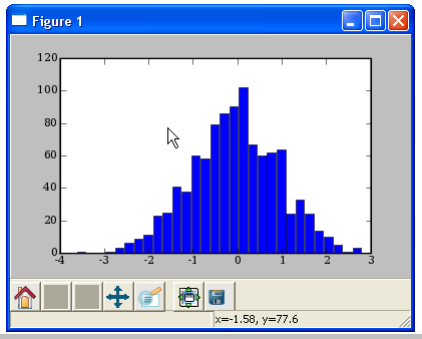
HISTOGRAM

# plot histogram
# default to 10 bins
>>> hist(randn(1000))



HISTOGRAM 2

# change the number of bins
>>> hist(randn(1000), 30)



Multiple Plots using Subplot

JOHN BRYCE
Leading in IT Education
a matrix company

DEMO/MATPLOTLIB_PLOTTING/EXAMPLES/SUBPLOT_DEMO.PY

```
def f(t):
    s1 = cos(2*pi*t)
    e1 = exp(-t)
    return multiply(s1,e1)

t1 = arange(0.0, 5.0, 0.1)
t2 = arange(0.0, 5.0, 0.02)
t3 = arange(0.0, 2.0, 0.01)

subplot(211)
l = plot(t1, f(t1), 'bo', t2, f(t2),
        'k--')
setp(l, 'markerfacecolor', 'g')
grid(True)
title('A tale of 2 subplots')
ylabel('Damped oscillation')

subplot(212)
plot(t3, cos(2*pi*t3), 'r.')
grid(True)
xlabel('time (s)')
ylabel('Undamped')
show()
```

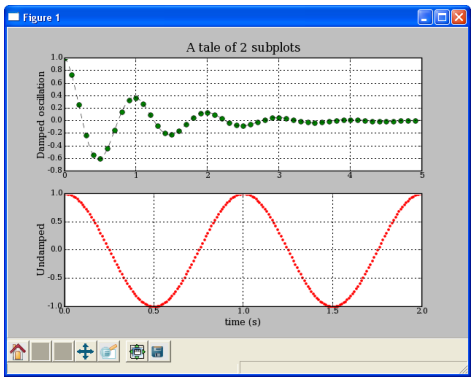


Image demo

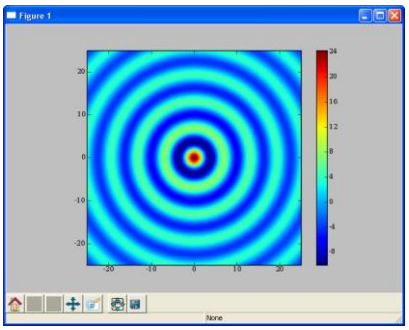
JOHN BRYCE
Leading in IT Education
a matrix company

```
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook

image_file = cbook.get_sample_data('/Users/liran/hires.png')
image = plt.imread(image_file)

plt.imshow(image)
plt.axis('off') # clear x- and y-axes
plt.show()
```

Image Display

JOHN BRYCE
Leading in IT Education
a matrix company

```
# Create 2d array where values
# are radial distance from
# the center of array.
>>> from numpy import mgrid
>>> from scipy import special
>>> x,y = mgrid[-25:25:100j,
...             -25:25:100j]
>>> r = sqrt(x**2+y**2)
# Calculate bessel function of
# each point in array and scale
>>> s = special.j0(r)*25

# Display surface plot.
>>> imshow(s, extent=[-25,25,-25,25])
>>> colorbar()
```



Animation

JOHN BRYCE
Leading in IT Education
a matrix company

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

def update_line(num, data, line):
    line.set_data(data[...,:num])
    return line,

fig1 = plt.figure()

data = np.random.rand(2, 25)
l, = plt.plot([], [], 'r-')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel('x')
plt.title('test')
line_ani = animation.FuncAnimation(fig1, update_line, 25, fargs=(data, l),
    interval=50, blit=True)
#line_ani.save('lines.mp4')

fig2 = plt.figure()

x = np.arange(-9, 10)
y = np.arange(-9, 10).reshape(-1, 1)
base = np.hypot(x, y)
ims = []
for add in np.arange(15):
    ims.append((plt.pcolor(x, y, base + add, norm=plt.Normalize(0, 30)),))

im_ani = animation.ArtistAnimation(fig2, ims, interval=50, repeat_delay=3000,
    blit=True)
#im_ani.save('/Users/liran/im.mp4', metadata={'artist':'Guido'})

plt.show()
```

Events

JOHN BRYCE
Leading in IT Education
a matrix company

```
from __future__ import print_function
import matplotlib.pyplot as plt

def handle_close(evt):
    print('Closed Figure!')

fig = plt.figure()
fig.canvas.mpl_connect('close_event', handle_close)

plt.text(0.35, 0.5, 'Close Me!', dict(size=30))
plt.show()
```


Saving figures

JOHN BRYCE
Leading in IT Education
a matrix company

- Matplotlib can generate high-quality output in a number formats, including PNG, JPG, EPS, SVG, PGF and PDF.
- fig.savefig("filename.png")
- fig.savefig("filename.png", dpi=200)

JOHN BRYCE
Leading in IT Education
a matrix company

SciPy

SciPy Overview

JOHN BRYCE
Leading in IT Education
a matrix company

- Available at www.scipy.org
- Open Source BSD Style License
- Over 30 svn “committers” to the project

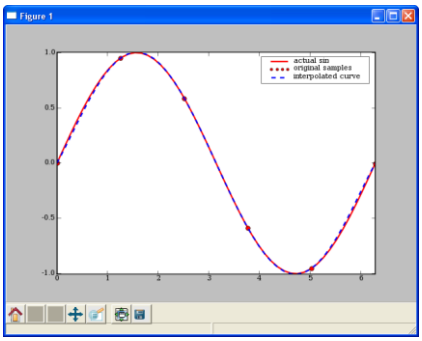
CURRENT PACKAGES

- Special Functions (scipy.special)
- Signal Processing (scipy.signal)
- Image Processing (scipy.ndimage)
- Fourier Transforms (scipy.fftpack)
- Optimization (scipy.optimize)
- Numerical Integration (scipy.integrate)
- Linear Algebra (scipy.linalg)
- Input/Output (scipy.io)
- Statistics (scipy.stats)
- Fast Execution (scipy.weave)
- Clustering Algorithms (scipy.cluster)
- Sparse Matrices (scipy.sparse)
- Interpolation (scipy.interpolate)
- More (e.g. scipy.odr, scipy.maxentropy)

1D Spline Interpolation

JOHN BRYCE
Leading in IT Education
a matrix company

```
from scipy.interpolate import interp1d
from pylab import plot, axis, legend
from numpy import linspace

# sample values
x = linspace(0,2*pi,6)
y = sin(x)

# Create a spline class for interpolation.
# kind=5 sets to 5th degree spline.
# kind=0 -> zeroth order hold.
# kind=1 or 'linear' -> linear interpolation
# kind=2 or
spline_fit = interp1d(x,y,kind=5)
xx = linspace(0,2*pi, 50)
yy = spline_fit(xx)

# display the results.
plot(xx, sin(xx), 'r-', x,y,'ro',xx,yy, 'b--',linewidth=2)
axis('tight')
legend(['actual sin', 'original samples', 'interpolated curve'])
```

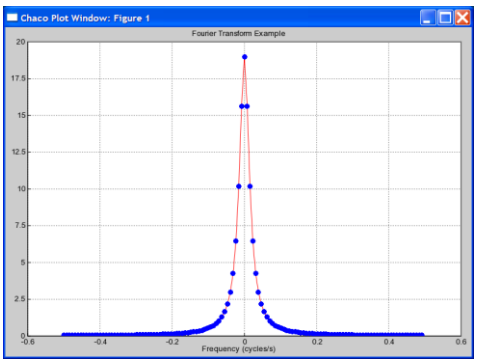


FFT

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.fft --- FFT and related functions

```
>>> n = fftfreq(128)*128
>>> f = fftfreq(128)
>>> ome = 2*pi*f
>>> x = (0.9)**abs(n)
>>> X = fft(x)
>>> z = exp(1j*ome)
>>> Xexact = (0.9**2 - 1)/0.9*z /   \
...          (z-0.9) / (z-1/0.9)
>>> f = fftshift(f)
>>> plot(f, fftshift(X.real),'r-',
...      f, fftshift(Xexact.real),'bo')
>>> title('Fourier Transform Example')
>>> xlabel('Frequency (cycles/s)')
>>> axis(-0.6,0.6, 0, 20)
```

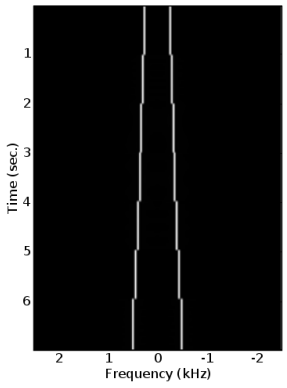


FFT

JOHN BRYCE
Leading in IT Education
a matrix company

EXAMPLE --- Short-Time Windowed Fourier Transform

```
rate, data = read('scale.wav')
dT, T_window = 1.0/rate, 50e-3
N_window = int(T_window * rate)
N_data = len(data)
window = get_window('hamming', N_window)
result, start = [], 0
# compute short-time FFT for each block
while (start < N_data – N_window):
      end = start + N_window
   val = fftshift(fft(window*data[start:end]))
   result.append(val)
   start = end
lastval = fft(window*data[-N_window:])
result.append(fftshift(lastval))
result = array(result,result[0].dtype)
```



Signal Processing

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.signal --- Signal and Image Processing

What's Available?

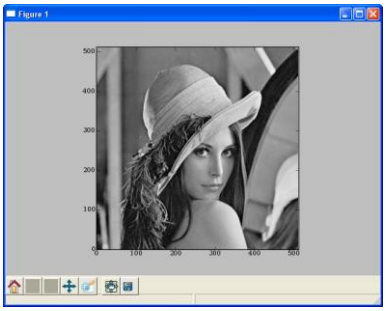
- Filtering
  - General 2-D Convolution (more boundary conditions)
  - N-D convolution
  - B-spline filtering
  - N-D Order filter, N-D median filter, faster 2d version,
  - IIR and FIR filtering and filter design
- LTI systems
  - System simulation
  - Impulse and step responses
  - Partial fraction expansion

Image Processing

JOHN BRYCE
Leading in IT Education
a matrix company

```
# The famous lena image is packaged with scipy
>>> from scipy import misc.lena, signal
>>> lena = lena().astype(float32)
>>> imshow(lena, cmap=cm.gray)
# Blurring using a median filter
>>> fl = signal.medfilt2d(lena, [15,15])
>>> imshow(fl, cmap=cm.gray)
```

LENA IMAGE



MEDIAN FILTERED IMAGE

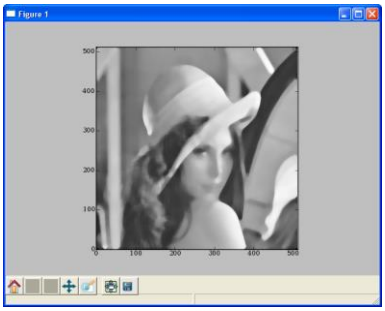
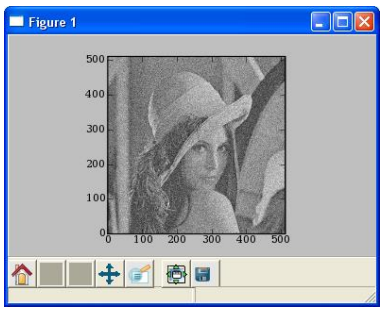


Image Processing

JOHN BRYCE
Leading in IT Education
a matrix company

```
# Noise removal using wiener filter
>>> from scipy.stats import norm
>>> ln = lena + norm(0,32).rvs(lena.shape)
>>> imshow(ln)
>>> cleaned = signal.wiener(ln)
>>> imshow(cleaned)
```

NOISY IMAGE



FILTERED IMAGE

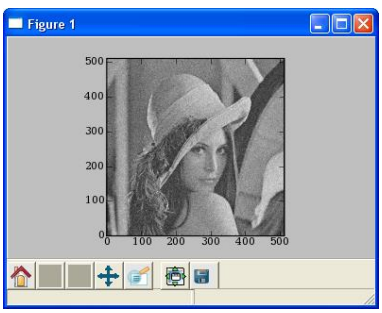
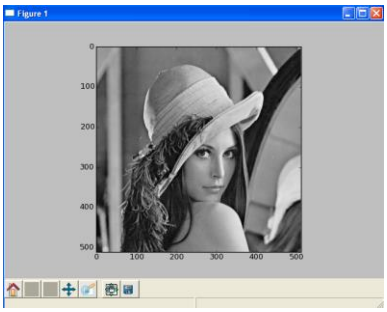


Image Processing

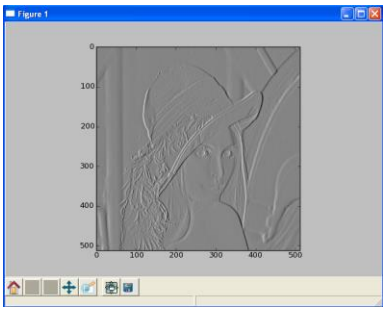
JOHN BRYCE
Leading in IT Education
a matrix company

```
# Edge detection using Sobel filter
>>> from scipy.ndimage.filters import sobel
>>> imshow(lena)
>>> edges = sobel(lena)
>>> imshow(edges)
```

NOISY IMAGE



FILTERED IMAGE

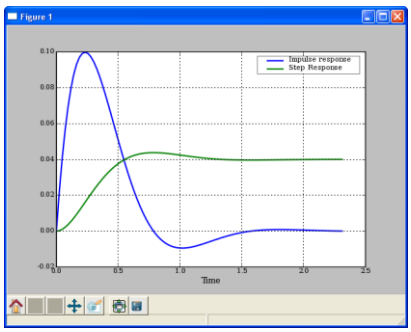


LTI Systems

JOHN BRYCE
Leading in IT Education
a matrix company

```
>>> b,a = [1],[1,6,25]
>>> ltisys = signal.lti(b,a)
>>> t,h = ltisys.impulse()
>>> ts,s = ltisys.step()
>>> plot(t,h,ts,s)
>>> legend(['Impulse response','Step response'])
```

$$H(s) = \frac{1}{s^2 + 6s + 25}$$



Optimization

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.optimize --- unconstrained minimization and root finding

- Unconstrained Optimization

  `fmin` (Nelder-Mead simplex), `fmin_powell` (Powell's method), `fmin_bfgs` (BFGS quasi-Newton method), `fmin_ncg` (Newton conjugate gradient), `leastsq` (Levenberg-Marquardt), `anneal` (simulated annealing global minimizer), `brute` (brute force global minimizer), brent (excellent 1-D minimizer), golden, bracket

- Constrained Optimization

  `fmin_l_bfgs_b`, `fmin_tnc` (truncated newton code), `fmin_cobyla` (constrained optimization by linear approximation), `fminbound` (interval constrained 1-d minimizer)

- Root finding

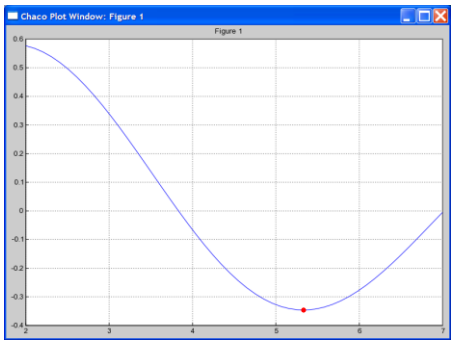
  `fsolve` (using MINPACK), `brentq`, `brenth`, `ridder`, `newton`, `bisect`, `fixed_point` (fixed point equation solver)

Optimization

JOHN BRYCE
Leading in IT Education
a matrix company

EXAMPLE: MINIMIZE BESSEL FUNCTION

```
# minimize 1st order bessel
# function between 4 and 7
>>> from scipy.special import j1
>>> from scipy.optimize import \
    fminbound

>>> x = r_[2:7.1:.1]
>>> j1x = j1(x)
>>> plot(x,j1x,'-')
>>> hold(True)
>>> x_min = fminbound(j1,4,7)
>>> j1_min = j1(x_min)
>>> plot([x_min],[j1_min],'ro')
```



Optimization

JOHN BRYCE
Leading in IT Education
a matrix company

EXAMPLE: SOLVING NONLINEAR EQUATIONS

Solve the non-linear equations

$$3x_0 - \cos(x_1 x_2) + a = 0$$
$$x_0^2 - 81(x_1 + 0.1)^2 + \sin(x_2) + b = 0$$
$$e^{-x_0 x_1} + 20x_2 + c = 0$$

```
>>> def nonlin(x,a,b,c):
>>>     x0,x1,x2 = x
>>>     return [3*x0-cos(x1*x2)+ a,
>>>             x0*x0-81*(x1+0.1)**2                    +
sin(x2)+b,
>>>             exp(-x0*x1)+20*x2+c]
>>> a,b,c = -0.5,1.06,(10*pi-3.0)/3
>>> root = optimize.fsolve(nonlin,   [0.1,0.1,-
0.1],args=(a,b,c))
>>> print root
[ 0.5     0.     -0.5236]
>>> print nonlin(root,a,b,c)
[0.0, -2.231104190e-12, 7.46069872e-14]
```

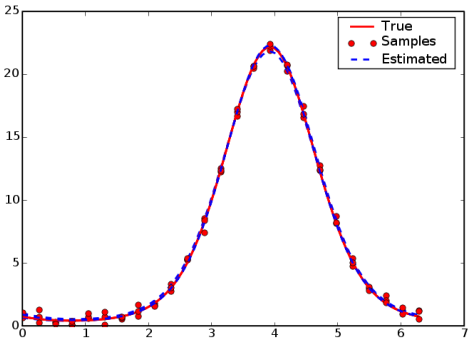
starting location for search

Optimization

JOHN BRYCE
Leading in IT Education
a matrix company

EXAMPLE: Non-linear least-squares data fitting

```
# fit data-points to a curve
# demo/data_fitting/datafit.py
>>> from numpy.random import randn
>>> from numpy import exp, sin, pi
>>> from numpy import linspace
>>> from scipy.optimize import leastsq
>>> def func(x,A,a,f,phi):
    return A*exp(-a*sin(f*x+pi/4))
>>> def errfunc(params, x, data):
    return func(x, *params) - data
>>> ptrue = [3,2,1,pi/4]
>>> x = linspace(0,2*pi,25)
>>> true = func(x, *ptrue)
>>> noisy = true + 0.3*randn(len(x))
>>> p0 = [1,1,1,1]
>>> pmin, ier = leastsq(errfunc, p0,
                args=(x, noisy))
>>> pmin
array([3.1705, 1.9501, 1.0206, 0.7034])
```

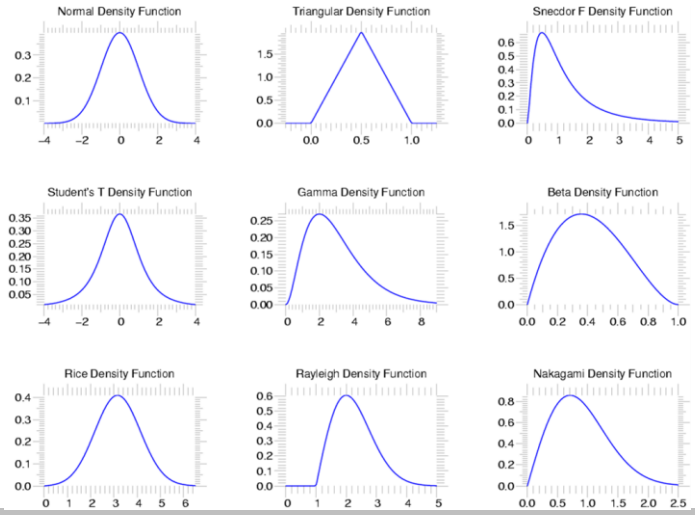


Statistics

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.stats --- CONTINUOUS DISTRIBUTIONS

over 80 continuous distributions!



METHODS

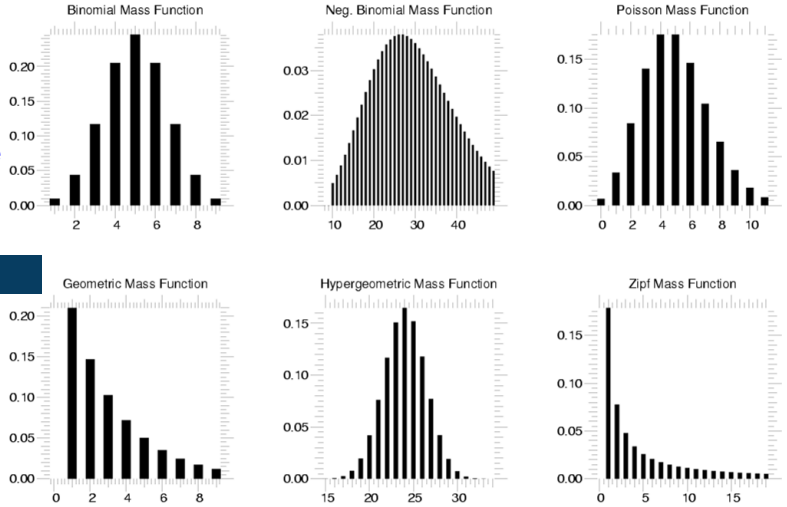
- pdf
- cdf
- rvs
- ppf
- stats

Statistics

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.stats --- Discrete Distributions

10 standard discrete distributions (plus any arbitrary finite RV)



METHODS

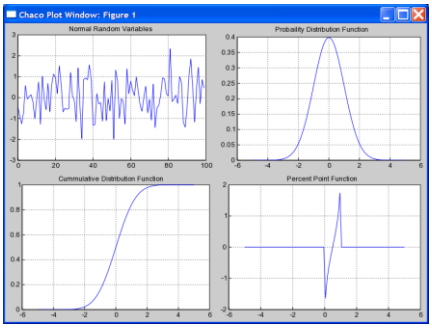
- pdf
- cdf
- rvs
- ppf
- stats

Using stats objects

JOHN BRYCE
Leading in IT Education
a matrix company

DISTRIBUTIONS

```
# Sample normal dist. 100 times.
>>> samp = stats.norm.rvs(size=100)

>>> x = r_[-5:5:100j]
# Calculate probability dist.
>>> pdf = stats.norm.pdf(x)
# Calculate cummulative Dist.
>>> cdf = stats.norm.cdf(x)
# Calculate Percent Point Function
>>> ppf = stats.norm.ppf(x)
```



Statistics

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.stats --- Basic Statistical Calculations on Data

- numpy.mean, numpy.std, numpy.var, numpy.cov
- stats.skew, stats.kurtosis, stats.moment

scipy.stats.bayes_mvs --- Bayesian mean, variance, and std.

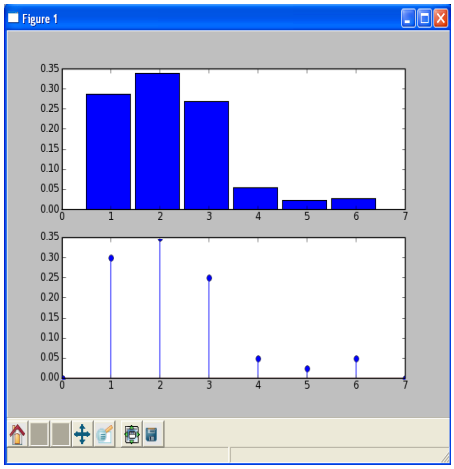
```
# Create “frozen” Gamma distribution with a=2.5
>>> grv = stats.gamma(2.5)
>>> grv.stats()    # Theoretical mean and variance
(array(2.5), array(2.5))
# Estimate mean, variance, and std with 95% confidence
>>> vals = grv.rvs(size=100)
>>> stats.bayes_mvs(vals, alpha=0.95)
((2.52887906081, (2.19560839724, 2.86214972438)),
 (2.87924964268, (2.17476164549, 3.8070215789)),
 (1.69246760584, (1.47470730841, 1.95115903475)))
# (expected value and confidence interval for each of
# mean, variance, and standard-deviation)
```

Using stats objects

JOHN BRYCE
Leading in IT Education
a matrix company

CREATING NEW DISCRETE DISTRIBUTIONS

```
# Create a loaded dice.
>>> from scipy.stats import rv_discrete
>>> xk = [1,2,3,4,5,6]
>>> pk = [0.3,0.35,0.25,0.05,
          0.025,0.025]
>>> new = rv_discrete(name='loaded',
                      values=(xk,pk))

# Calculate histogram
>>> samples = new.rvs(size=1000)
>>> bins=linspace(0.5,5.5,6)
>>> subplot(211)
>>> hist(samples,bins=bins,normed=True)
# Calculate pmf
>>> x = range(0,8)
>>> subplot(212)
>>> stem(x,new.pmf(x))
```

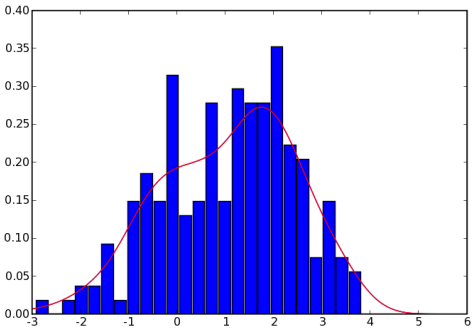


Statistics

JOHN BRYCE
Leading in IT Education
a matrix company

Continuous PDF Estimation using Gaussian Kernel Density Estimation

```
# Sample normal dist. 100 times.
>>> rv1 = stats.norm()
>>> rv2 = stats.norm(2.0,0.8)
>>> samp = r_[rv1.rvs(size=100),
              rv2.rvs(size=100)]
# Kernel estimate (smoothed histogram)
>>> apdf = stats.kde.gaussian_kde(samp)
>>> x = linspace(-3,6,200)
>>> plot(x, apdf(x),'r')

# Histogram
>>> hist(x, bins=25, normed=True)
```



Linear Algebra

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.linalg --- FAST LINEAR ALGEBRA

- Uses ATLAS if available --- very fast
- Low-level access to BLAS and LAPACK routines in modules linalg.fblas, and linalg.flapack (FORTRAN order)
- High level matrix routines
  - Linear Algebra Basics: inv, solve, det, norm, lstsq, pinv
  - Decompositions: eig, lu, svd, orth, cholesky, qr, schur
  - Matrix Functions: expm, logm, sqrtm, cosm, coshm, funm (general matrix functions)

Linear Algebra

JOHN BRYCE
Leading in IT Education
a matrix company

LU FACTORIZATION

```
>>> from scipy import linalg
>>> a = array([[1,3,5],
...            [2,5,1],
...            [2,3,6]])
# time consuming factorization
>>> lu, piv = linalg.lu_factor(a)

# fast solve for 1 or more
# right hand sides.
>>> b = array([10,8,3])
>>> linalg.lu_solve((lu, piv), b)
array([-7.82608696,  4.56521739,
        0.82608696])
```

EIGEN VALUES AND VECTORS

```
>>> from scipy import linalg
>>> a = array([[1,3,5],
...            [2,5,1],
...            [2,3,6]])
# compute eigen values/vectors
>>> vals, vecs = linalg.eig(a)
# print eigen values
>>> vals
array([ 9.39895873+0.j,
        -0.73379338+0.j,
         3.33483465+0.j])
# eigen vectors are in columns
# print first eigen vector
>>> vecs[:,0]
array([-0.57028326,
        -0.41979215,
        -0.70608183])
# norm of vector should be 1.0
>>> linalg.norm(vecs[:,0])
1.0
```

Matrix Objects

JOHN BRYCE
Leading in IT Education
a matrix company

STRING CONSTRUCTION

```
>>> from numpy import mat
>>> a = mat('[1,3,5;2,5,1;2,3,6]')
>>> a
matrix([[1, 3, 5],
        [2, 5, 1],
        [2, 3, 6]])
```

TRANSPOSE ATTRIBUTE

```
>>> a.T
matrix([[1, 2, 2],
        [3, 5, 3],
        [5, 1, 6]])
```

INVERTED ATTRIBUTE

```
>>> a.I
matrix([[-1.1739, 0.1304,  0.956],
        [ 0.4347, 0.1739, -0.391],
        [ 0.1739, -0.130,  0.0434]
       ])
```

DIAGONAL

```
>>> a.diagonal()
matrix([[1, 5, 6]])
>>> a.diagonal(-1)
matrix([[3, 1]])
```

SOLVE

```
>>> b = mat('10;8;3')
>>> a.I*b
matrix([[-7.82608696],
        [ 4.56521739],
        [ 0.82608696]])

>>> from scipy import linalg
>>> linalg.solve(a,b)
matrix([[-7.82608696],
        [ 4.56521739],
        [ 0.82608696]])
```


Integration

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.integrate --- General purpose Integration

- Ordinary Differential Equations (ODE)

  integrate.odeint, integrate.ode

- Samples of a 1-d function

  integrate.trapz (trapezoidal Method), integrate.simps (Simpson Method), integrate.romb (Romberg Method)

- Arbitrary callable function

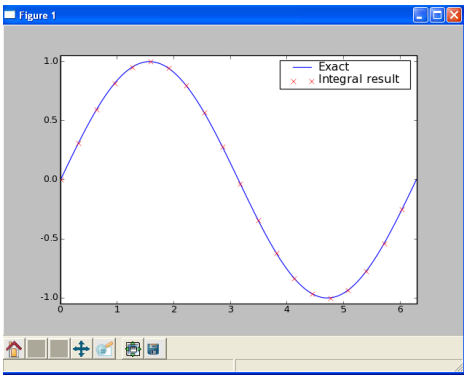
  integrate.quad (general purpose), integrate.dblquad (double integration), integrate.tplquad (triple integration), integrate.fixed_quad (fixed order Gaussian integration), integrate.quadrature (Gaussian quadrature to tolerance), integrate.romberg (Romberg)

Integration

JOHN BRYCE
Leading in IT Education
a matrix company

scipy.integrate --- Example

```
# Compare sin to integral(cos)
>>> def func(x):
    return integrate.quad(cos,0,x)[0]
>>> vecfunc = vectorize(func)

>>> x = r_[0:2*pi:100j]
>>> x2 = x[::5]
>>> y = sin(x)
>>> y2 = vecfunc(x2)
>>> plot(x,y,x2,y2,'rx')
>>> legend(['Exact',
...         'Integral Result'])
```



Special Functions

JOHN BRYCE
Leading in IT Education
a matrix company

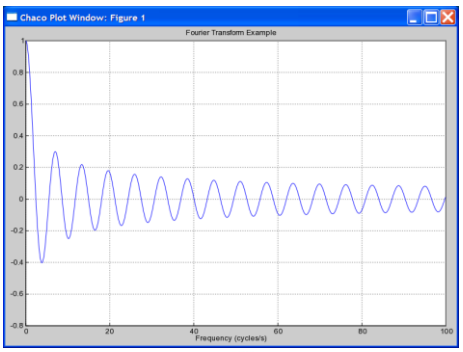
scipy.special

Includes over 200 functions:

Airy, Elliptic, Bessel, Gamma, HyperGeometric, Struve, Error, Orthogonal Polynomials, Parabolic Cylinder, Mathieu, Spheroidal Wave, Kelvin

FIRST ORDER BESSEL EXAMPLE

```
>>> from scipy import special
>>> x = r_[0:100:0.1]
>>> j0x = special.j0(x)
>>> plot(x,j0x)
```



JOHN BRYCE
Leading in IT Education
a matrix company

Pandas

Overview

JOHN BRYCE
Leading in IT Education
a matrix company

- Python Library to provide data analysis features similar to : R, MATLB, SAS
- Rich data structures and functions to make working with data structure fast, easy and expressive.
- It is built on top of NumPy which provides it agility
- Key components provided by Pandas : Two new data structures to Python
  - Series
  - DataFrame

Pandas is well suited for:

JOHN BRYCE
Leading in IT Education
a matrix company

- Tabular Data (SQL table & Excel spreadsheet)
- Ordered and Unordered Time Series Data
- Arbitrary Matrix Data
- Any other form of observational / statistical data sets

Series

JOHN BRYCE
Leading in IT Education
a matrix company

- One dimensional array like object like an array or list
- It contains array of data (of any NumPy data type) with associated indexes.
- By default , the series will get indexing from 0 to N where N = size -1

Series

JOHN BRYCE
Leading in IT Education
a matrix company

```
In [2]: import pandas as pd

In [3]: ls=['avi', 'dani', 'rina']

In [4]: s1 = pd.Series(ls)

In [5]: s1
Out[5]:
0     avi
1    dani
2    rina
dtype: object
```


Adding labels

JOHN BRYCE
Leading in IT Education
a matrix company

```
In [10]: ls2=['avi', 'dani', 'rina']

In [11]: ls1 = [1000, 2000, 3000]

In [12]: s2 = pd.Series(ls1,ls2)

In [13]: s2
Out[13]:
avi     1000
dani    2000
rina    3000
dtype: int64
```

Using Dictionary

JOHN BRYCE
Leading in IT Education
a matrix company

```
In [14]: d=dict(avi=2000, dani=3000, rina=4000)

In [15]: d
Out[15]: {'avi': 2000, 'dani': 3000, 'rina': 4000}

In [16]: pd.Series(d)
Out[16]:
avi     2000
dani    3000
rina    4000
dtype: int64
```


Operations

JOHN BRYCE
Leading in IT Education
a matrix company

```
In [36]: d1
Out[36]: {'avi': 2000, 'dani': 3000, 'rina': 4000}

In [37]: d2
Out[37]: {'avi': 4000, 'dani': 2000, 'dina': 2000, 'rina': 5000}

In [38]: s1=pd.Series(d1)

In [39]: s2=pd.Series(d2)
```

```
In [43]: s1['avi']
Out[43]: 2000

In [44]: s1
Out[44]:
avi     2000
dani    3000
rina    4000
dtype: int64

In [45]: s2
Out[45]:
avi     4000
dani    2000
dina    2000
rina    5000
dtype: int64

In [46]: s1+s2
Out[46]:
avi     6000.0
dani    5000.0
dina       NaN
rina    9000.0
dtype: float64
```

Handling Nulls

JOHN BRYCE
Leading in IT Education
a matrix company

```
In [71]: s3 = s1+s2

In [72]: s3
Out[72]:
avi     6000.0
dani    5000.0
dina       NaN
rina    9000.0
dtype: float64

In [73]: s3.isnull()
Out[73]:
avi     False
dani    False
dina     True
rina    False
dtype: bool

In [74]: s3.notnull()
Out[74]:
avi      True
dani     True
dina    False
rina     True
dtype: bool
```

```
In [75]: s4=s3[s3.notnull()]

In [76]: s4
Out[76]:
avi     6000.0
dani    5000.0
rina    9000.0
dtype: float64
```


Dataframe

JOHN BRYCE
Leading in IT Education
a matrix company

- A dataFrame is a tabular data structure comprised of rows and columns, akin to a spreadsheet or database table.
- It can be treated as a series of objects sharing common index

Operations

JOHN BRYCE
Leading in IT Education
a matrix company

- Filtering
- Summarizing
- Group by – split apply combine
- Merge, join, aggregate
- Time series/ Data functionality
- Plotting with Matplotlib and many more…

DataFrame From NumPy Array

JOHN BRYCE
Leading in IT Education
a matrix company

```
import pandas as pd
import numpy as np
```

```
samp=np.random.randint(100, 600,size=(4,5))
```

```
samp
```

```
array([[205, 225, 129, 549, 328],
       [150, 348, 325, 474, 268],
       [495, 348, 488, 579, 371],
       [407, 158, 478, 120, 575]])
```

```
df=pd.DataFrame(samp,index=['avi','dani','rina','dina'],
                     columns=['Jan','Feb','Mar','Apr','May'])
```

	Jan	Feb	Mar	Apr	May
avi	205	225	129	549	328
dani	150	348	325	474	268
rina	495	348	488	579	371
dina	407	158	478	120	575

Selecting and Indexing

JOHN BRYCE
Leading in IT Education
a matrix company

```
df['Jan']
```

```
avi     205
dani    150
rina    495
dina    407
Name: Jan, dtype: int64
```

```
df['Jan']['avi']
```

```
205
```

```
# Pass a list of column names
df[['Jan','May']]
```

	Jan	May
avi	205	328
dani	150	268
rina	495	371
dina	407	575

Using SQL Syntax

JOHN BRYCE
Leading in IT Education
a matrix company

```
df.Feb
```

```
avi     558
dani    336
rina    168
dina    571
Name: Feb, dtype: int64
```

```
df.Feb.avi
```

```
558
```

Not Recommended

```
df.Feb[df.Feb>500]
```

```
avi     558
dina    571
Name: Feb, dtype: int64
```

Types

JOHN BRYCE
Leading in IT Education
a matrix company

- Its important to understand the types and the permitted operations on that types

```
type(df['May'])
```

```
pandas.core.series.Series
```

```
type(df)
```

```
pandas.core.frame.DataFrame
```

```
type(df['May']>300)
```

```
pandas.core.series.Series
```

```
type(df['May']['avi'])
```

```
numpy.int64
```


Unique Data

JOHN BRYCE
Leading in IT Education
a matrix company

```
df['Jan']['rina'] = 300
df['Jan']['avi'] = 300
```

```
df
```

	Jan	Feb	Mar	Apr	May
avi	300	455	367	207	440
dani	276	169	213	561	510
rina	300	423	533	411	314
dina	399	257	592	356	215

```
df['Jan'].unique()
```

```
array([300, 276, 399])
```

```
df['Jan'].nunique()
```

```
3
```

```
df['Jan'].value_counts()
```

```
300    2
276    1
399    1
Name: Jan, dtype: int64
```

Update Data

JOHN BRYCE
Leading in IT Education
a matrix company

```
def add_bonus(x):
    return x*1.2
df.apply(add_bonus)
```

	Jan	Feb	Mar	Apr	May
avi	360.0	546.0	440.4	248.4	528.0
dani	331.2	202.8	255.6	673.2	612.0
rina	360.0	507.6	639.6	493.2	376.8
dina	478.8	308.4	710.4	427.2	258.0

```
df['Jan'].apply(lambda x:x+1000)
```

```
avi     1300
dani    1276
rina    1300
dina    1399
Name: Jan, dtype: int64
```

```
df['Jan'].sum()
```

```
1275L
```


Adding New Data

JOHN BRYCE
Leading in IT Education
a matrix company

Column

```
df['Expected'] = df['May'] + 20
```

```
df
```

	Jan	Feb	Mar	Apr	May	Expected
avi	205	225	129	549	328	348
dani	150	348	325	474	268	288
rina	495	348	488	579	371	391
dina	407	158	478	120	575	595

Row

```
df.loc['eli'] = [100,200,300,400,500,600]
```

```
df
```

	Jan	Feb	Mar	Apr	May	Expected
avi	500	558	140	589	560	580
dani	177	336	405	267	277	297
rina	506	168	269	161	217	237
dina	237	571	138	598	273	293
eli	100	200	300	400	500	600

Removing Data

JOHN BRYCE
Leading in IT Education
a matrix company

```
df.drop('rina',axis=0)
```

	Jan	Feb	Mar	Apr	May	Expected
avi	205	225	129	549	328	348
dani	150	348	325	474	268	288
dina	407	158	478	120	575	595

```
# Not inplace unless specified!
df
```

	Jan	Feb	Mar	Apr	May	Expected
avi	205	225	129	549	328	348
dani	150	348	325	474	268	288
rina	495	348	488	579	371	391
dina	407	158	478	120	575	595

```
df.drop('Apr',axis=1,inplace=True)
```

```
df
```

	Jan	Feb	Mar	May	Expected
avi	205	225	129	328	348
dani	150	348	325	268	288
rina	495	348	488	371	391
dina	407	158	478	575	595

Selecting Data

JOHN BRYCE
Leading in IT Education
a matrix company

Using label

```
df.loc['avi']
```

```
Jan          205
Feb          225
Mar          129
May          328
Expected     348
Name: avi, dtype: int64
```

Using position

```
df.iloc[2]
```

```
Jan          495
Feb          348
Mar          488
May          371
Expected     391
Name: rina, dtype: int64
```

Selecting subset

```
df.loc['avi','Jan']
```

```
205
```

```
df.loc[['avi','rina'],['Feb','Apr']]
```

	Feb	Apr
avi	225.0	NaN
rina	348.0	NaN

More Selections

JOHN BRYCE
Leading in IT Education
a matrix company

```
df
```

	Jan	Feb	Mar	Apr	May
avi	360	133	185	320	411
dani	132	211	228	226	422
rina	247	336	107	470	515
dina	210	491	156	282	574

```
df['Jan'][1:3]
```

```
dani    132
rina    247
Name: Jan, dtype: int64
```

```
df.loc['dani':,'Feb':'Apr']
```

	Feb	Mar	Apr
dani	211	228	226
rina	336	107	470
dina	491	156	282

```
df
```

	Jan	Feb	Mar	Apr	May
avi	360	133	185	320	411
dani	132	211	228	226	422
rina	247	336	107	470	515
dina	210	491	156	282	574

```
df.iloc[:3]
```

	Jan	Feb	Mar	Apr	May
avi	360	133	185	320	411
dani	132	211	228	226	422
rina	247	336	107	470	515

```
df.iloc[[1,3],[2,3]]
```

	Mar	Apr
dani	228	226
dina	156	282

Conditional Selection

JOHN BRYCE
Leading in IT Education
a matrix company

```
df>200
```

	Jan	Feb	Mar	May	Expected
avi	True	True	False	True	True
dani	False	True	True	True	True
rina	True	True	True	True	True
dina	True	False	True	True	True

Boolean operation – returns new DF

```
df[df>200]
```

	Jan	Feb	Mar	May	Expected
avi	205.0	225.0	NaN	328	348
dani	NaN	348.0	325.0	268	288
rina	495.0	348.0	488.0	371	391
dina	407.0	NaN	478.0	575	595

Selecting based on another DF

```
df[df['Jan']>200]
```

	Jan	Feb	Mar	May	Expected
avi	205	225	129	328	348
rina	495	348	488	371	391
dina	407	158	478	575	595

Selecting based on a series

More Conditions

JOHN BRYCE
Leading in IT Education
a matrix company

```
df[df['Jan']>200]['May']
```

```
avi     328
rina    371
dina    575
Name: May, dtype: int64
```

```
df[df['Jan']>200][['Jan','Mar']]
```

	Jan	Mar
avi	205	129
rina	495	488
dina	407	478

For two conditions you can use | and & with parenthesis:

```
df[(df['Jan']>200) & (df['Feb'] > 300)]
```

	Jan	Feb	Mar	May	Expected
rina	495	348	488	371	391

Using Callback

JOHN BRYCE
Leading in IT Education
a matrix company

```
df.loc[lambda df:df['Jan']>200,:]
```

	Jan	Feb	Mar	Apr	May
avi	360	133	185	320	411
rina	247	336	107	470	515
dina	210	491	156	282	574

```
df.loc[:,lambda df: ['Jan','Mar']]
```

	Jan	Mar
avi	360	185
dani	132	228
rina	247	107
dina	210	156

Sorting

JOHN BRYCE
Leading in IT Education
a matrix company

```
df
```

	Jan	Feb	Mar	Apr	May
avi	300	455	367	207	440
dani	276	169	213	561	510
rina	300	423	533	411	314
dina	399	257	592	356	215

```
df.sort_values('Jan')
```

	Jan	Feb	Mar	Apr	May
dani	276	169	213	561	510
avi	300	455	367	207	440
rina	300	423	533	411	314
dina	399	257	592	356	215

Use inplace=True to apply sorting

NaN values

JOHN BRYCE
Leading in IT Education
a matrix company

```
df['Jan']['avi'] = np.nan
df['Feb']['dani'] = np.nan
df
```

	Jan	Feb	Mar	Apr	May
avi	NaN	455.0	367	207	440
dani	276.0	NaN	213	561	510
rina	300.0	423.0	533	411	314
dina	399.0	257.0	592	356	215

```
df.isnull()
```

	Jan	Feb	Mar	Apr	May
avi	True	False	False	False	False
dani	False	True	False	False	False
rina	False	False	False	False	False
dina	False	False	False	False	False

```
df.dropna()
```

	Jan	Feb	Mar	Apr	May
rina	300.0	423.0	533	411	314
dina	399.0	257.0	592	356	215

```
df.fillna(1000)
```

	Jan	Feb	Mar	Apr	May
avi	1000.0	455.0	367	207	440
dani	276.0	1000.0	213	561	510
rina	300.0	423.0	533	411	314
dina	399.0	257.0	592	356	215

Multi-index

JOHN BRYCE
Leading in IT Education
a matrix company

```
# Index Levels
years = ['2016','2016','2016','2016','2017','2017','2017','2017']
q = [1,2,3,4,1,2,3,4]
t = list(zip(years,q))
mi = pd.MultiIndex.from_tuples(t)
```

```
mi
```

```
MultiIndex(levels=[[u'2016', u'2017'], [1, 2, 3, 4]],
           labels=[[0, 0, 0, 0, 1, 1, 1, 1], [0, 1, 2, 3, 0, 1, 2, 3]])
```

```
df = pd.DataFrame(np.random.randn(8,2),index=mi,columns=['A','B'])
df
```

		A	B
2016	1	-0.889180	0.311152
	2	-0.612847	-0.353895
	3	-0.866984	0.711970
	4	-0.057056	-0.564472
2017	1	-1.537100	0.851859
	2	0.725848	-0.918994
	3	1.189998	0.580911
	4	-1.893752	-0.996923

Multi-index

JOHN BRYCE
Leading in IT Education
a matrix company

```
df.loc['2016']
```

	A	B
1	-0.889180	0.311152
2	-0.612847	-0.353895
3	-0.866984	0.711970
4	-0.057056	-0.564472

```
df.loc['2017'].loc[1]
```

```
A   -1.537100
B    0.851859
Name: 1, dtype: float64
```

```
df.index.names = ['Year','Q']
```

```
df
```

		A	B
Year	Q		
2016	1	-0.889180	0.311152
	2	-0.612847	-0.353895
	3	-0.866984	0.711970
	4	-0.057056	-0.564472
2017	1	-1.537100	0.851859
	2	0.725848	-0.918994
	3	1.189998	0.580911
	4	-1.893752	-0.996923

Multi-index

JOHN BRYCE
Leading in IT Education
a matrix company

```
df.xs('2017')
```

Q	A	B
1	-1.462987	0.345033
2	0.577788	-0.654170
3	-1.341800	-0.188705
4	0.477607	0.096507

```
df.xs(['2016',1])
```

```
A    1.266636
B   -0.576670
Name: (2016, 1), dtype: float64
```

```
df.xs(1,level='Q')
```

Year	A	B
2016	1.266636	-0.576670
2017	-1.462987	0.345033

Concatenation

JOHN BRYCE
Leading in IT Education
a matrix company

```
df2
```

	Jan	Feb	Mar	Apr	May
eli	213	337	252	273	352
yosi	140	180	513	547	439
rani	127	191	215	547	207
meir	343	376	269	408	179

```
df
```

	Jan	Feb	Mar	Apr	May
avi	213	337	252	273	352
dani	140	180	513	547	439
rina	127	191	215	547	207
dina	343	376	269	408	179

```
pd.concat([df,df2])
```

	Jan	Feb	Mar	Apr	May
avi	213	337	252	273	352
dani	140	180	513	547	439
rina	127	191	215	547	207
dina	343	376	269	408	179
eli	213	337	252	273	352
yosi	140	180	513	547	439
rani	127	191	215	547	207
meir	343	376	269	408	179

```
pd.concat([df,df2],axis=1)
```

	Jan	Feb	Mar	Apr	May	Jan	Feb	Mar	Apr	May
avi	213.0	337.0	252.0	273.0	352.0	NaN	NaN	NaN	NaN	NaN
dani	140.0	180.0	513.0	547.0	439.0	NaN	NaN	NaN	NaN	NaN
dina	343.0	376.0	269.0	408.0	179.0	NaN	NaN	NaN	NaN	NaN
eli	NaN	NaN	NaN	NaN	NaN	213.0	337.0	252.0	273.0	352.0
meir	NaN	NaN	NaN	NaN	NaN	343.0	376.0	269.0	408.0	179.0
rani	NaN	NaN	NaN	NaN	NaN	127.0	191.0	215.0	547.0	207.0
rina	127.0	191.0	215.0	547.0	207.0	NaN	NaN	NaN	NaN	NaN
yosi	NaN	NaN	NaN	NaN	NaN	140.0	180.0	513.0	547.0	439.0

Merging Data

JOHN BRYCE
Leading in IT Education
a matrix company

```
left = pd.DataFrame({'id': [10, 20, 30, 40],
                     'Name': ['avi', 'dani', 'rina', 'dina'],
                     'City': ['haifa', 'aco', 'tel aviv', 'yafo']})

right = pd.DataFrame({'id': [10, 20, 30, 50],
                      'Salary': [1000, 2000, 3000, 4000],
                      'Position': ['CTO', 'CEO', 'COO', 'Marketing']})
```

```
1 left
```

	City	Name	id
0	haifa	avi	10
1	aco	dani	20
2	tel aviv	rina	30
3	yafo	dina	40

```
right
```

	Position	Salary	id
0	CTO	1000	10
1	CEO	2000	20
2	COO	3000	30
3	Marketing	4000	50

```
pd.merge(left,right,how='inner',on='id')
```

	City	Name	id	Position	Salary
0	haifa	avi	10	CTO	1000
1	aco	dani	20	CEO	2000
2	tel aviv	rina	30	COO	3000

```
pd.merge(left,right,how='left',on='id')
```

	City	Name	id	Position	Salary
0	haifa	avi	10	CTO	1000.0
1	aco	dani	20	CEO	2000.0
2	tel aviv	rina	30	COO	3000.0
3	yafo	dina	40	NaN	NaN

```
pd.merge(left,right,how='right',on='id')
```

	City	Name	id	Position	Salary
0	haifa	avi	10	CTO	1000
1	aco	dani	20	CEO	2000
2	tel aviv	rina	30	COO	3000
3	NaN	NaN	50	Marketing	4000

Joining

JOHN BRYCE
Leading in IT Education
a matrix company

```
left = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3'],
                     'key': ['K0', 'K1', 'K0', 'K1']})


right = pd.DataFrame({'C': ['C0', 'C1'],
                      'D': ['D0', 'D1']},
                     index=['K0', 'K1'])


result = left.join(right, on='key')
```

Grouping

JOHN BRYCE
Leading in IT Education
a matrix company

```
df3 = pd.DataFrame({'id': [10, 20, 30, 40],
                    'Name': ['avi', 'dani', 'rina', 'dina'],
                    'City': ['haifa', 'haifa', 'tel aviv', 'yafo']})
```

```
df3.groupby('City').count()
```

City	Name	id
haifa	2	2
tel aviv	1	1
yafo	1	1

Also:

- **min()**
- **max()**
- **std()**
- **describe()**
- **…**

Working With Files

JOHN BRYCE
Leading in IT Education
a matrix company

- pd.read_csv('cust.csv')
- df.to_csv(cust.csv',index=False)
- pd.read_excel('samp.xslx')
- df.to_excel('samp.xlsx',sheet_name='cust')
- Also supported:
  - HTML
  - JSON
  - …

Visualization

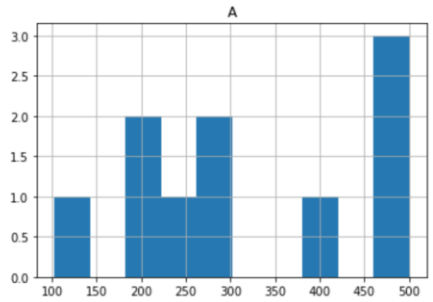
JOHN BRYCE
Leading in IT Education
a matrix company

```
%matplotlib inline
```

```
dh=pd.DataFrame(samp,columns="A,B,C,D,E,F,G,H,I,J".split(','))
```

```
dh.hist('A')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x119916b50>]], dtype=object)
```

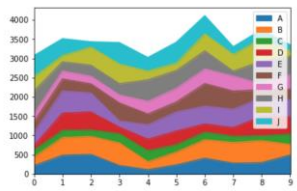


More Graphs

JOHN BRYCE
Leading in IT Education
a matrix company

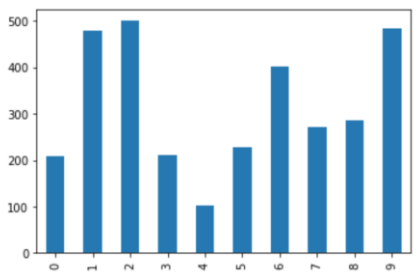
```
dh.plot.area()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1199d2a10>
```




```
dh['A'].plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a717950>
```



SQL Servers

JOHN BRYCE
Leading in IT Education
a matrix company

- The pandas.io.sql module provides a collection of query wrappers to both facilitate data retrieval and to reduce dependency on DB-specific API.
- Database abstraction is provided by SQLAlchemy if installed.
- In addition you will need a driver library for your database.
- Examples of such drivers are
  - psycopg2 for PostgreSQL
  - pymysql for MySQL.
  - SQLite - included in Python's standard library by default.

SQL - Function

JOHN BRYCE
Leading in IT Education
a matrix company

- read_sql_table(table_name, con[, schema, ...])
  - Read SQL database table into a DataFrame.
- read_sql_query(sql, con[, index_col, ...])
  - Read SQL query into a DataFrame.
- read_sql(sql, con[, index_col, ...])
  - Read SQL query or database table into a DataFrame.
- DataFrame.to_sql(name, con[, flavor, ...])
  - Write records stored in a DataFrame to a SQL database

Example - Teradata

JOHN BRYCE
Leading in IT Education
a matrix company

- Install python module:
  - # pip install teradata
- Connect using:
  - REST API (teradata.tdrest)
  - ODBC (teradata.tdodbc)

```
import teradata

udaExec = teradata.UdaExec (appName="HelloWorld", version="1.0",
        logConsole=False)

session = udaExec.connect(method="odbc", system="tdprod",
        username="xxx", password="xxx");

for row in session.execute("SELECT GetQueryBand()"):
    print(row)
```


Cursors

JOHN BRYCE
Leading in IT Education
a matrix company

```
import teradata
udaExec = teradata.UdaExec()
with udaExec.connect("${dataSourceName}") as session:
for row in session.execute("""SELECT InfoKey AS name, InfoData as val
          FROM DBC.DBCInfo"""):
        print(row[0] + ": " + row[1])
        print(row["name"] + ": " + row["val"])
        print(row.name + ": " + row.val)
```

```
import teradata
udaExec = teradata.UdaExec()
with udaExec.connect("${dataSourceName}") as session:
    with session.cursor() as cursor:
    for row in cursor.execute("SELECT * from ${tableName}"):
            session.execute("DELETE FROM ${tableName} WHERE id = ?", (row.id, )):
```

Stored Procedures

JOHN BRYCE
Leading in IT Education
a matrix company

- invoked using the "callproc" method
- OUT parameters should be specified as teradata.OutParam instances
- Example:

```
results = session.callproc("MyProcedure",
        (teradata.InOutParam("inputValue",
"inoutVar1"),
          teradata.OutParam(),
           teradata.OutParam("outVar2",
dataType="PERIOD")))
print(results.inoutVar1)
print(results.outVar1)
```


Transactions

JOHN BRYCE
Leading in IT Education
a matrix company

```
import teradata
udaExec = teradata.UdaExec()
with udaExec.connect("${dataSourceName}", autoCommit=False) as session:
    session.execute("CREATE TABLE ${tableName} (${columns})")
    session.commit()
```

Seaborn

Seaborn

- Statistical plotting library
- Great styles
- Works great with NumPy arrays and Pandas Dataframes

Seaborn

JOHN BRYCE
Leading in IT Education
a matrix company

- Some built in datasets

```
import seaborn as sns
%matplotlib inline
```

```
tips = sns.load_dataset('tips')
```

```
tips.head(10)
```

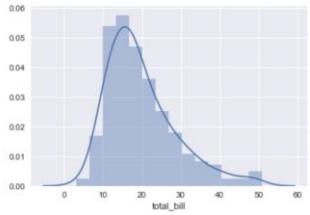
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

Distribution Plot

JOHN BRYCE
Leading in IT Education
a matrix company

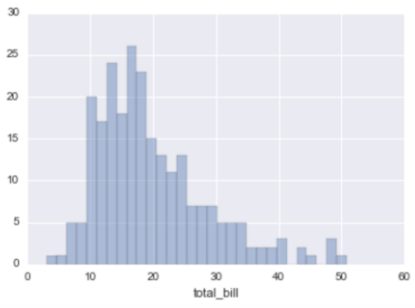
```
sns.distplot(tips['total_bill'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11982a7d0>
```




```
sns.distplot(tips['total_bill'],kde=False,bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c7b8668>
```



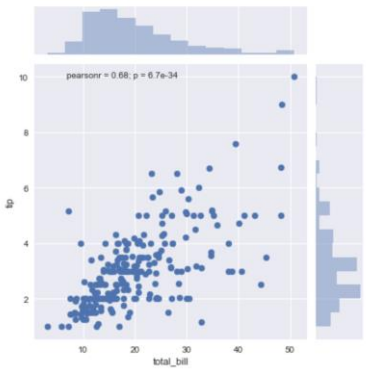
JointPlot

JOHN BRYCE
Leading in IT Education
a matrix company

- jointplot() allows you to basically match up two distplots() for bivariate data. With your choice of what **kind** parameter to compare with:
  - scatter
  - reg
  - resid
  - kde
  - hex

```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='scatter')
```

```
<seaborn.axisgrid.JointGrid at 0x11d05f250>
```



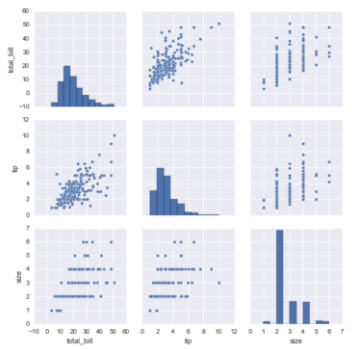
Pairplot

JOHN BRYCE
Leading in IT Education
a matrix company

- pairplot will plot pairwise relationships across an entire dataframe (for the numerical columns) and supports a color hue argument (for categorical columns)

```
sns.pairplot(tips)
```

```
<seaborn.axisgrid.PairGrid at 0x11e844208>
```



Categorical Data Plots

JOHN BRYCE
Leading in IT Education
a matrix company

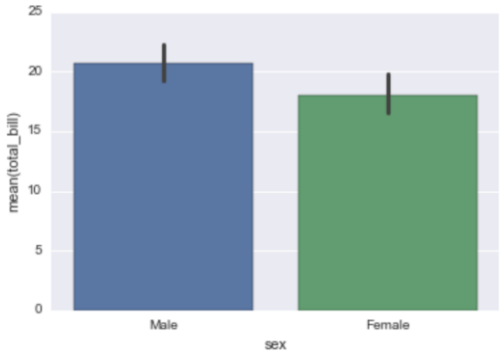
- Factorplot
- boxplot
- violinplot
- stripplot
- swarmplot
- barplot
- countplot

Barplot

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.barplot(x='sex',y='total_bill',data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c99b8d0>
```

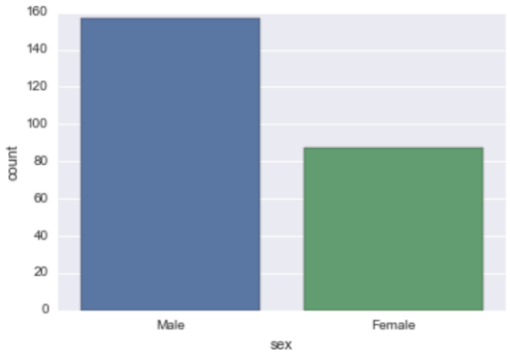


countplot

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.countplot(x='sex',data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1153276d8>
```

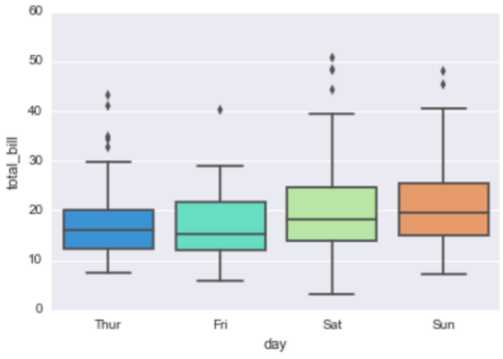


Boxplot

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11db81630>
```

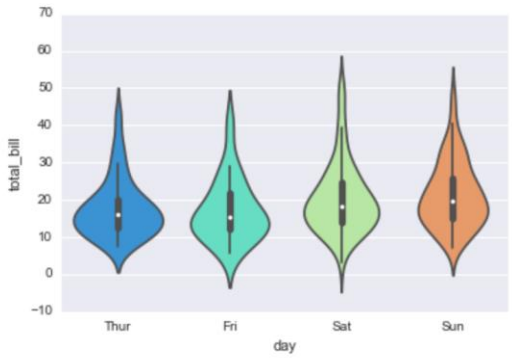


voilinplot

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.violinplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e682ba8>
```

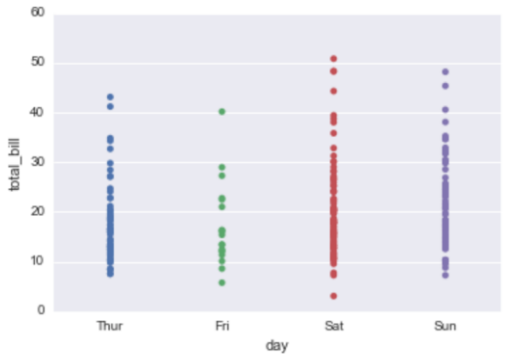


stripplot

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.stripplot(x="day", y="total_bill", data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x120272278>
```

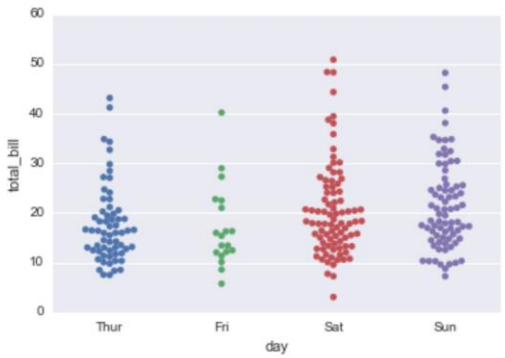


Swarmplot

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.swarmplot(x="day", y="total_bill", data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x120c463c8>
```

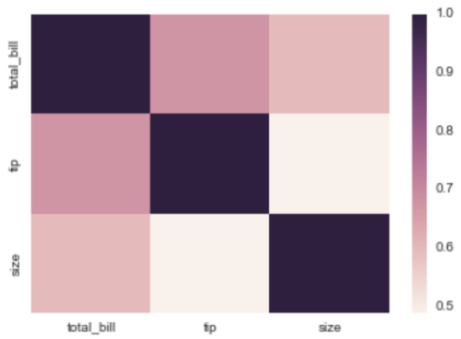


Matrix Plots

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.heatmap(tips.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c31d470>
```

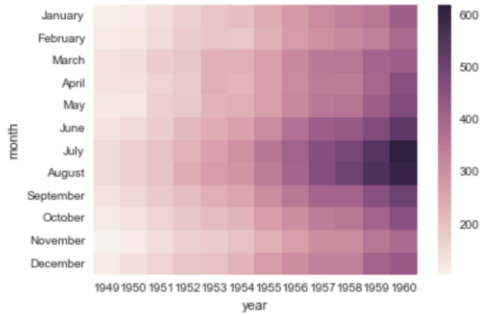


heatmap

JOHN BRYCE
Leading in IT Education
a matrix company

```
pvflights = flights.pivot_table(values='passengers',index='month',columns='year')
sns.heatmap(pvflights)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11cd09320>
```

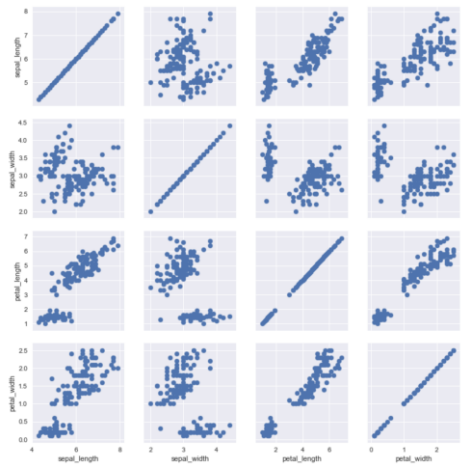


Grids

JOHN BRYCE
Leading in IT Education
a matrix company

```
g = sns.PairGrid(iris)
g.map(plt.scatter)
```

```
<seaborn.axisgrid.PairGrid at 0x10e227a10>
```

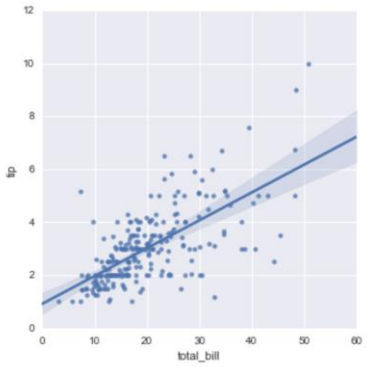


Regression Plots

JOHN BRYCE
Leading in IT Education
a matrix company

```
sns.lmplot(x='total_bill',y='tip',data=tips)
```

```
<seaborn.axisgrid.FacetGrid at 0x117c81048>
```



JOHN BRYCE
Leading in IT Education
a matrix company

Machine Learning

Overview

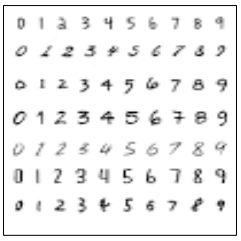
JOHN BRYCE
Leading in IT Education
a matrix company

- Machine Learning is the science of programming computers so they can *learn from data*
- *"field of study that gives computers the ability to learn without being explicitly programmed" (*Arthur Samuel, 1959)
- Example: spam filter - Machine Learning program that can learn to flag spam given examples of spam emails
- **Machine learning is a comprehensive approach to solving problems**. Its not a list of algorithms

Why Learn?

JOHN BRYCE
Leading in IT Education
a matrix company

- Learn it when you can't code it
  - Complex tasks where deterministic solution don't suffice
  - e.g. speech recognition, handwriting recognition
- Learn it when you can't scale it
  - Repetitive task needing human-like expertise (e.g. recommendations, spam & fraud detection)
  - Speed, scale of data, number of data points
- Learn it when you need to adapt/personalize
  - e.g., personalized product recommendations, stock predictions



Why use Machine Learning

JOHN BRYCE
Leading in IT Education
a matrix company

- Programming all rules can be very complex. Instead let the system build it rules
- Example: a spam filter based on Machine Learning techniques automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples compared to the ham examples. The program is much shorter, easier to maintain and most likely more accurate.

Why using ML

JOHN BRYCE
Leading in IT Education
a matrix company

- If spammers notice that all their emails containing the word "4U" are blocked, they might start writing "For U" instead.
- A spam filter using traditional programming techniques would need to be updated to flag "For U" emails.
- If spammers keep working around your spam filter, you will need to keep writing new rules forever.
- In contrast, a spam filter based on Machine Learning techniques automatically notices that "For U" has become unusually frequent in spam flagged by users, and it starts flagging them without your intervention

Applications

JOHN BRYCE
Leading in IT Education
a matrix company

- Fraud detection.
- Web search results.
- Real-time ads on web pages
- Credit scoring and next-best offers.
- Prediction of equipment failures.
- New pricing models.
- Network intrusion detection.
- Recommendation Engines
- Customer Segmentation
- Text Sentiment Analysis
- Predicting Customer Churn
- Pattern and image recognition.
- Email spam filtering.
- Financial Modeling

Terminology

JOHN BRYCE
Leading in IT Education
a matrix company

- **Model** - a set of patterns learned from data.
- **Algorithm** - a specific ML process used to train a model.
- **Training data** - the dataset from which the **algorithm** learns the **model**.
- **Test data** - a new dataset for reliably evaluating model performance.
- **Features** - Variables (columns) in the dataset used to train the model.
- **Target variable** - A specific variable you're trying to predict.
- **Observations** - Data points (rows/records) in the dataset

ML Task

JOHN BRYCE
Leading in IT Education
a matrix company

- A **task** is a specific objective for your algorithms.
  - Predict age.. (linear regression)
  - Predict male/female (logistic reg)
  - Analyze data (clustering)
- You can use many algorithms for a task
  - Try multiple algorithms

Machine Learning

JOHN BRYCE
Leading in IT Education
a matrix company

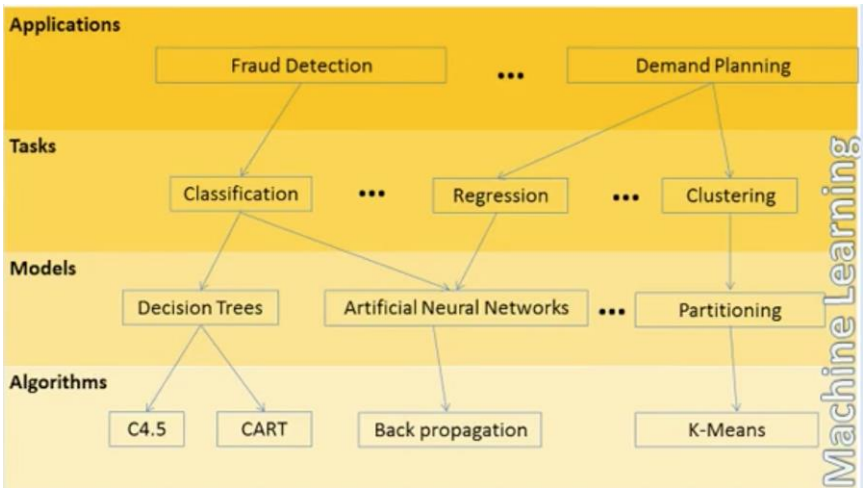
- Supervised Learning - predicting using labeled data
  - Classification, Regression
- Unsupervised Learning - group unlabeled data using similarities
  - Clustering, Density Estimation, Dimensionality Reduction
- Semi-supervised Learning – small amount of labeled data with large amount of unlabeled data
- Active Learning – Semi-supervised learning with access to a human labeler during training
- Reinforcement Learning
  - Perform an action from experience
  - Feedback received after a sequence of actions/predictions
  - iRobot

More Types

JOHN BRYCE
Leading in IT Education
a matrix company

- Online Learning
  - Learn in real time
- Batch Learning
  - Learn from datasets
- Instance based learning
  - Learning is based on instances
  - New data -> try to find a similar instance
  - For example: email spam/ham
- model based learning
  - Learning is based on model – function for example

JOHN BRYCE
Leading in IT Education
a matrix company



Types of Supervised Learning

JOHN BRYCE
Leading in IT Education
a matrix company

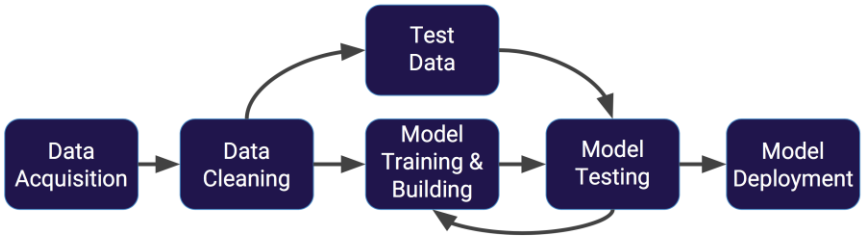
- Classification: *Y* is categorical
  - Examples:
    - Web page classification as e-Commerce/non e-Commerce (Binary)
    - Product classification into categories (Multi-class)
  - Model F: Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, Naïve Bayes, etc.
- Regression: *Y* is numeric (ordinal/real-valued)
  - Examples:
    - Base price markup prediction for a product
    - Forecasting demand for a product
  - Model F: Linear Regression, Regression Trees, Kernel Regression, etc.

Steps

JOHN BRYCE
Leading in IT Education
a matrix company

- **Exploratory Analysis -** "get to know" the data.
- **Data Cleaning –** clean, restructure , scale, transform
- **Feature Engineering –** find new features
- **Algorithm Selection –** find the best algorithms
- **Model Training –** use the algorithms to train the model

Machine Learning Process



Exploratory Analysis

- Ask questions about the data:
  - How many records
  - How many features
  - What are the data types of my features?
    Are they numeric? Categorical?
  - Do I have a target variable?
- Look at the data (samples)
  - Do the columns make sense?
  - Do the values in those columns make sense?
  - Are the values on the right scale?
  - Is **missing data** going to be a big problem

Machine Learning Problem Definition

JOHN BRYCE
Leading in IT Education
a matrix company

- Key elements of Prediction Problem
  - Target variable to be predicted
  - Training examples
  - Features in each example (Categorical, Numeric, Text)
- Example: Income classification problem
  - Predict if a person makes more than $50K

Age	Education	Years of education	Marital status	Occupation	Sex	Label
39	Bachelors	16	Single	Adm-clerical	Male	<50K (-1)
31	Masters	18	Married	Engineering	Female	>=50K (+1)

Types of Features

JOHN BRYCE
Leading in IT Education
a matrix company

Age	Education	Years of education	Marital status	Occupation	Sex	Label
39	Bachelors	16	Single	Adm-clerical	Male	<50K (-1)
31	Masters	18	Married	Engineering	Female	>=50K (+1)

- Categorical/Nominal – Occupation, Marital Status, Prime Subscriber
- Numeric – Age, Orders in the last month, Total spend in the last year
  - Quantity (Integer or Real): Price, Votes
  - Interval: Dates, Temperature
  - Ratio: Quarterly growth
- Ordinal – Education level, Star rating for a product

Types of Data

JOHN BRYCE
Leading in IT Education
a matrix company

- Matrix Data – A design matrix **X** and label vector **y**
- Text – Customer reviews, product descriptions
- Images – Product images, Maps
- Set Data – Items purchased together
- Sequence Data – Clickstream, Purchase history
- Time Series – Audio/Video, Stock prices
- Graph/Network – Social Networks, WWW

Data Visualization & Analysis

JOHN BRYCE
Leading in IT Education
a matrix company

- Better understanding of data -> Better feature engineering & modeling

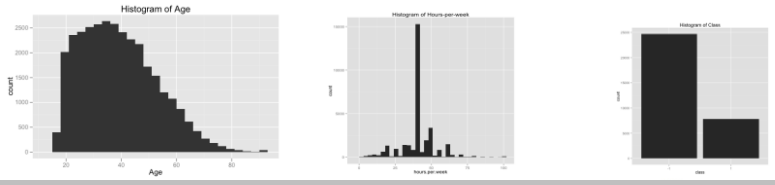
Types of visualization & analysis

- Feature and target summaries
  - Feature and target data distribution, histograms
  - Identify outliers in data, detect skew in feature/class distribution
- Feature-Target correlation
  - Correlation measures like mutual information, Pearson's correlation coefficient
  - Class distribution conditioned on feature values, scatter plots
  - Identify features with predictive power, target leakers

Distributions of numeric features

JOHN BRYCE
Leading in IT Education
a matrix company

- A quick and dirty grid of **histograms** is enough to understand the distributions
- Things to look out for:
  - Distributions that are unexpected
  - Potential outliers that don't make sense
  - Features that should be binary (i.e. "wannabe indicator variables")
  - Boundaries that don't make sense
  - Potential measurement errors



Distributions of categorical features

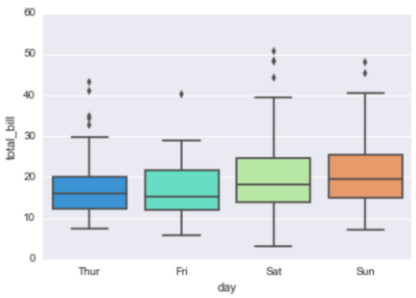
JOHN BRYCE
Leading in IT Education
a matrix company

- Categorical features cannot be visualized through histograms. Instead, you can use **bar plots** (count)
- Look out for **sparse classes**, which are classes that have a very small number of observations
- They tend to be problematic when building models.
  - In the best case, they don't influence the model much.
  - In the worse case, they can cause the model to be **overfit**.

Segmentations

JOHN BRYCE
Leading in IT Education
a matrix company

- Segmentations are powerful ways to observe the *relationship between categorical features and numeric features*
- Use **Box Plot**
  - Find median, min, max and any numeric relationship



Correlations

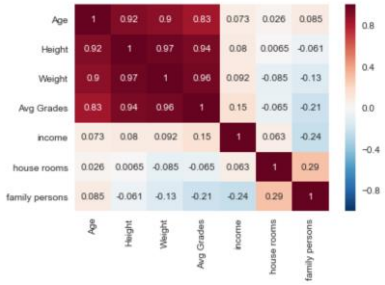
JOHN BRYCE
Leading in IT Education
a matrix company

- Correlations allow you to look at the *relationships between numeric features and other numeric features*
- Value between -1 and 1 that represents how closely two features move in unison
  - **Positive** correlation means that as one feature increases, the other increases. E.g. a child's age and her height.
  - **Negative** correlation means that as one feature increases, the other decreases. E.g. hours spent studying and number of parties attended
  - Correlations near -1 or 1 indicate a **strong relationship**.
  - Those closer to 0 indicate a **weak relationship**.
  - 0 indicates **no relationship**

Correlation

JOHN BRYCE
Leading in IT Education
a matrix company

- Use **Heat map**
- you should look out for:
  - Which features are strongly correlated with the **target variable**?
  - Are there interesting or **unexpected** strong correlations between other features



Data Preparation

JOHN BRYCE
Leading in IT Education
a matrix company

- Transform data to appropriate input format
  - CSV format, headers specifying column names and data types
  - Filter XML/HTML from text
- Split data into train and test files
  - Training data used to learn models
  - Test data used to evaluate model performance
- Randomly shuffle data
  - Speeds convergence of online training algorithms
- Feature scaling (for numeric attributes)
  - Subtract mean and divide by standard deviation -> zero mean, unit variance
  - Speeds convergence of gradient-based training algorithms

Data Cleaning

JOHN BRYCE
Leading in IT Education
a matrix company

- Missing feature values, outliers can hurt model performance
- Strategies for handling missing values, outliers
  - Introduce new indicator variable to represent missing value
  - Replace missing numeric values with mean, categorical values with mode

Age	Education	Years of education	Marital status	Occupation	Sex	Label
39	Bachelors	16	Single	Adm-clerical	Male	0
31	Masters	18	Married	Engineer	Female	1
44	Bachelors			Accounting	Male	0
	Bachelors	14	Married	Engineer	Female	0
150						

Outlier

Missing values

Data Cleaning

JOHN BRYCE
Leading in IT Education
a matrix company

- Remove unwanted observations from your dataset.
  - This includes **duplicate** or **irrelevant** observations
- Fix structural errors.
  - Structural errors are those that arise during measurement, data transfer, or other types of **"poor housekeeping."**
  - For instance, you can check for **typos** or **inconsistent capitalization.** This is mostly a concern for categorical features, and you can look at your bar plots to check
    - NY
    - New York
    - New-York

Dealing with missing values

JOHN BRYCE
Leading in IT Education
a matrix company

- You cannot simply ignore missing values in your dataset
- Option1: Do nothing (not recommended)
- Option2: Delete the cases/rows
  - The fact that the value was missing may be informative in itself
  - in the real world, you often need to make predictions on new data even if some of the features are missing
- Option3: Impute the missing values
  - Imputation is replacing a missing value with a reasonable one
    - Average
  - Imputation is found to improve the model performance dramatically when done right.

Missing Values

JOHN BRYCE
Leading in IT Education
a matrix company

- **Missing categorical data**
  - The best way to handle missing data for *categorical* features is to simply label them as 'Missing'!
- **Missing numeric data**
  - You can also **flag and fill** the values.
  - Flag the observation with an indicator variable of missingness.
  - Then, fill the original missing value with 0 just to meet the technical requirement of no missing values.
  - By using this technique of flagging and filling, you are essentially **allowing the algorithm to estimate the optimal constant for missingness**, instead of just filling it in with the mean

Feature Engineering

JOHN BRYCE
Leading in IT Education
a matrix company

- Creating new input features from your existing ones.
- Isolates and highlights key information, which helps your algorithms "focus" on what's important
- You can bring in your own and other people **domain expertise**
- **Domain knowledge**
  - You have a date column and you know something about it (crisis in 2008-2010)

Feature Engineering

JOHN BRYCE
Leading in IT Education
a matrix company

- **Interaction features**
  - Combinations of two or more features
    Product, Sum, Diff, etc
  - Example: Session end – Session start
- **Sparse classes**
  - Group together
- **Dummy Variables**
  - Convert categories/text to bit
  - Set of binary (0 or 1) variables that each represent a single class from a categorical feature
- **Remove unused**
  - ID columns
  - Features that wouldn't be available at the time of prediction
  - Other text descriptions

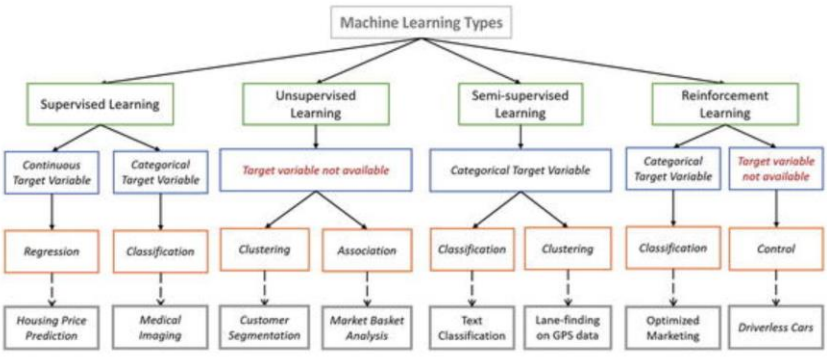
Algorithm Selection

JOHN BRYCE
Leading in IT Education
a matrix company

- In applied machine learning, individual algorithms should be swapped in and out depending on which performs best for the problem and the dataset
- Learn all common algorithms
  - Pros , cons, typical use
- Train and Test the model
  - Split data – train/test
  - Fit and tune
  - Repeat the process with random train/test data

Algorithm Selection

JOHN BRYCE
Leading in IT Education
a matrix company

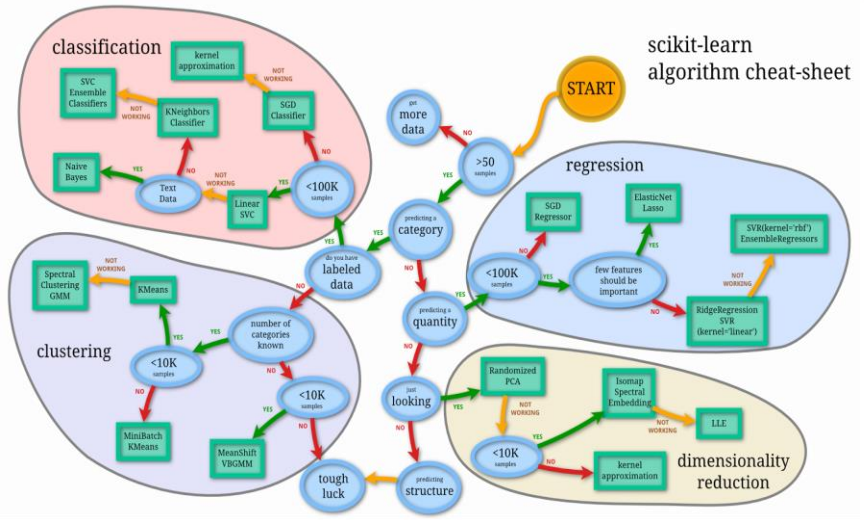


Scikit Learn

JOHN BRYCE
Leading in IT Education
a matrix company

- Every algorithm is exposed in scikit-learn via an "Estimator"
- import the algorithm:
  - from sklearn.linear_model import LinearRegression
- The process for each algorithm depends on its type

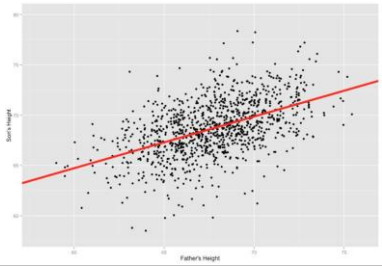
JOHN BRYCE
Leading in IT Education
a matrix company



Linear Regression

JOHN BRYCE
Leading in IT Education
a matrix company

- Our goal with linear regression is to **minimize the vertical distance** between all the data points and our line.
- So in determining the **best line**, we are attempting to minimize the distance between **all** the points and their distance to our line.



Simple Example - Linear Regression

JOHN BRYCE
Leading in IT Education
a matrix company

```
import numpy as np
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression(normalize=True)
```

```
xval = np.array([1,2,3,4,5]).reshape(-1,1)
```

```
yval = [1,2,3,4,5]
```

```
model.fit(xval,yval)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=True)
```

```
model.predict(12)
```

```
array([ 12.])
```

```
model.predict(44)
```

```
array([ 44.])
```

Example

JOHN BRYCE
Leading in IT Education
a matrix company

```
xval = np.array([1,2,3,3,3,3,7,8,9,10]).reshape(-1,1)
yval = [1,2,3,4,5,6,7,8,9,10]
model.fit(xval,yval)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=True)
```

```
model.predict(12)
```

```
array([ 11.74710221])
```

```
model.predict(44)
```

```
array([ 39.90305585])
```


Using Datasets

JOHN BRYCE
Leading in IT Education
a matrix company

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
%matplotlib inline
```

```
df = pd.read_csv('pupils.csv')
```

```
df.head()
```

	Name	Age	Country	Height	Weight	Avg Grades	income	house rooms	family persons
0	dina	10	ISR	110	26	64	10000	6	8
1	noya	6	SP	90	27	68	18200	6	5
2	itamar	7	EN	110	30	71	27000	2	5
3	adar	8	SP	113	30	70	16700	7	6
4	rina	7	EN	100	31	73	30000	2	5

Pupils data - we want to predict average grades

Build the model

JOHN BRYCE
Leading in IT Education
a matrix company

- First we need to define our features and the target we want to predict

```
X = df[['Height', 'Weight',
        'income', 'house rooms','family persons']]
```

```
y = df[['Avg Grades']]
```


Test our Model

JOHN BRYCE
Leading in IT Education
a matrix company

- We have a data and we want to build a model to predict targets
- How do we know that the model (algorithm) is working
- The solution is to split the data – Train and Test
  - For example 70% train and 30% test
- Then run the model on the Train data, and then test on the Test and see if the results are close to the real values.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test , y_train , y_test = train_test_split(X,y,test_size=0.35)
```

Train the model

JOHN BRYCE
Leading in IT Education
a matrix company

```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
model.intercept_
```

```
array([ 55.49313223])
```

```
model.coef_
```

```
array([[  1.04493523e-01,   2.52680760e-01,   6.71599205e-05,
          9.83805382e-02,  -7.24942291e-01]])
```

```
X_train.columns
```

```
Index([u'Height', u'Weight', u'income', u'house rooms', u'family persons'], dtype='object')
```


Model Coefficient

JOHN BRYCE
Leading in IT Education
a matrix company

```
pd.DataFrame(model.coef_.reshape(-1,1),X_train.columns,columns=["Coeff"])
```

	Coeff
Height	0.104494
Weight	0.252681
income	0.000067
house rooms	0.098381
family persons	-0.724942

How increase in one unit affect the target

Using the Test Data

JOHN BRYCE
Leading in IT Education
a matrix company

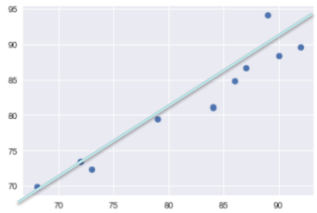
```
predictions = model.predict(X_test)
```

```
predictions
```

```
array([[ 81.14011874],
       [ 80.96265737],
       [ 88.40090084],
       [ 72.36243529],
       [ 94.07724114],
       [ 84.87499382],
       [ 73.48435341],
       [ 86.71113673],
       [ 89.61635687],
       [ 69.90781211],
       [ 79.46572653]])
```

```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x12373c8d0>
```



Predict the target

JOHN BRYCE
Leading in IT Education
a matrix company

```
vals = np.array([100,30,10000,7,3]).reshape(1,-1)
model.predict(vals)
```

```
array([[ 72.70834339]])
```

Logistic Regression

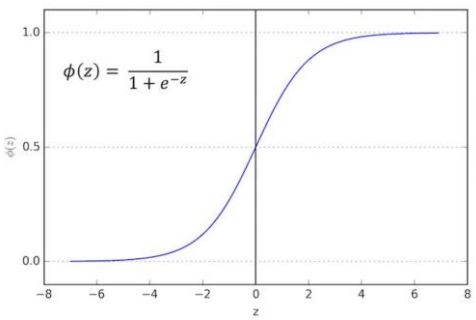
JOHN BRYCE
Leading in IT Education
a matrix company

- Classification method
- Binary classification
  - Answer a Boolean question – yes/no
- Examples:
  - Spam filter
  - Loan default
  - Disease diagnostics

Logistic Function

JOHN BRYCE
Leading in IT Education
a matrix company

- Cant be linear – we don't know how do divide the range
- The Sigmoid (aka Logistic) Function takes in any value and outputs it to be between 0 and 1



Model Evaluation

JOHN BRYCE
Leading in IT Education
a matrix company

- After you train a logistic regression model on some training data, you will evaluate your model's performance on some test data.
- You can use a confusion matrix to evaluate classification models

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Multi class classification

JOHN BRYCE
Leading in IT Education
a matrix company

- Each training point belongs to one of N different classes.
- The goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs
- Each input belongs to exactly one class. Not more, not less
  - If an input can be assigned multiple labels it is called *multi-label classification*

Examples

JOHN BRYCE
Leading in IT Education
a matrix company

- *Input*: hand-written character;
  Output: which character?

- *Input*: a photograph of an object;
  Output: which of a set of categories of objects is it?
- *Input*: a news article;
  Output: which section of the newspaper should it belong to?
- *Input*: an email;
  Output: which folder should an email be placed into?

Binary to multiclass

JOHN BRYCE
Leading in IT Education
a matrix company

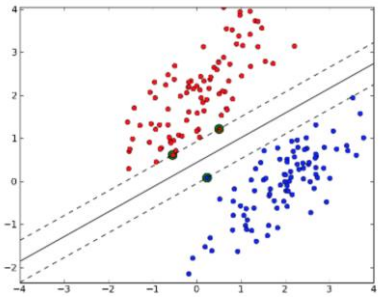
- Can we use a binary classifier to construct a multiclass classifier?
  - Decompose the prediction into multiple binary decisions
- How to decompose?
  - One-vs-all
  - All-vs-all
  - Error correcting codes

Support vector machines

JOHN BRYCE
Leading in IT Education
a matrix company

- SVM is a supervised machine learning algorithm which can be used for both classification or regression challenges.
- However, it is mostly used in classification problems.
- In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

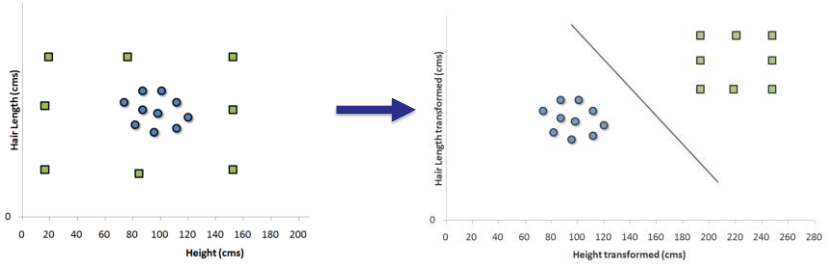
- Then, we perform classification by finding the hyper-plane that differentiate the two classes very well



SVM

JOHN BRYCE
Leading in IT Education
a matrix company

- Its not always easy to find a way to split the data

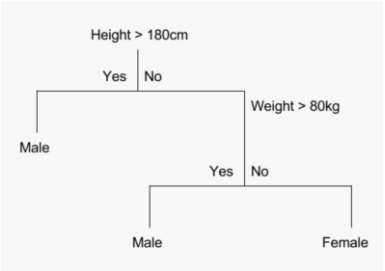


- In such cases, we need to map these vector to a higher dimension plane so that they get segregated from each other
- Many algorithms have been proposed to make these transformations

Decision trees

JOHN BRYCE
Leading in IT Education
a matrix company

- Decision trees algorithms can be used for classification or regression predictive modeling problems.
- Useful with non linear problems (you cant use SVM to linearly divide the space into 2 areas)
- Binary tree , Each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric).
- The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.
- Beware of overfitting



How to split

JOHN BRYCE
Leading in IT Education
a matrix company

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

first split on

X		Y		Z	
X = 0	X = 1	Y = 0	Y = 1	Z = 0	Z = 1
B	A A B	B B	A A	A B	A B

How to split

JOHN BRYCE
Leading in IT Education
a matrix company

- Entropy and Information Gain are the Mathematical Methods of choosing the best split
- To improve performance, we can use many trees with a random sample of features chosen as the split.
- A new random sample of features is chosen for **every single tree at every single split**
- This method is called **Random Forests**

Naive Bayes

JOHN BRYCE
Leading in IT Education
a matrix company

- Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems
- The naive Bayes classifier greatly simplify learning by assuming that features are independent given class
- Naïve Bayes Classifier is amongst the most popular learning method grouped by similarities, that works on the popular Bayes Theorem of Probability- to build machine learning models particularly for disease prediction and document classification

Applications

JOHN BRYCE
Leading in IT Education
a matrix company

- **Sentiment Analysis**- It is used at Facebook to analyse status updates expressing positive or negative emotions.
- **Document Categorization**- Google uses document classification to index documents and find relevancy scores i.e. the PageRank. PageRank mechanism considers the pages marked as important in the databases that were parsed and classified using a document classification technique.
- **Classifying news articles** about Technology, Entertainment, Sports, Politics, etc.
- **Email Spam Filtering**-Google Mail uses Naïve Bayes algorithm to classify your emails as Spam or Not Spam

Advantages

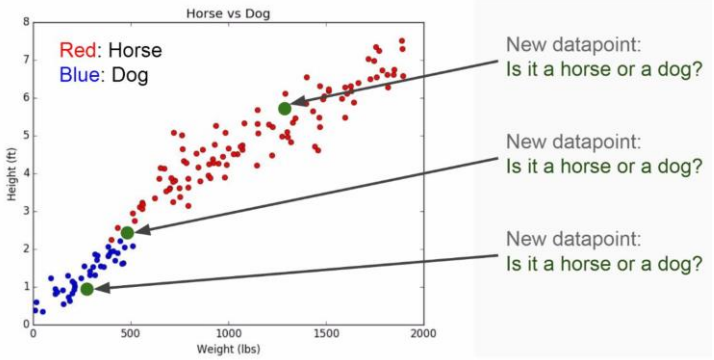
JOHN BRYCE
Leading in IT Education
a matrix company

- Naïve Bayes Classifier algorithm performs well when the input variables are categorical.
- A Naïve Bayes classifier converges faster, requiring relatively little training data than other discriminative models like logistic regression, when the Naïve Bayes conditional independence assumption holds.
- With Naïve Bayes Classifier algorithm, it is easier to predict class of the test data set. A good bet for multi class predictions as well.
- Though it requires conditional independence assumption, Naïve Bayes Classifier has presented good performance in various application domains

Nearest neighbors

JOHN BRYCE
Leading in IT Education
a matrix company

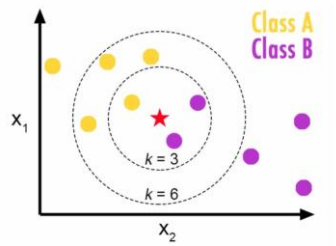
- K Nearest Neighbors is a **classification** algorithm that operates on a very simple principle



KNN

JOHN BRYCE
Leading in IT Education
a matrix company

- Training Algorithm: Store all the Data
- Prediction Algorithm:
  - Calculate the distance from x to all points in your data
  - Sort the points in your data by increasing distance from x
  - Predict the majority label of the "k" closest points
- Choosing a K will affect what class a new point is assigned to



KNN

JOHN BRYCE
Leading in IT Education
a matrix company

- Pros
  - Very simple
  - Training is trivial
  - Works with any number of classes
  - Easy to add more data
  - Few parameters : K , distance metric
- Cons
  - High Prediction Cost (worse for large data sets)
  - Not good with high dimensional data
  - Categorical Features don’t work well

Natural Languages Processing

JOHN BRYCE
Leading in IT Education
a matrix company

- Processing text is a complex task
- We will want to:
  - Compile Documents
  - Featurize Them
  - Compare their features
- Tools:
  - String ops in python
  - Regular expressions
  - External packages

String Operations

JOHN BRYCE
Leading in IT Education
a matrix company

- find, index, count , …
- split , rsplit, lsplit , join ,…
- replace, translate
- isdigit, isalpha, isspace, …

Regular Expressions

JOHN BRYCE
Leading in IT Education
a matrix company

- Find complex text combination
- Extract numbers
- Split with multiple delimiters
- Expression substitution

Regular Expressions

JOHN BRYCE
Leading in IT Education
a matrix company

```
import re
str = 'SELinux : avc:  denied  { add } for' \
       'service=com.android.sampservice.ISampService ' \
       'pid=16277 uid=1000 scontext=u:r:system_app:s0 ' \
       'tcontext=u:object_r:my_service:s0 tclass=service_manager permissive=0'

m = re.search(r"(denied|allowed).{1,20}(service|application)", str)
if m:
    print('Matched',m.groups())
    print('Start index', m.start())
    print('End index', m.end())
```

```
('Matched', ('denied', 'service'))
('Start index', 16)
('End index', 42)
```


Regular Expressions

JOHN BRYCE
Leading in IT Education
a matrix company

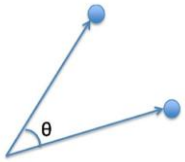
```
import re

str1 = 'string with456, some111:hello 888 num+bers'
txt = re.sub('[0-9]+', '', str1)
print(txt)
# string with, some:hello  num+bers

txt = re.split('[0-9,:+\s]+',str1)
print(txt)
# ['string', 'with', 'some', 'hello', 'num', 'bers']

txt = re.findall('[0-9]+', str1)
print(txt)
# ['456', '111', '888']
```

NLP

JOHN BRYCE
Leading in IT Education
a matrix company

- A document represented as a vector of word counts is called a **Bag of Words**
  - "Blue House" -> (red,blue,house) -> (0,1,1)
  - "Red House" -> (red,blue,house) -> (1,0,1)
- You can use cosine similarity on the vectors made to determine similarity:

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



TF-IDF

JOHN BRYCE
Leading in IT Education
a matrix company

- We can improve on Bag of Words by adjusting word counts based on their frequency in corpus (the group of all the documents)
- We can use TF-IDF (Term Frequency - Inverse Document Frequency)
- Term Frequency - Importance of the term within that document
  - TF(d,t) = Number of occurrences of term t in document d
- Inverse Document Frequency - Importance of the term in the corpus
  - IDF(t) = log(D/t) where

    D = total number of documents

    t = number of documents with the term

TF-IDF

JOHN BRYCE
Leading in IT Education
a matrix company

- Mathematically, TF-IDF is then expressed:

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

NLTK

JOHN BRYCE
Leading in IT Education
a matrix company

- NLTK is a huge package with many natural language modules
- Remove stop words:

```
from nltk.corpus import stopwords

allstopwords = stopwords.words('english')

inputmessage = "I have to see the code today"

outmessage = [word for word in inputmessage.split()
                      if word.lower() not in allstopwords]
print (outmessage)
# ['see', 'code', 'today']
```

Scikit-learn Text processing

JOHN BRYCE
Leading in IT Education
a matrix company

- One useful package for text preprocessing is **sklearn.feature_extraction.text .**
- We can use it to extract and count words from a document, build a vocabulary and more

```
data = ['hello world',
        'have a good day',
        'hello all',
        'how are you',
        'hello and have a nice day'
        ]

from sklearn.feature_extraction.text import CountVectorizer

trans = CountVectorizer().fit(data)

print(trans.vocabulary_)

tr=trans.transform(data)

print (tr)
```


Deep Learning

JOHN BRYCE
Leading in IT Education
a matrix company

- If we have a large amount of features, It is not easy to find out the features that really matter
- For example – we need to find a car in a picture:
  - The pixels – our features (320 * 200 = 64000 features!!!)
  - Some pixels are not important and some very
- Deep Learning uses methods to split the problem to layers
- The popular algorithm is Neural Networks

Our Brain

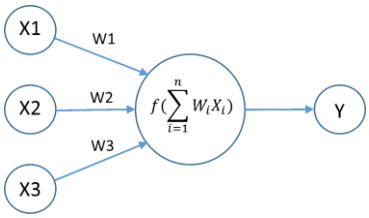
JOHN BRYCE
Leading in IT Education
a matrix company

- 100,000,000,000 neurons
- Each neuron is connected to 10000 other neurons using synapses
- Simple decision uses 200-300 neurons
- Communication speed is measured in ms
  - Signals are varying
- 100THz , 10000 bit computer (at least)

Neural Network

JOHN BRYCE
Leading in IT Education
a matrix company

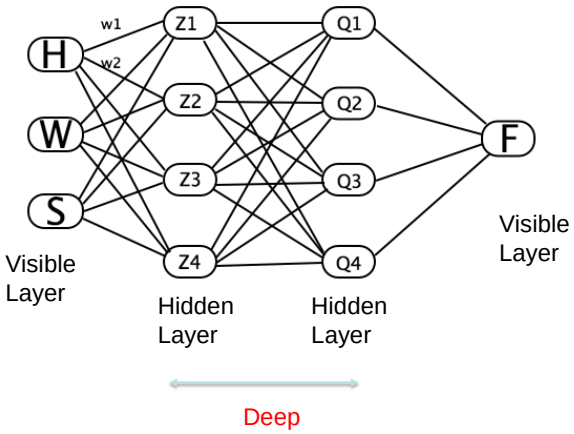
- Based on neurons
- Each neuron has N inputs with weights and one output
- Calculate
  - sum = $\sum_1^n (xi * Wi)$
  - Apply activation function
    - Output = f(sum)



- NN can solve any ML problem, the only difference is the activation function

Neural Network

- The most popular algorithm for deep learning.
- Simple Example:
  - Input: Height, Weight and Shoe size
  - Output: male or female ?
- We will build a network with 2 visible layers (first and last) and 2 hidden layers
- NN usually used for non-linear problems
  - You can solve linear problem with one neuron and identity activation function
- You can solve any ML problem with NN
  - The only difference is the activation function



Training the Network

JOHN BRYCE
Leading in IT Education
a matrix company

- **Forward Propagation**
  - Z1 = f1(H*w1 + W*w5 + S*w9)
    Z2, Z3, Z4 – the same way
  - Q1 = f2(Z1*w13 + Z2*w17 + Z3*w21 + Z4*w25)
    Q2, Q3, Q4 – the same way
  - F = f3(Q1*w29 + Q2*w30 + Q3*w31 + Q4*w32)
- **Back Propagation**
  - Calculate and minimize the error (based on our training data)
  - Need some math (algebra, calculus)
    - Uses gradient descent method
- The training process
  - Find the best w1..w32
  - Find the best activation functions (f1, f2, f3)

JOHN BRYCE
Leading in IT Education
matrix company

Data Analysis and Machine Learning Using Python

Labs

JOHN BRYCE
Leading in IT Education
matrix company

Table of Contents

1. Lab 1 – Python Refresher (45 min)

Your task is to manipulate 2 files and generate a new file

Open a file cust.csv :

Id , name , city
10 , eli , ashdod
20 , avi , haifa
30 , dani , tel aviv
40 , yosi , yafo

And orders file:

id,cust_id,date,sum
1, 10, 1/1/17, 123
2, 20, 2/4/16, 190
3, 10, 2/3/17, 2455
4, 30, 9/2/17, 240
5, 40, 1/5/16, 200
6, 30, 1/2/17, 1240
7, 40, 14/5/16, 1200

Create a file summery.txt with customer name and total orders

eli:2578
avi:190
dani:1480
yosi:1400

2. Lab 2 – Using NumPy (90 min)

- Write a function that takes as input a 2-D ndarray and scales the last row and column by 2
- Create a numpy array with 100 numbers from 1 to 20. You are not allowed to use any loop.
- Create a numpy array containing the sine of the numbers from the above array. You are not allowed to use any loop
- Compute the dot product of arrays [1,2,3,4] and [cos(1),cos(2), cos(3)], but first you have to reshape them into 2-dimensional arrays
- Create two numpy arrays (3,3) - **A, B**
- Compute **A + B**
- Compute **3A + B**
- Find the maximum and the minimum of **B**.
- Normalize the matrix **B** by adding a constant c that makes its minimum be equal to 0. How will you do it for any matrix?
- Compute the sum of the elements in A. You are not allowed to use any loop.
- Compute the transpose of **A**
- Compute the inverse of **A**. Remember there is a module called numpy.linalg.
- Compute the determinant of **A**
- Solve the following equations:
    - X+2Y=3
    - 3X+4Y = 0
- Load the file lin.data and compute least squares to create a linear regression model
    - Build the function yourself
    - Use numpy.linarg package
    - Write a class with the model and function to predict new value

3. Lab 3 – Data visualization with Matplotlib

- Create a numpy array with 100 numbers from 1 to 20. You are not allowed to use any loop.
- Create a numpy array containing the sine of the numbers from the above array. You are not allowed to use any loop
- Create a graph with the last 2 arrays
- Create a function

  def plot(fn,a,b,num_points)

  Shows the plot of function fn between a and b using num_points
- Use it for the function f (x) = x ∗ cos(1/x) in the range:
  - [0 ,pi/4]
  - [-10,10]
  - [0,2]
- Create a text file with 2 lines of numbers:

23 34 45 35 60 40 56 58

2 5 6 8 12 15 18 25

- Create a solid line graph from the above data (first line is y), to read the file use loadtxt method
- Change the graph to green solid line with square markers
- Change the graph to pie graph(use only the first line)
- Add a legend to the pie chart

4. Lab 4 – Image Processing

- Open and display the image file python.png using pyplot
- What is the image shape (dimentions)?
- Write a simple function rgb2gray to convert the image to grayscale

  you need to calculate for each pixel:

  R * 0.299 + G * 0.587 + B * 0.144

  Use dot product

- Repeat the task with the file ang.png – fix the function to support both files (with or without alpha)
- Use scipy signal package to apply med filter on the image
- Use scikit-image to convert the image file to RGB

5. Lab 5 – Pandas

- Open the log file sessions.log with pandas.
- Use head, info to display dataframe
- The dataset fields:
  - Session Id
  - Date – starting session date
  - Time – request time
  - IP – source ip address
  - SSL – T/F if using Http/Https
  - User ID – 0 for anonymous user
  - Landing page – first URL for the session
- Find out :
  - IP addresses with more than 10 visits per month
  - Distribution of authenticated and anonymous users
  - Most visited landing page
  - Find if 2 or more IP addresses was used by 2 or more same users in different days
    - IP X,Y was used by USERS W,Z in the same dates (more than one)
  - How many session was shorter than 5 seconds?
  - How many sessions visited more than 10 pages

6. Lab 6 – Seaborn

Use the log file from lab 5 to illustrate the following:

- Create barplot comparing authenticated and anonymous sessions
- Create a count plot to count sessions by groups:
  - Up to 5 seconds
  - 6 – 30 seconds
  - 30 – 120 seconds
  - more than 2 minutes
- Create box plot with the data from the above sections (grouping and authentication)
- Create heat map to find out dependencies

7. Lab 7 – Linear Regression

In this lab, you will build a model using scikit-learn to predict real estate value

Use the CSV file linearex.csv as your dataset, load it into a data frame

The data contains the following columns:

- 'Avg. Area Income': Avg. Income of residents of the city house is located in.
- 'Avg. Area House Age': Avg Age of Houses in same city
- 'Avg. Area Number of Rooms': Avg Number of Rooms for Houses in same city
- 'Avg. Area Number of Bedrooms': Avg Number of Bedrooms for Houses in same city
- 'Area Population': Population of city house is located in
- 'Price': Price that the house sold at
- 'Address': Address for the house

Tasks:

- Create a pairplot to find correlation between features
- Create a distplot and heatmap
- Create a linear regression model
    - Split the data randomly
    - Fit and test the model
    - Display the results using scatter graph

8. Lab 8 – Logistic Regression

In this lab we will build a model for spam filter. Use the emails.csv file to build a logistic regression model

The fields in the dataset:

- Sender
- Number of rec.
- Email length
- Has attachment
- Subject length

Use heatmap, pairplot to find the relevant data

Build a logistic regression model

- Split the data randomly
- Fit and test the model
- Display the result using confusion matrix

9. Lab 9 – NLP

In this lab you will build an SMS spam filter.

Use the dataset file sms.data

Clean the data , remove all stop words and use feature engineering to create new features

Build a model to predict spam SMS, test it using your data

Use the packages NLTK, scikit learn

10. Lab 10 – Building a simple Neural network

Use the dataset mal_users.csv to build a simple neural network to predict unauthorized users.

The dataset has 14 features and one target column – MAL – T/F

Build a neural network with one hidden layer. Use the sigmoid activation function.

Split the data into train/test , train your network and test the results