

Welcome to the Crash Course in Data Science

If you haven't done so already, please install the Anaconda distro of Python here:

<http://continuum.io/downloads#py34>

Also, please download the files from our github, available here:

<https://github.com/BAH-DSST/DSCrashCourse>

Visit course challenges at: www.datasciencechallenge.com SSID: 26_SouthSeasH_3
Login: blackhat Pass: 2015 Pass: SouthSeasHSeaLion13

Crash Course in Data Science

Charles S. Givre
Edward Raff
Kelly Simmons
Austin Taylor

Who are we?

Charles Givre @cgivre
givre_charles@bah.com

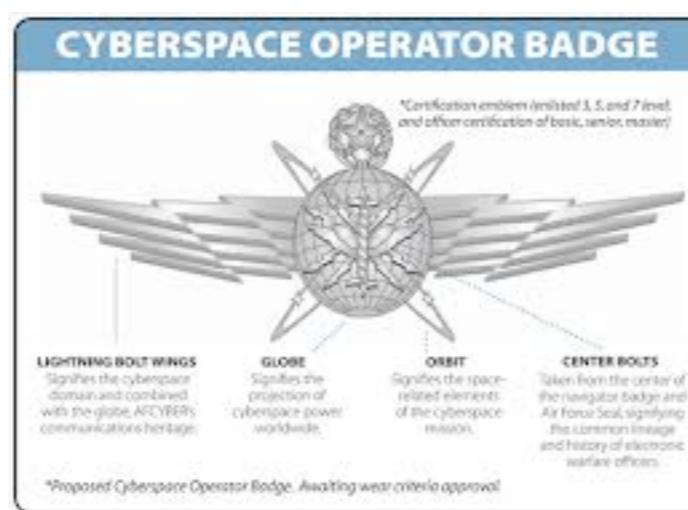


Booz | Allen | Hamilton
— 100 —
YEARS



Austin Taylor
taylor_austin@bah.com

Booz | Allen | Hamilton
100 YEARS



Edward Raff

raff_edward@bah.com

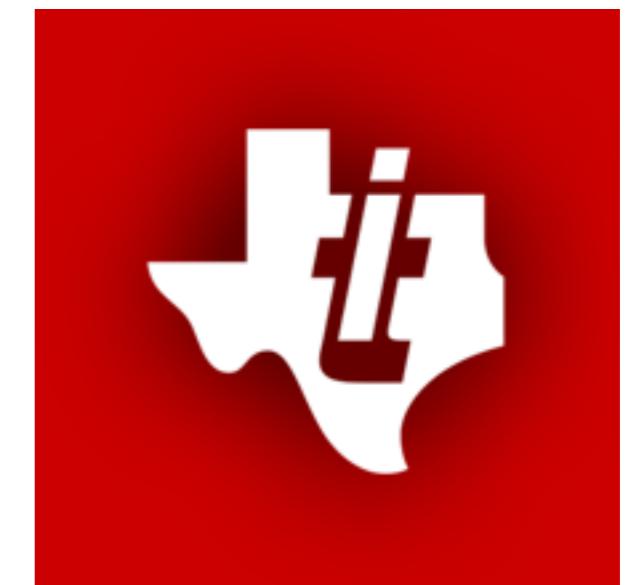


JSAT



Booz | Allen | Hamilton

100 YEARS



Emily Schumm
schumm_emily@bah.com



What is data science?

Data Science is the art of **turning data into actions**. This is accomplished through the creation of data products, which provide actionable information without exposing decision makers to the underlying data or analytics

Field Guide to Data Science, Pg. 17

Data Science is the **extraction of knowledge** from large volumes of data that are structured or unstructured, which is a continuation of the field data mining and predictive analytics, also known as knowledge discovery and data mining (KDD).

https://en.wikipedia.org/wiki/Data_science

Similar to a business/data analyst, data scientists combine knowledge of computer science and applications, modeling, statistics, analytics and math to **uncover insights in data**. Evolving beyond the business/data analyst, the data scientist takes those insights and combines them with **strong business acumen** and **effective communication** to **change the way an organization approach challenges**.

<http://www.i-programmer.info/professional-programmer/accreditation/8331-what-is-a-data-scientist-and-how-do-i-become-one.html>

**Extracting useful
information**

Extracting useful
information **from data**

Answering questions

Answering business questions

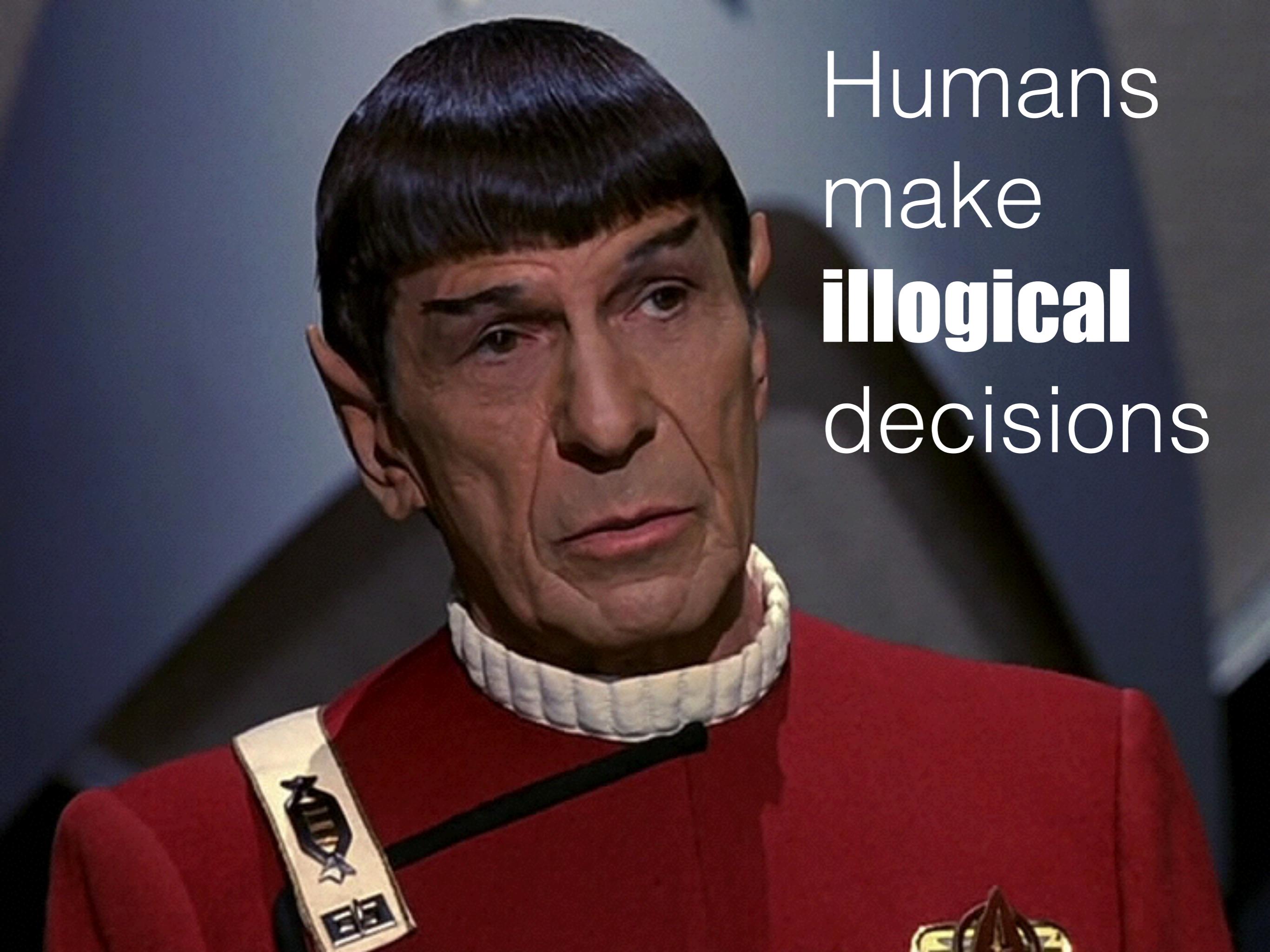
Answering business questions
with data

Answering business questions with data

- Know what you want to know

Answering business questions with data

- Know what you want to know
- Have the **technical skills** to get it



Humans
make
illogical
decisions

Analyst

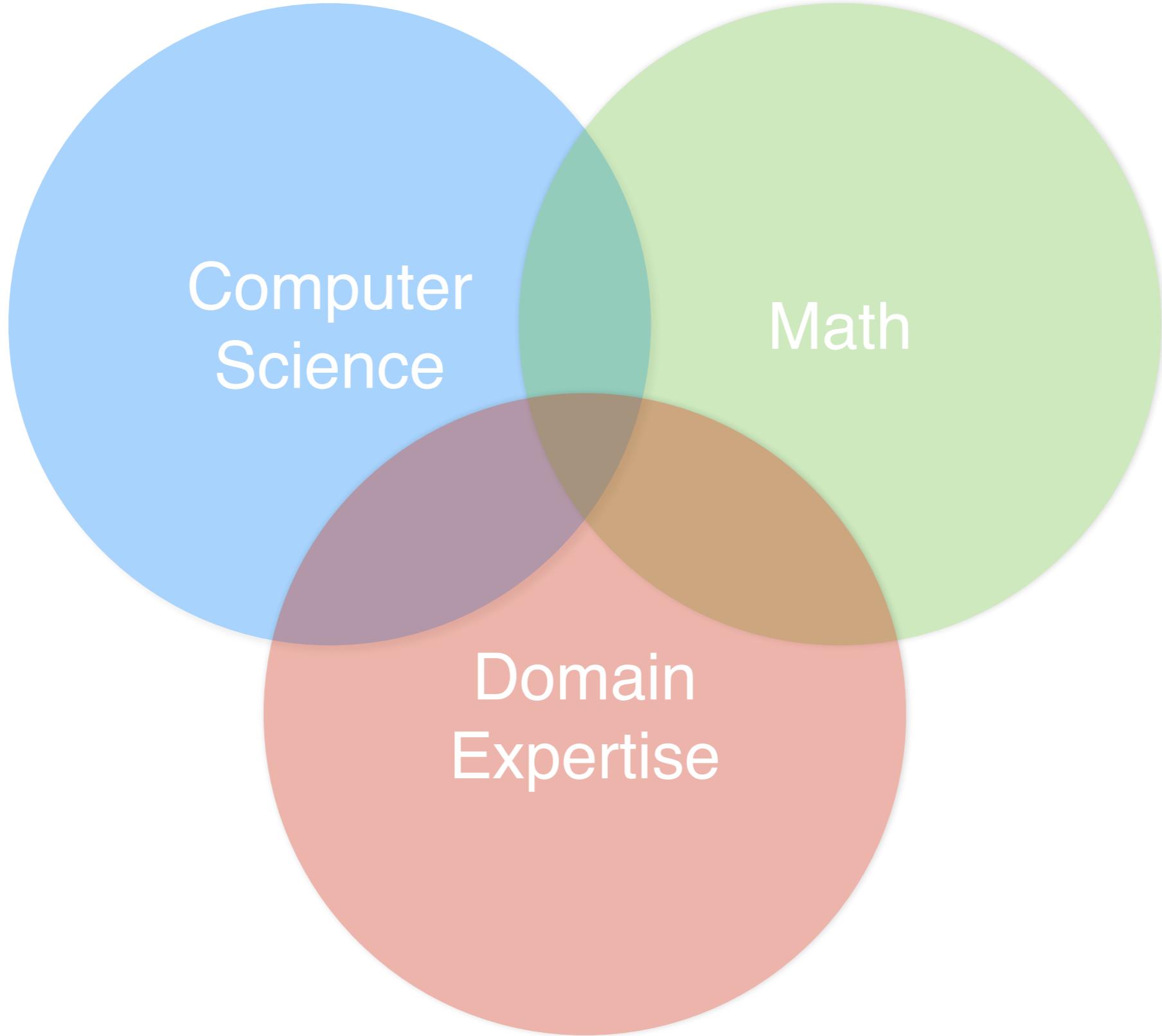


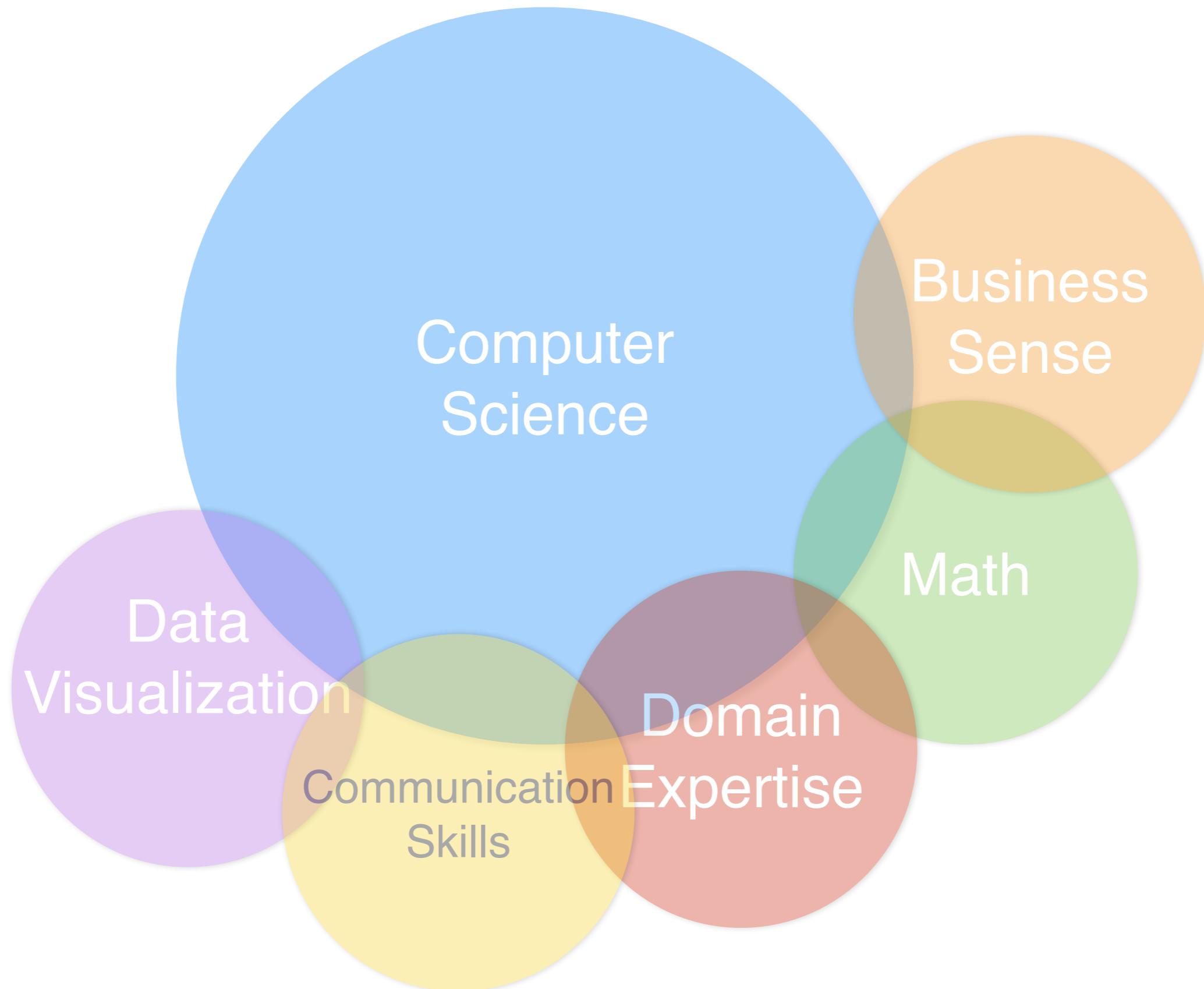
Developer

Analyst + Developer

“The term “data scientist” will subside and may well sound dated five years from now. **The skills will become more commonplace and commoditized. When that happens, the real boom will begin**, because the technology will become widely adopted and thus more useful. . . **Instead, we need self-service tools that empower smart and tenacious business people to perform Big Data analysis themselves.**

–Andrew Brust, “Data scientists don't scale”, <http://www.zdnet.com/article/data-scientists-dont-scale/>





Data Scientists spend
50-90% of their time
being...

Data Janitors!!



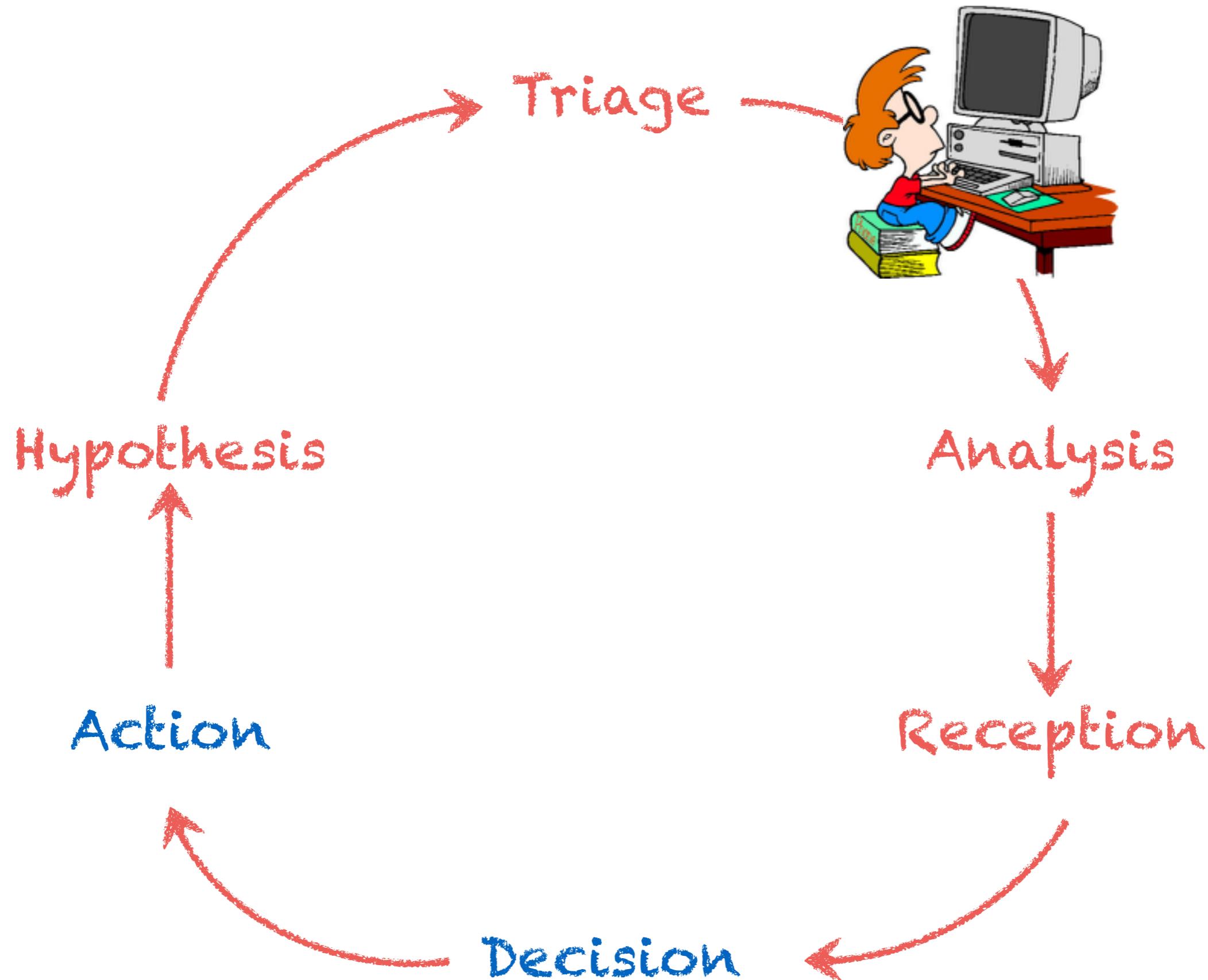
How do you do that?

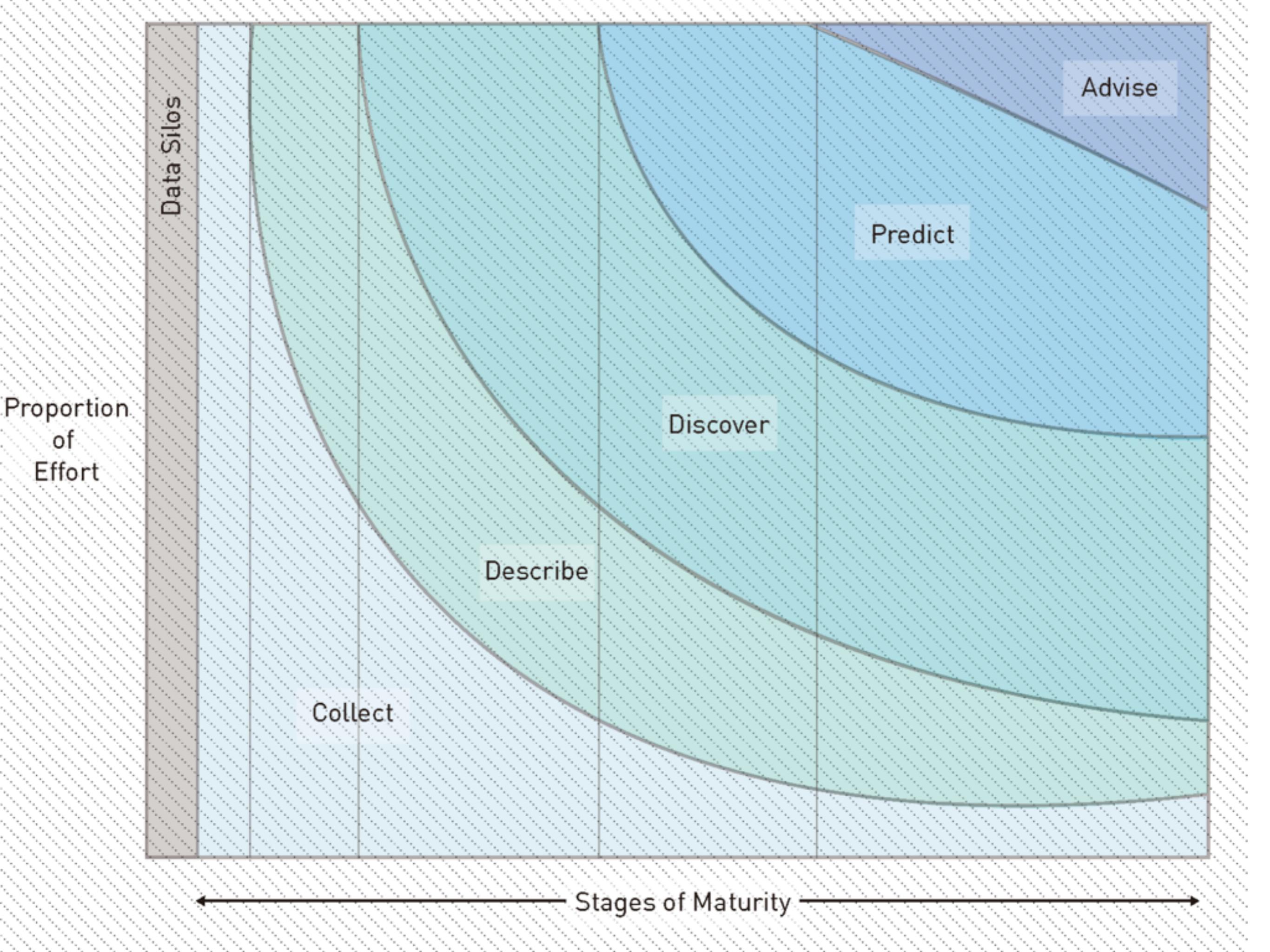


Goals



Strategy





Data is a Strategic Asset... not a cost



Align Projects to Corporate Strategy

Align Projects to Corporate Strategy



Your:
Time
Money
Job?

Build the right team for Data Initiatives

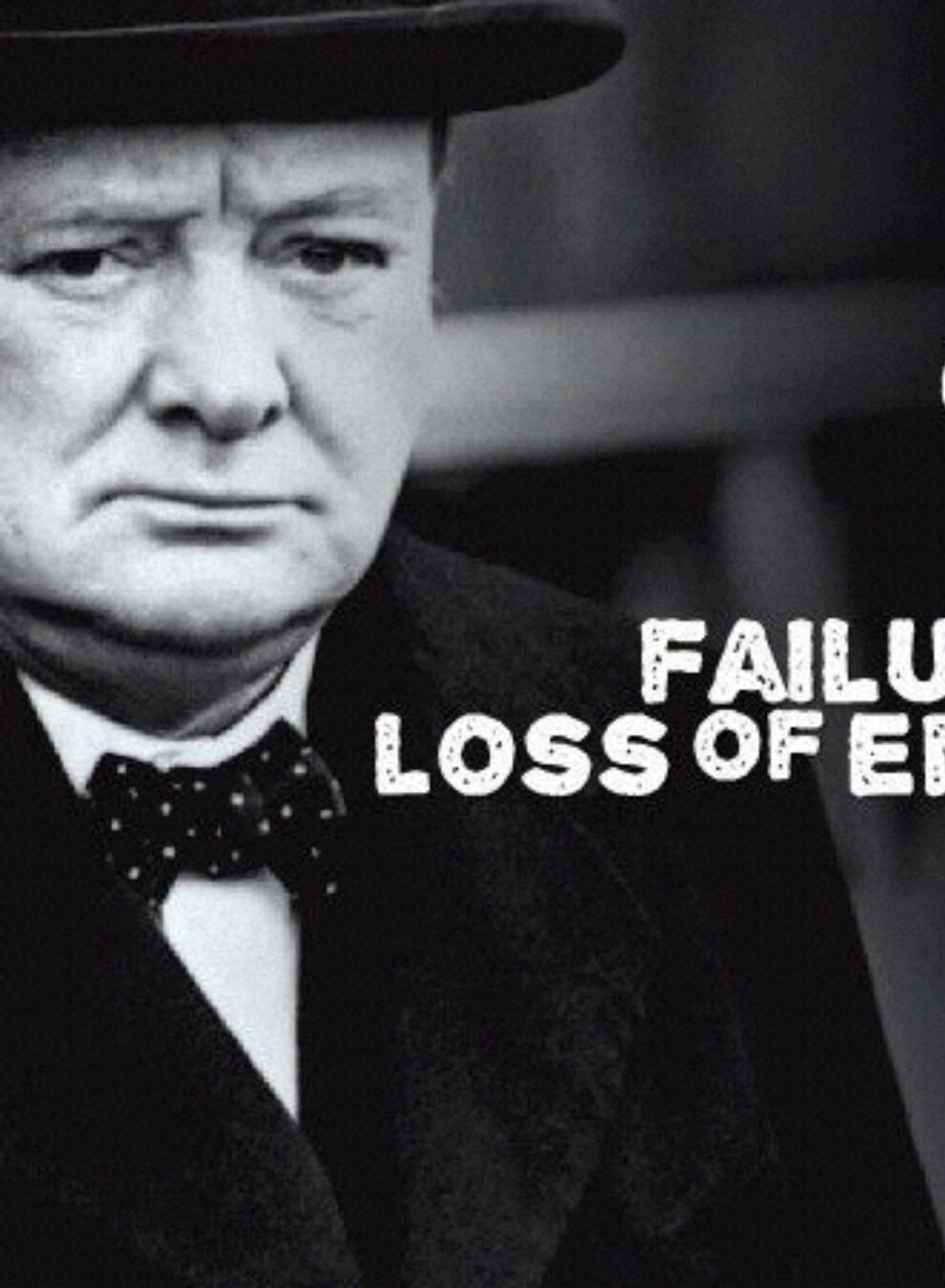


Prioritize building appropriate data platform



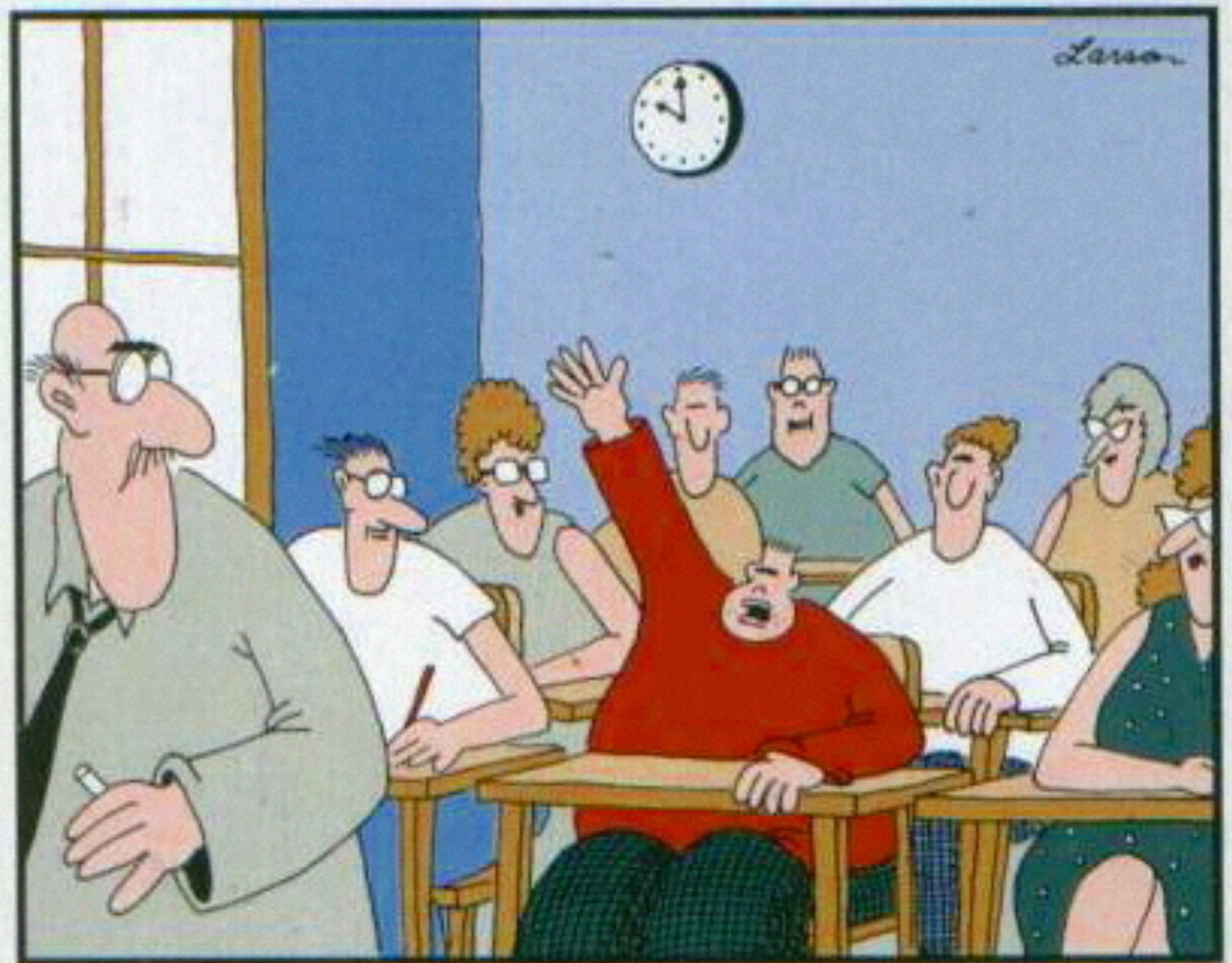
Ingest Data... lots of it





**"SUCCESS
CONSISTS OF
GOING FROM
FAILURE TO
FAILURE WITHOUT
LOSS OF ENTHUSIASM."**

Winston Churchill



**"Mr. Osborne, may I be excused?
My brain is full."**

Build a Data Science Team

Build a Data Science Team

Programmer + Statistician + SME =
DS Team?

Build a Data Science Team

Programmer + Statistician + SME =
DS Team?

Sort of...



Data Ambassador

A close-up profile of a person's face, looking down at a glowing blue sphere. The person has dark hair and is wearing a light-colored striped shirt. The sphere is glowing with a bright blue light, and the background is dark.

Data Scientist (Science Officer)



**Data Storyteller
(Communication Officer)**

A character from Star Trek: The Next Generation, played by actor Robert Picardo, is shown from the waist up. He is wearing a red Starfleet uniform with a gold-striped collar and a gold rank insignia on his left shoulder. A small gold star insignia is pinned to his left chest. He has a dark mustache and is looking slightly to his right with a neutral expression. The background is the interior of a Starfleet ship, featuring metallic walls and a control panel with various buttons and screens.

Data Engineer

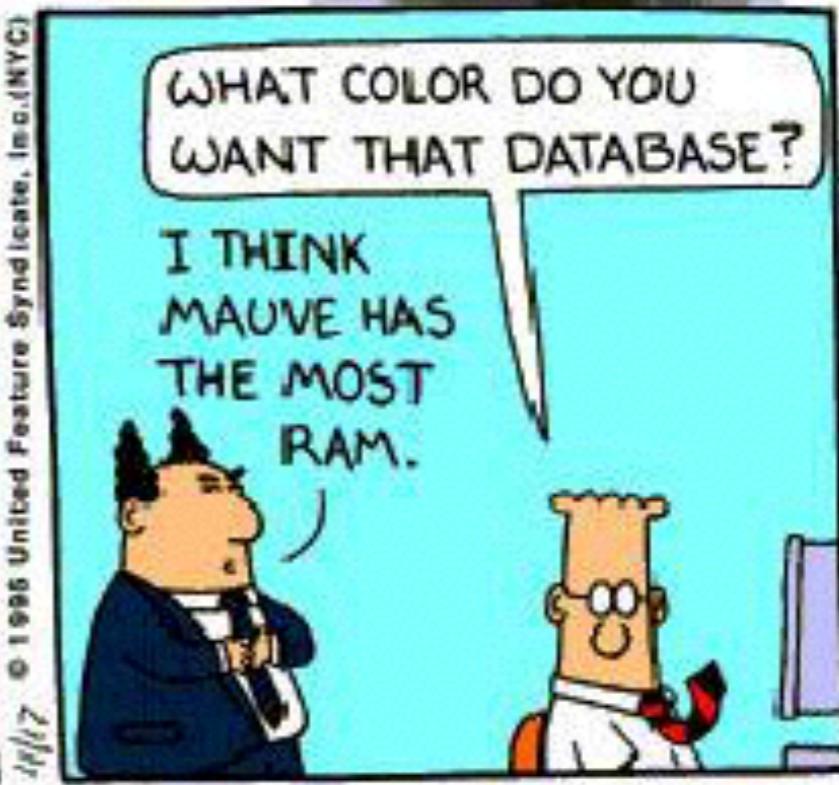
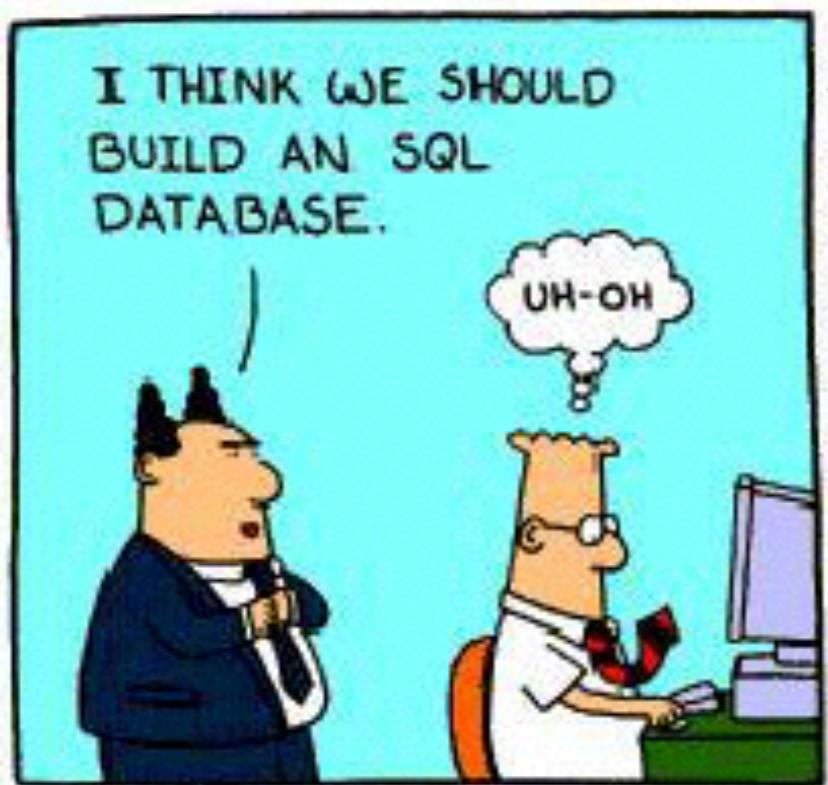


Business Representative

**Someone to ask
stupid questions**



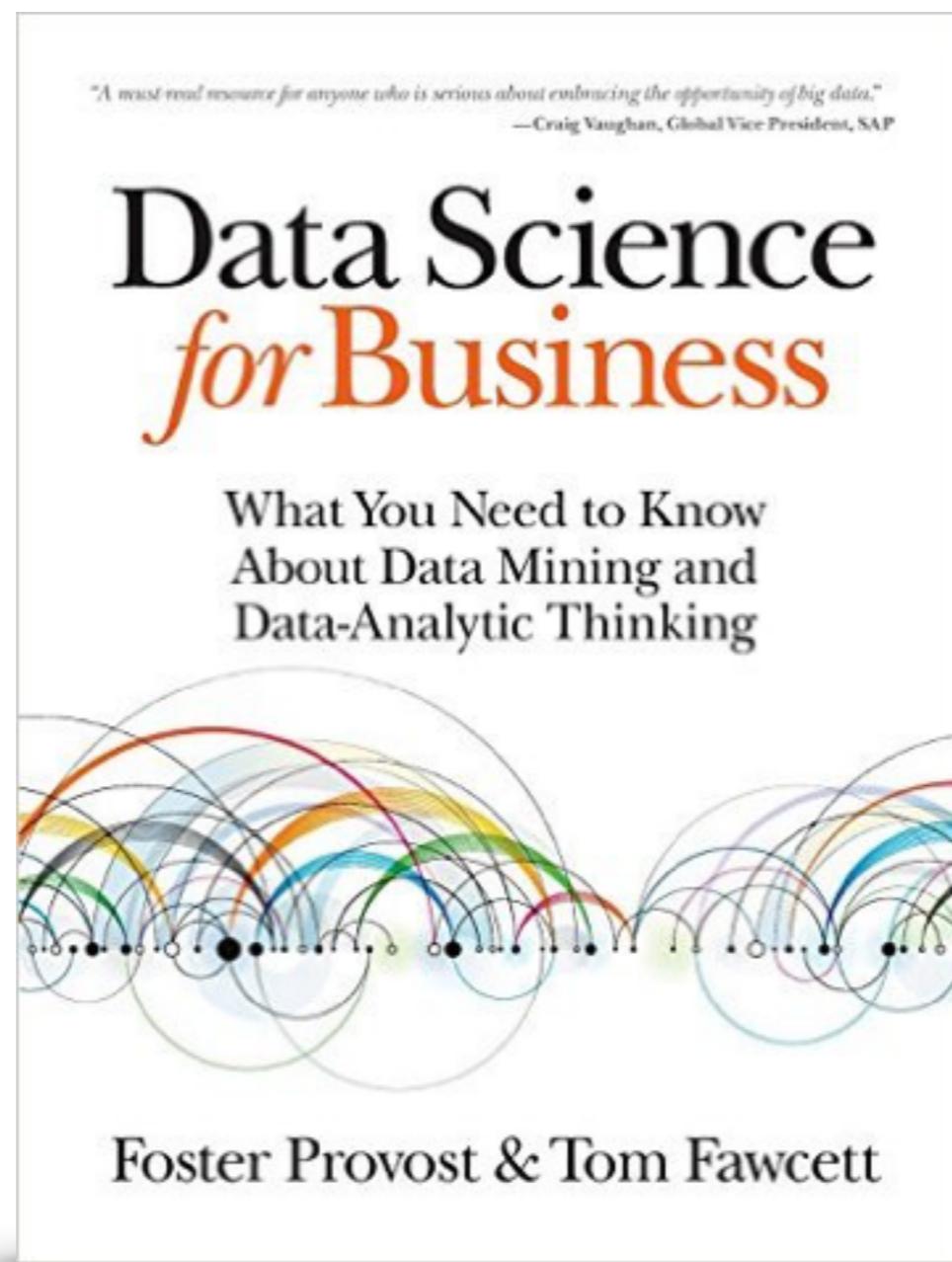
- Quickly and effectively prepare data for analysis
- Apply machine learning techniques to enhance network security



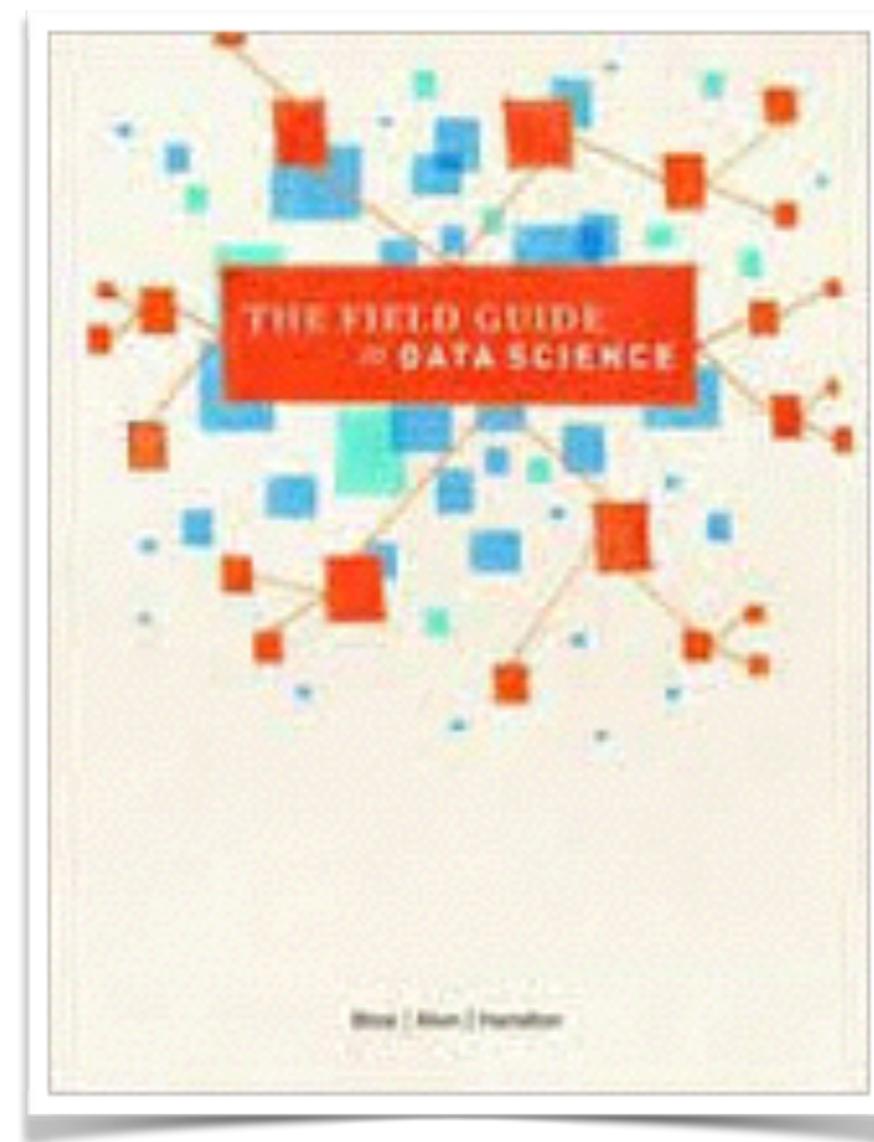


Buzzword

Recommended Reading

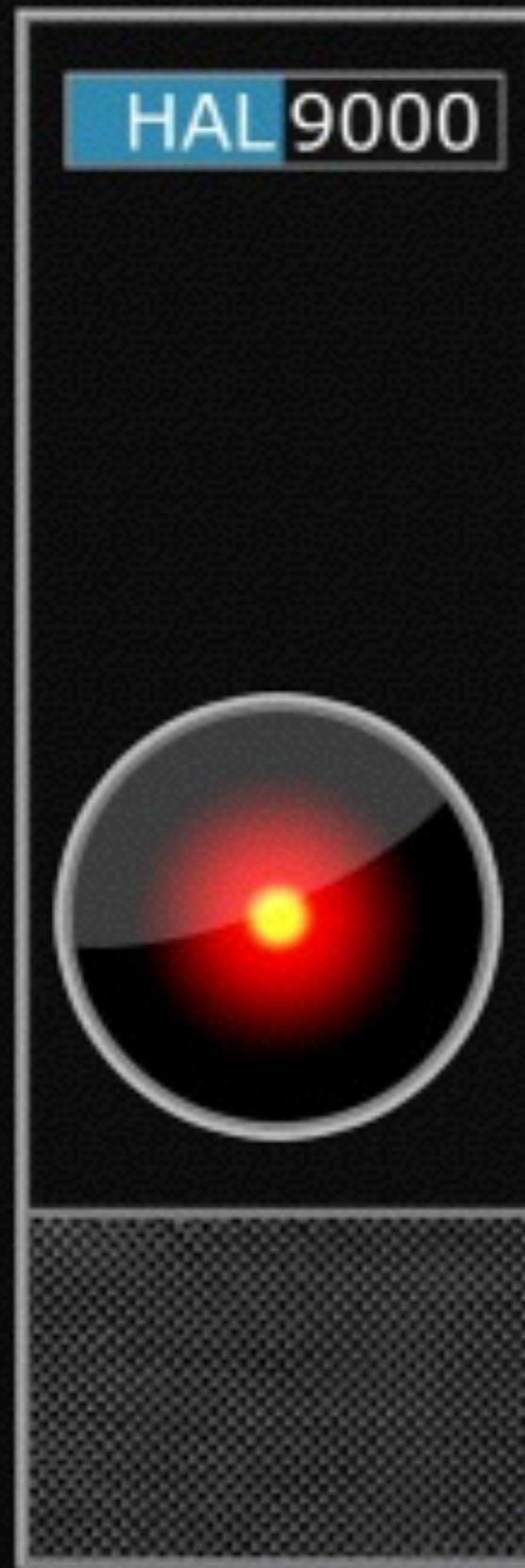


Recommended Reading



<http://www.boozallen.com/media/file/The-Field-Guide-to-Data-Science.pdf>

Machine Learning



What is machine
learning?

“I have no idea. The latest buzzword? ”

–The guy in the back of the class

“Machine learning is the science of getting computers to **act without being explicitly programmed.** ”

– <https://www.coursera.org/course/ml>

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

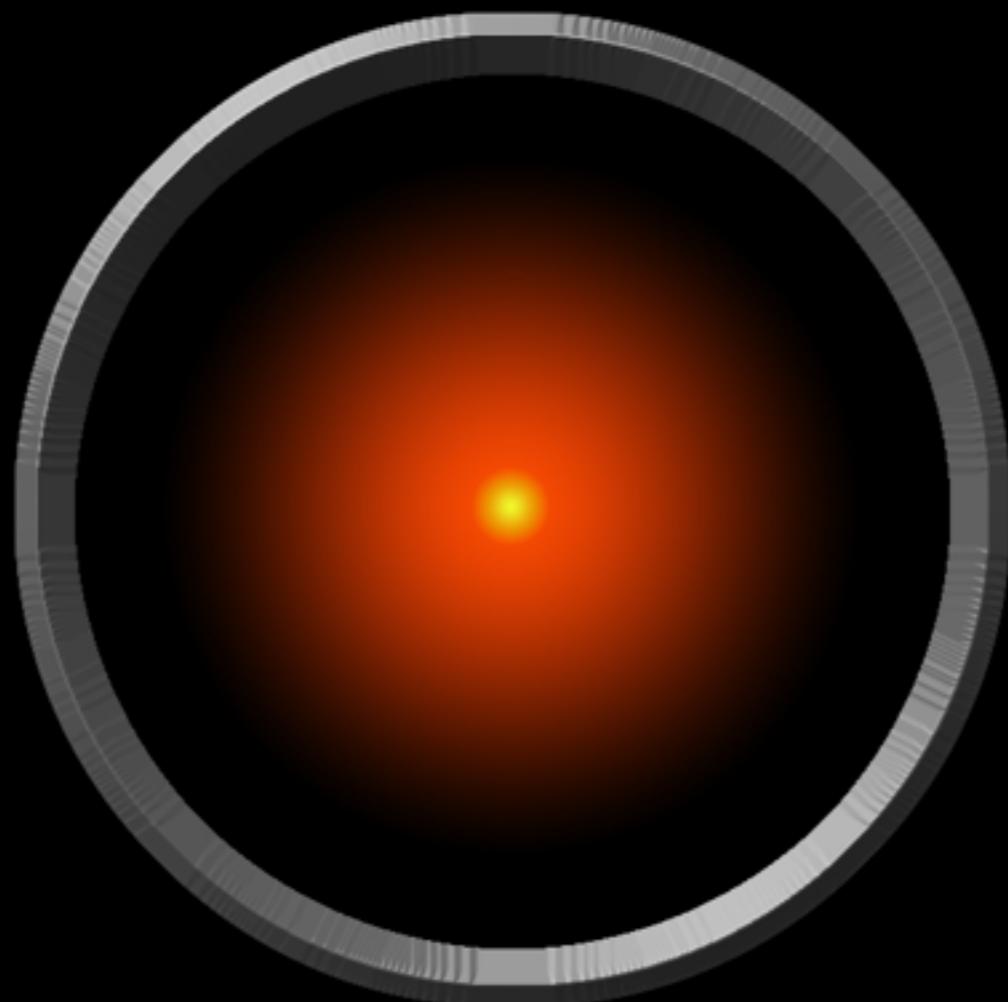
–Tom Mitchell, Carnegie Mellon University

“Machine learning explores the construction and study of algorithms that can learn from and **make predictions on data**. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, **rather than following strictly static program instructions.**”

–https://en.wikipedia.org/wiki/Machine_learning

Automating Decisions

I'm sorry Dave,
I'm afraid I can't do that.



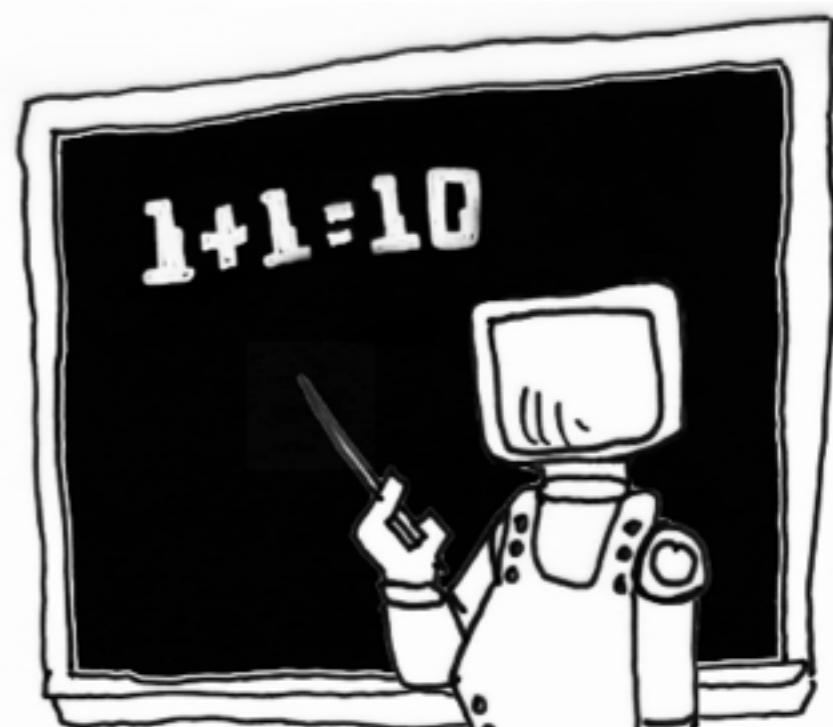
Value Estimation/ Prediction

Identifying Malicious URLs

UNSUPERVISED MACHINE LEARNING



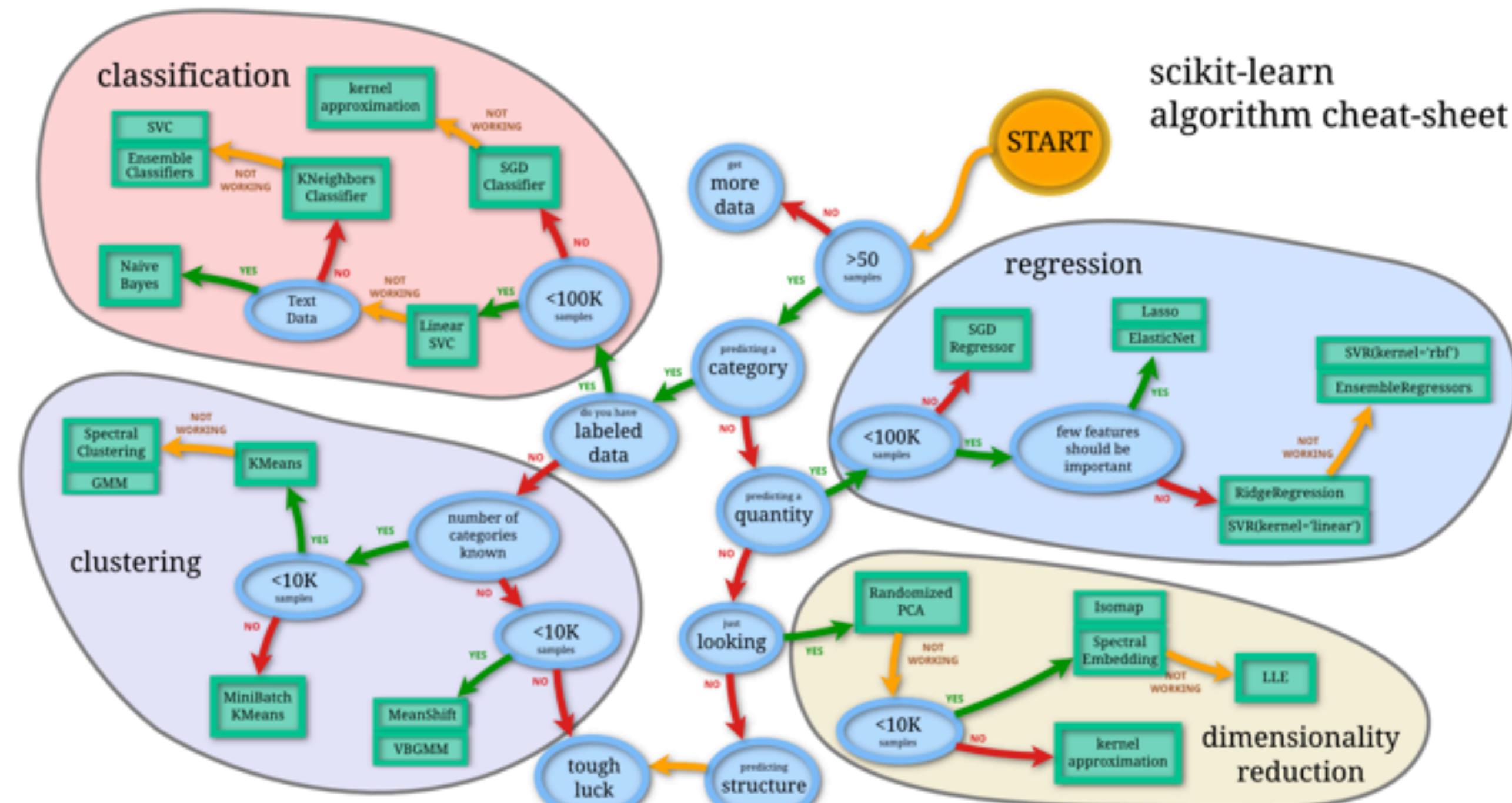
SUPERVISED MACHINE LEARNING





Overfitting

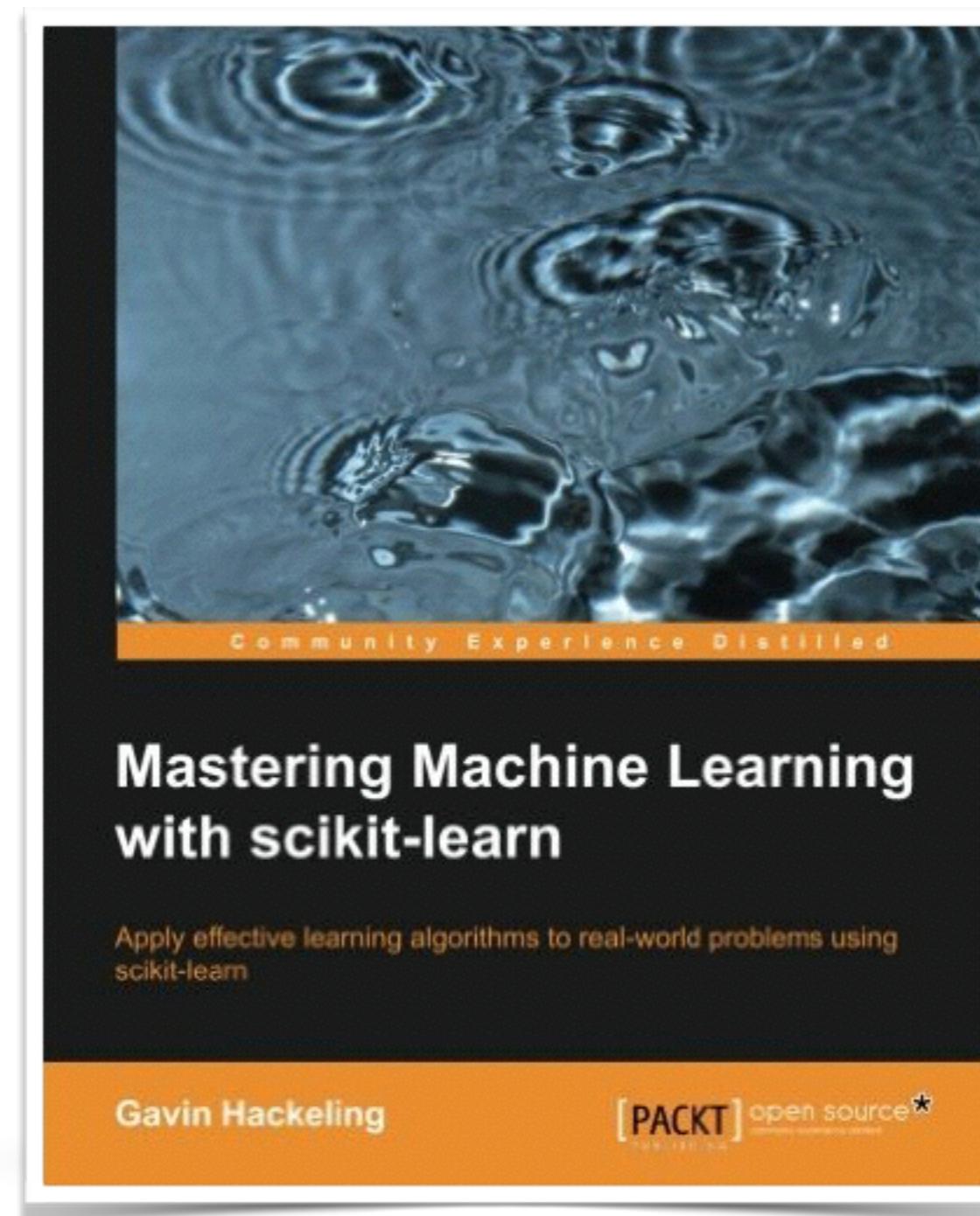
scikit-learn algorithm cheat-sheet



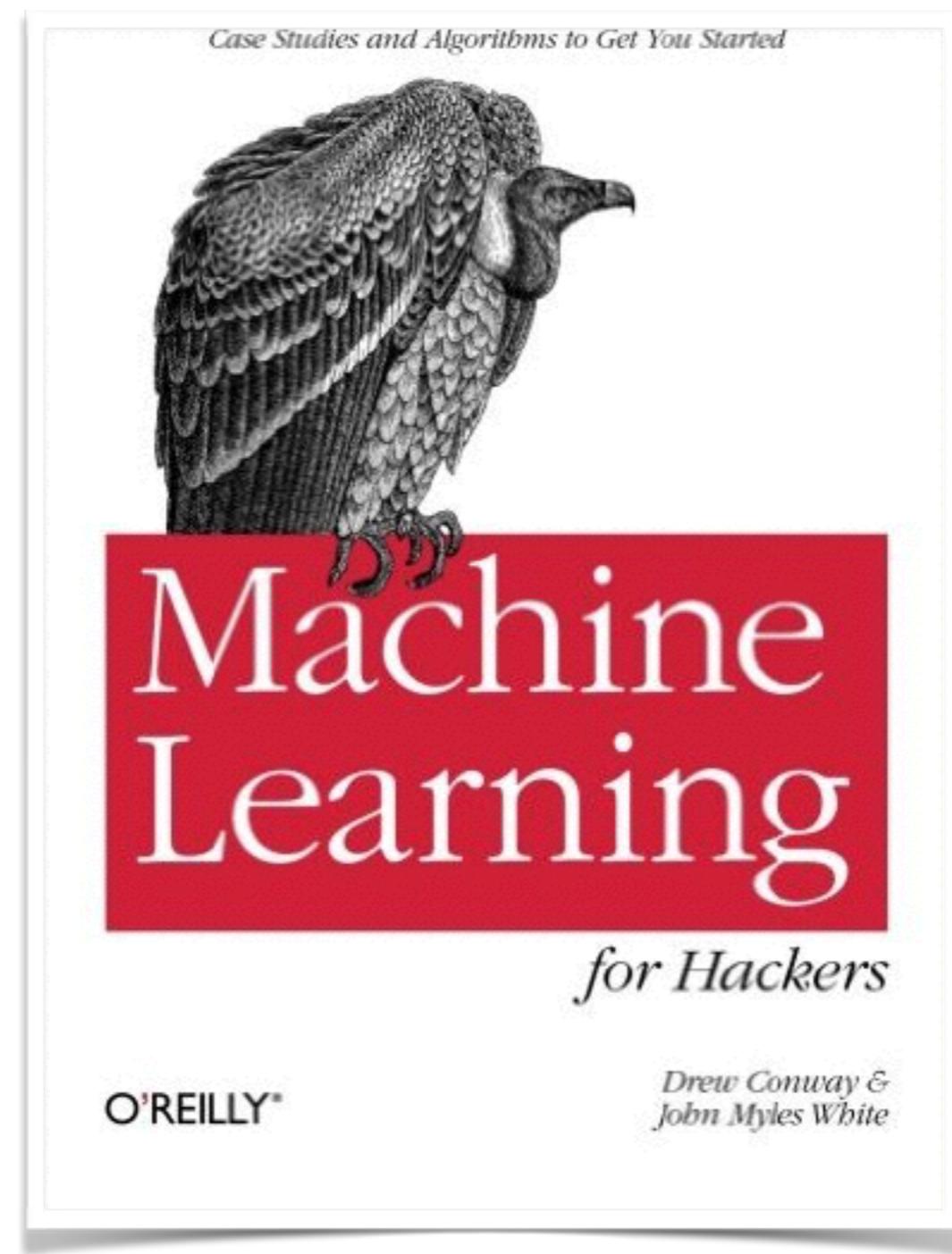
Back

scikit
learn

Recommended Reading



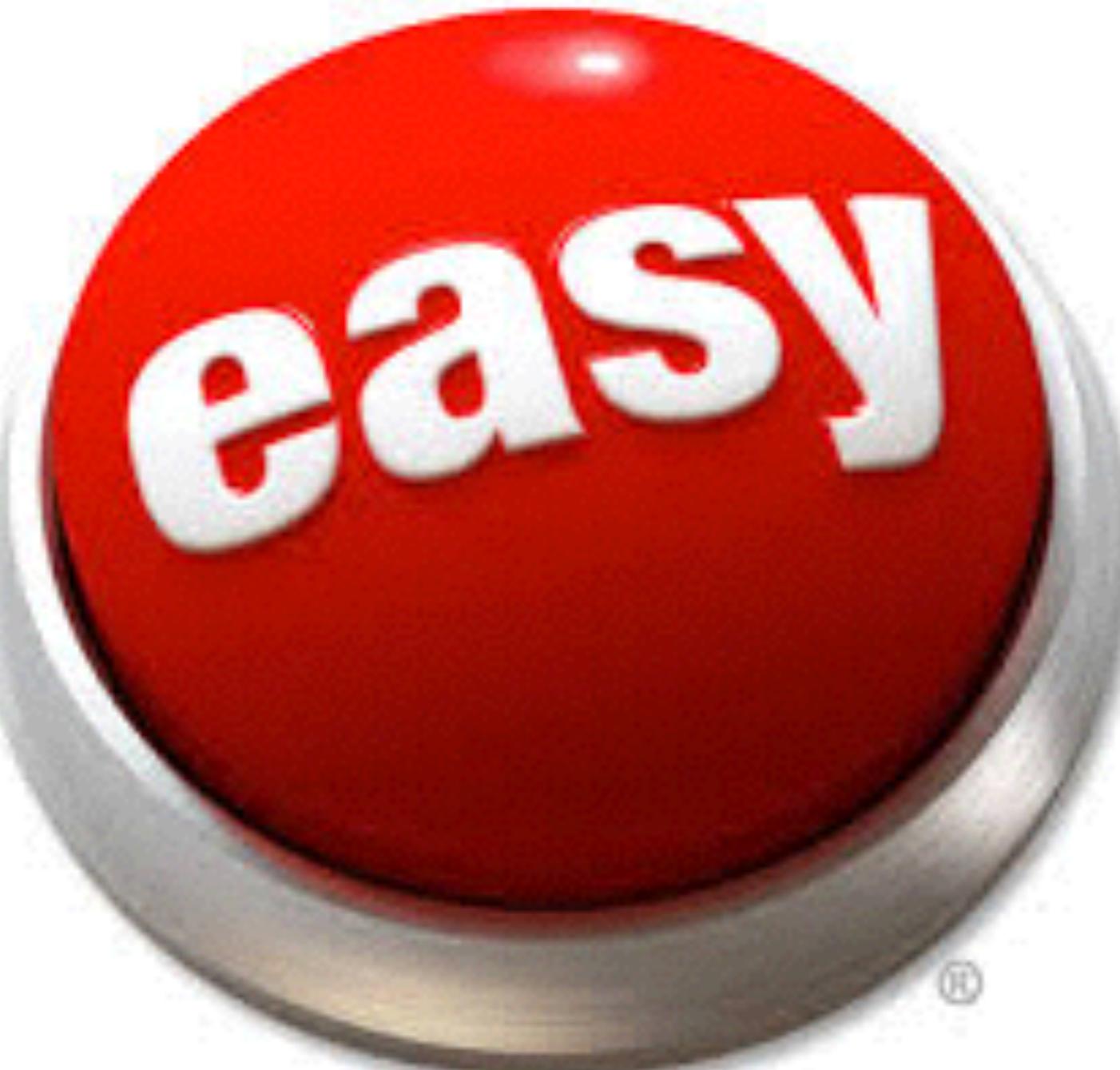
Recommended Reading





Pandas

<https://github.com/BAH-DSST/DSCrashCourse>



easy

®

Dimensions	Name	Description
1	Series	Indexed 1 dimensional data structure
1	Timeseries	Series using timestamps as an index
2	DataFrame	A two dimensional table
3	Panel	A three dimensional mutable data structure

pip install pandas

Pandas Series Object

A Series is like a list

A Series is like a list
or dictionary

A Series is like a list
or dictionary
but a lot **BETTER!**

The Series is the primary building block for all the other data structures we will discuss

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

Creating a Series

```
myData = pd.Series( <data>, index=<index> )
```

Creating a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )  
series2 = pd.Series( { 'firstName' : 'Charles' ,  
                      'lastName' : 'Givre' } )
```

Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1 )
```

0	a
1	b
2	c
3	d
4	e

dtype: object

Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1[3] )
```

d

Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1[1:3] )
```

1	b
2	c

dtype: object

Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1[1:3] )
```

1	b
2	c

dtype: object

```
series2 = series1[1:3]
series2.reset_index( drop=True )
```

Accessing a Series

```
record = pd.Series(  
    {'firstname': 'Charles',  
     'lastname': 'Givre',  
     'middle': 'classified' } )
```

```
record[ 'firstname' ]
```

```
Charles
```

Accessing a Series

```
record = pd.Series(  
    { 'firstname': 'Charles',  
      'lastname': 'Givre',  
      'middle': 'classified' } )
```

```
record[ [ 'firstname', 'lastname' ] ]
```

```
firstname      Charles  
lastname       Givre  
dtype: object
```

Accessing a Series

```
record = pd.Series(  
    { 'firstname': 'Charles',  
      'lastname': 'Givre',  
      'middle': 'classified' } )
```

```
record[0]
```

Charles

Accessing a Series

```
randomNumbers = pd.Series(  
    np.random.randint(1, 100, 50) )
```

```
randomNumbers.head()
```

```
0      48  
1      34  
2      84  
3      85  
4      58  
dtype: int64
```

Accessing a Series

```
randomNumbers = pd.Series(  
    np.random.randint(1, 100, 50) )
```

```
randomNumbers.tail( 7 )
```

```
43      66  
44      66  
45      43  
46      55  
47      99  
48      82  
49      19  
dtype: int64
```

Filtering Data in a Series

```
randomNumbers [ <boolean condition> ]
```

```
randomNumbers [ randomNumbers < 10 ]
```

```
12      5
```

```
21      2
```

```
24      1
```

```
27      1
```

```
29      1
```

```
dtype: int64
```

Filtering Data in a Series

```
record.str.contains('Cha')
```

```
firstname      True
```

```
lastname      False
```

```
middle        False
```

```
dtype: bool
```

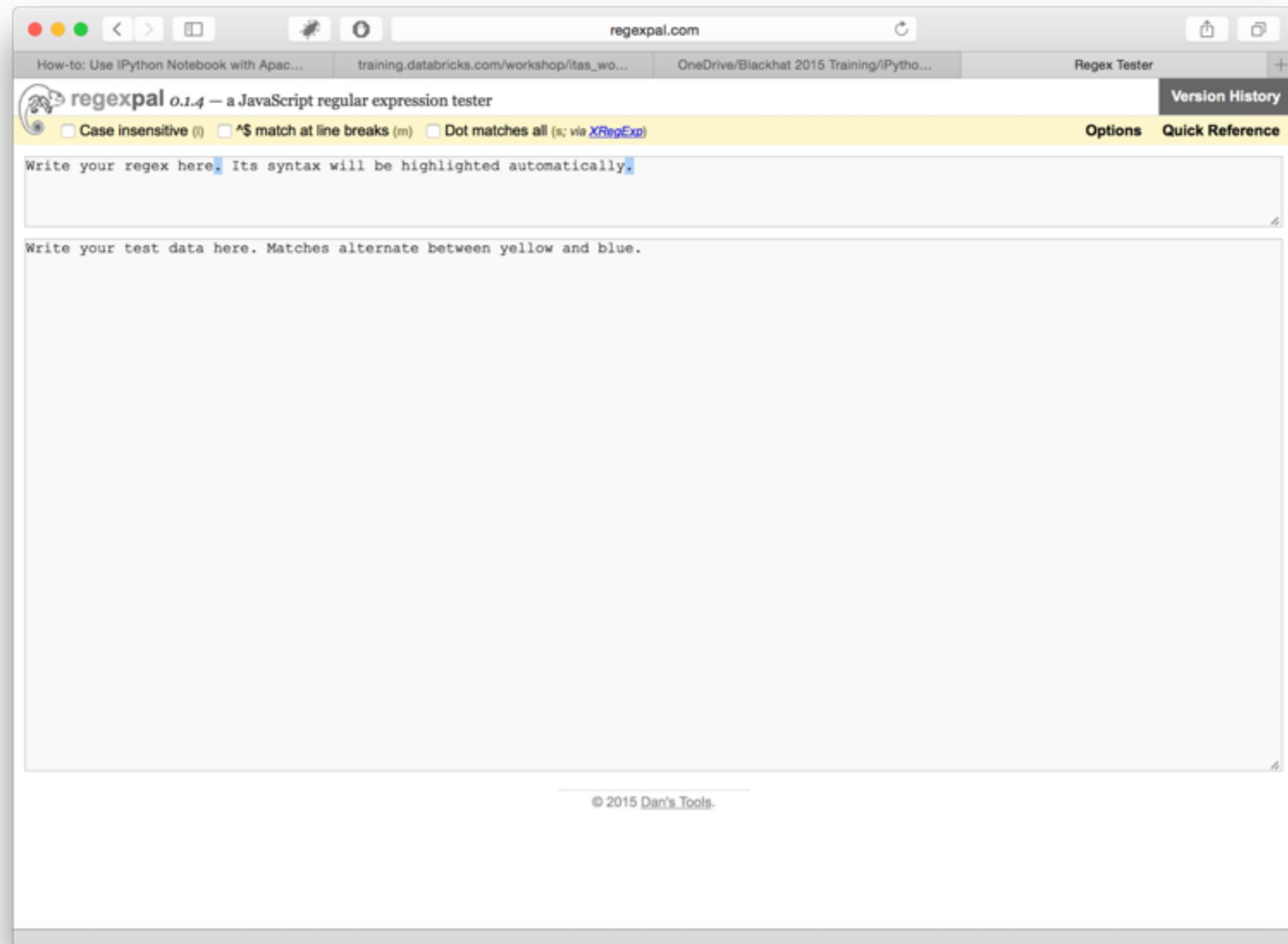
```
record[ record.str.contains('Cha') ]
```

```
firstname    Charles
```

```
dtype: object
```

Regular Expressions

Overview



regexpal.com

Basic Regex – Literal Characters

- The most basic regex will match any characters.
- Regexes are case sensitive
`/grand/`

Matches	Doesn't Match
<ul style="list-style-type: none">• grand• grandfather• grandstand• Babygrand•	<ul style="list-style-type: none">• Grand• Grandmother

Special Characters

- There are a series of special characters that have special functions.
[] \ ^ \$. | ? * + ()
- If you wish to match these characters, you will have to escape them with a \

Special Characters - Example

If you wanted to match:

Booz | Allen | Hamilton

You would have to do it like this:

/Booz \| Allen \| Hamilton/

Non-Printable Characters

- \n – New line
- \t – Tab
- \r – Carriage Return: Note windows files terminate lines with \r\n, UNIX with just \n
- \xA9 – Hex code

Exercises

Write regexes to match:

- $2+2=4$
- “I like regexes”
- (301) 546-6413
- What is my name?
- The price is \$4.50
- givre_charles@bah.com
- c:\temp\secret_files\classifiedData.txt

Character Sets

- Character sets are limited wildcards, and allow you to specify a set of characters to match
- Use square brackets to define a character set
- The dash can define a range of characters
- You can specify multiple ranges as well

Examples

- [aeiou] matches vowels
- [a-z] matches lower case letters
- [a-f] matches any letter a thru f
- [A-Z] matches upper case letters
- [a-zA-Z0-9] matches any alpha-numeric character

Exercises:

Write regexes which match:

- Filenames in the following format: yyyyymmdd-data.xls
- IP Addresses in the format XXX.XXX.XXX.XXX
- Social Security Numbers
- Any 4 letter word beginning with a vowel
- Any 4 letter word with a number at the end

Negated Character Contexts

What if you want to say everything BUT certain characters?

You can create a negated character class with the ^ symbol.

Examples:

- [^0-9] matches everything BUT a digit
- [^a-z] matches everything BUT a lower case letter
- [^aeiou] matches everything BUT a vowel

Exercises:

Use negated character sets to write a character set which defines:

- Any even digit (not odd digits)
- Any character that isn't a vowel

Shorthand for Character Sets

Shortcut	Definition	Example
\s	Any whitespace character	/a\s b/ matches: a b
\S	Any non-whitespace character	/a\S b/ matches : abb
\d	Any digit	\d\d-\d matches 12-3
\D	Any non-digit	/a\Da/ matches aBc or abc
\w	Any alpha-numeric character	
\W	Any non-alpha-numeric character	

Exercises

Rewrite regexes using shortcuts:

- Filenames in the following format: yyyyymmdd-data.xls
- IP Addresses in the format XXX.XXX.XXX.XXX
- Social Security Numbers in the format XXX-XX-XXXX
- Any 4 letter word beginning with a vowel
- Any 4 letter word with a number at the end

Wildcards – The Dot

The first wildcard we will cover is the dot.

- Matches one of any character except a newline.

Scenario: You want to match dates in the following format: mm/dd/yyyy, but you're not sure what the separator is, how could you write a regex to match this?

Wildcards – The dot

Would this work?

`/\d\d.\d\d.\d\d\d\d/`

Try it and see.

Exercise

As we saw, that regex from the previous slide doesn't work because it matches too much. Rewrite it so that it does work.

Hint: use character classes

Repetition – The question mark

The ? indicates something optional. It matches zero or one of the preceding token.

/colou?r/

- Matches color OR colour

Can be combined with parentheses to make larger elements optional:

/Feb 23(rd)?/

- Matches: Feb 23 OR Feb 23rd

Limited Repetition

Regular Expressions allow you to specify the number of times you want something repeated by using the curly braces:

- {n} repeats the previous item n number of times.
- {min, max} allows you to specify a range of times.
- {n,} specifies n or more of the previous item

Examples

Let's say we're looking for dates. What's easier to decipher?

- [0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]
- \d\d\d\d-\d\d-\d\d
- \d{4}-\d{2}-\d{2}

Exercise

Write regexes which match the following:

- IP addresses in the format XXX.XXX.XXX.XXX
- Phone numbers in the format (xxx) xxx-xxxx
- Dates like: Jan 3, 2012. (For now, just use three letters for the month)

Repetition Using Star and Plus

There are two additional repetition operators, the * and the +.

- The * matches zero or more of the previous element
- The + matches one or more of the previous element

Examples

/[A-Za-z]\w*/

Matches HTML tags such as , <html>, <h1> etc.

Exercises

Write regexes which match:

- Email addresses in the format username@domain
- URLs beginning with http or https
- UNIX file paths in the /var/ folder (ie: /var/etc/data.txt)

Pitfalls

Let's say we're trying to find HTML tags:

<h2>This is an html heading</h2>

Consider the following regex:

/<.+\>/

Would this match the HTML tags?

Anchors

There are two characters which can be used to denote position within a string:

- `^` matches strings beginning with the next element
- `$` matches strings ending with the previous element.

Examples:

- `/^$/` matches a completely empty string
- `/^\d*$` matches a string containing zero or more digits, and only digits.

Exercises

Use anchors to write a regex to:

- Match passwords that are exactly 14 characters long and contain upper and lower case letters and numbers.
- Match strings which begin with anything other than a space.
- Match text which ends with a date in the format yyyy-mm-dd

Word Boundaries

Sometimes you might want to match words as opposed to substrings:

Regular Expressions allow you to do this with the \b operator.

`/\bBooz\b/`

Exercises

Write a regexes which match:

- Words beginning with an upper case letter
- Words ending with a number

Alternation

You can use the alternation operator to specify several possibilities. In this case, you might want to use parens to avoid ambiguity.

Examples:

- `/M(o|u)hammad/` matches Mohammad or Muhammad
- `/^notes\.(doc|xls|ppt)x?$/` matches various file names.

String Functions

Function	Explanation
Series.str.contains(<pattern>)	Returns true/false if text matches a pattern
Series.str.count(<pattern>)	Returns number of occurrences of a pattern in a string
Series.str.extract(<pattern>)	Returns matching groups from a string
Series.str.find(<string>)	Returns index of first occurrences of a substring (Note: not regex)
Series.str.findall(<pattern>)	Returns all occurrences of a regex
Series.str.len()	Returns the length of text
Series.str.replace(<pat>, <replace>)	Replaces matches with a replacement string

In Class Exercises

Please open the Series Worksheet and complete exercises 1 & 2.

Exercise 1

```
randomNumbers =  
pd.Series( np.random.random_integers(1, 100, 100) )  
  
evenRandomNumbers =  
randomNumbers[ randomNumbers % 2 == 0 ].reset_index( drop=True )
```

Exercise 2

```
phoneNumbers = pd.Series( numbers )  
  
validPhoneNumbers =  
    phoneNumbers[  
        phoneNumbers.str.match( r'\(\d{3}\)\d{3}-\d{4}' )  
    ].reset_index( drop=True )
```

Operations on a Series

```
for x in range(0, len( data )):  
    data[ x ] = data[ x ] + 2
```

Pandas Data Structures allow you to perform operations on
your **entire data set at once.**

odds = evens + 1

```
combinedSeries = series1.add( series2 )
```

```
def addTwo( n ):  
    return n + 2  
  
odds.apply( addTwo )
```

```
odds.apply( lambda x: x + 2 )
```

IP Address Module

What is it?

- `ipaddress` provides the capabilities to create, manipulate and operate on IPv4 and IPv6 addresses and networks.

Import `ipaddress`

The `ipaddress` module supports both IPv4 and IPv6

```
myIP = ipaddress.ip_address('192.168.0.1')
IPv4Address('192.168.0.1')
```

or

```
myIP = ipaddress.ip_address('2001:db8::')
IPv6Address('2001:db8::')
```

Ipaddress.ip_network

Ipaddress module also allows for creation of network objects.

```
myNetwork = ipaddress.ip_network('192.168.0.0/28')
IPv4Network('192.168.0.0/28')
```

Bringing it together

```
In [1]: import ipaddress
```

```
In [2]: myIP = ipaddress.ip_address('192.168.0.1')
```

```
In [3]: myNetwork = ipaddress.ip_network('192.168.0.0/28')
```

```
In [4]: myIP in myNetwork
```

What will be returned?

Out[4]: True

Loop through network

```
>>> for ip in myNetwork:  
....:     print(ip)
```

```
192.168.0.0
```

```
192.168.0.1
```

```
...
```

```
...
```

```
192.168.0.13
```

```
192.168.0.14
```

```
192.168.0.15
```

IP Class Methods

is_multicast

- True if the address is reserved for multicast use.
See RFC 3171 (for IPv4) or RFC 2373 (for IPv6)

is_private

- True if the address is allocated for private networks. See iana-ipv4-special-registry (for IPv4) or iana-ipv6-special-registry (for IPv6).

is_global

- True if the address is allocated for public networks.
New in version 3.4.

IP Class

<https://docs.python.org/3/library/ipaddress.html>

In Class Exercise

In Series Worksheet 2, please complete exercises 1 & 2

Exercise 1

```
tempsinCelsius = tempsInFarenheit.apply( toCelsius )
print( tempsinCelsius )
```

```
0      33.333333
1      0.555556
2     -20.555556
3     -8.333333
4      50.000000
5     30.555556
dtype: float64
```

Exercise 2

```
from ipaddress import ip_address
IPData = pd.Series( hosts )
privateIPs = IPData[
    IPData.apply( lambda x : ip_address(x).is_private )
]
print( privateIPs )
```

A few odds & ends

```
mySeries = pd.Series( [1,2,3,4] )
```

```
#Sort by value  
mySeries.sort()
```

```
#Sort by index  
mySeries.sort_index()
```

```
#Reverse order  
mySeries.sort( ascending=False )
```

A few odds & ends

```
mySeries = pd.Series( [1,2,3,4] )
```

#You can use the append method

```
mySeries.append( pd.Series( [5] ) )
```

#Or by place

```
mySeries[6] = 6
```



Explore your Data

- **Mean:** The mean is the average value of a set of numbers
- **Median:** The median is the middle value of an ordered set of numbers
- **Mode:** The mode is the value from a set which is repeated the most frequently.
- **Range:** Difference between minimum and maximum
- **Standard Deviation:** Measures dispersion in a data set. Close to zero indicates little dispersion.

Series.abs()	Absolute Value of the Series
Series.count()	Returns number of non-empty values in the series
Series.max()	Returns maximum value in the Series
Series.mean()	Returns the mean of a Series
Series.median()	Returns the median of a Series
Series.min()	Returns the minimum value in a Series
Series.mode()	Take a guess..
Series.quantile([q])	Returns the quantiles of a Series
Series.sum	Returns the sum of a series
Series.std	Returns the Std. Dev of a Series

`Series.describe()`

Series.describe()

```
count      50.000000
mean      52.000000
std       29.154759
min       3.000000
25%      27.500000
50%      52.000000
75%      76.500000
max     101.000000
dtype: float64
```

```
names = pd.Series(  
    [ 'Jim' ,  
    'Bob' ,  
    'Bob' ,  
    'Steve' ,  
    'Jim' ,  
    'Jane' ,  
    'Steph' ,  
    'Emma' ,  
    'Rachel' ] )
```

```
names.describe( )
```

```
names = pd.Series( ['Jim', 'Bob', 'Bob',  
'Steve', 'Jim', 'Jane', 'Steph', 'Emma',  
'Rachel'])
```

```
names.describe()
```

count	9
unique	7
top	Jim
freq	2
dtype:	object

Find Unique Values

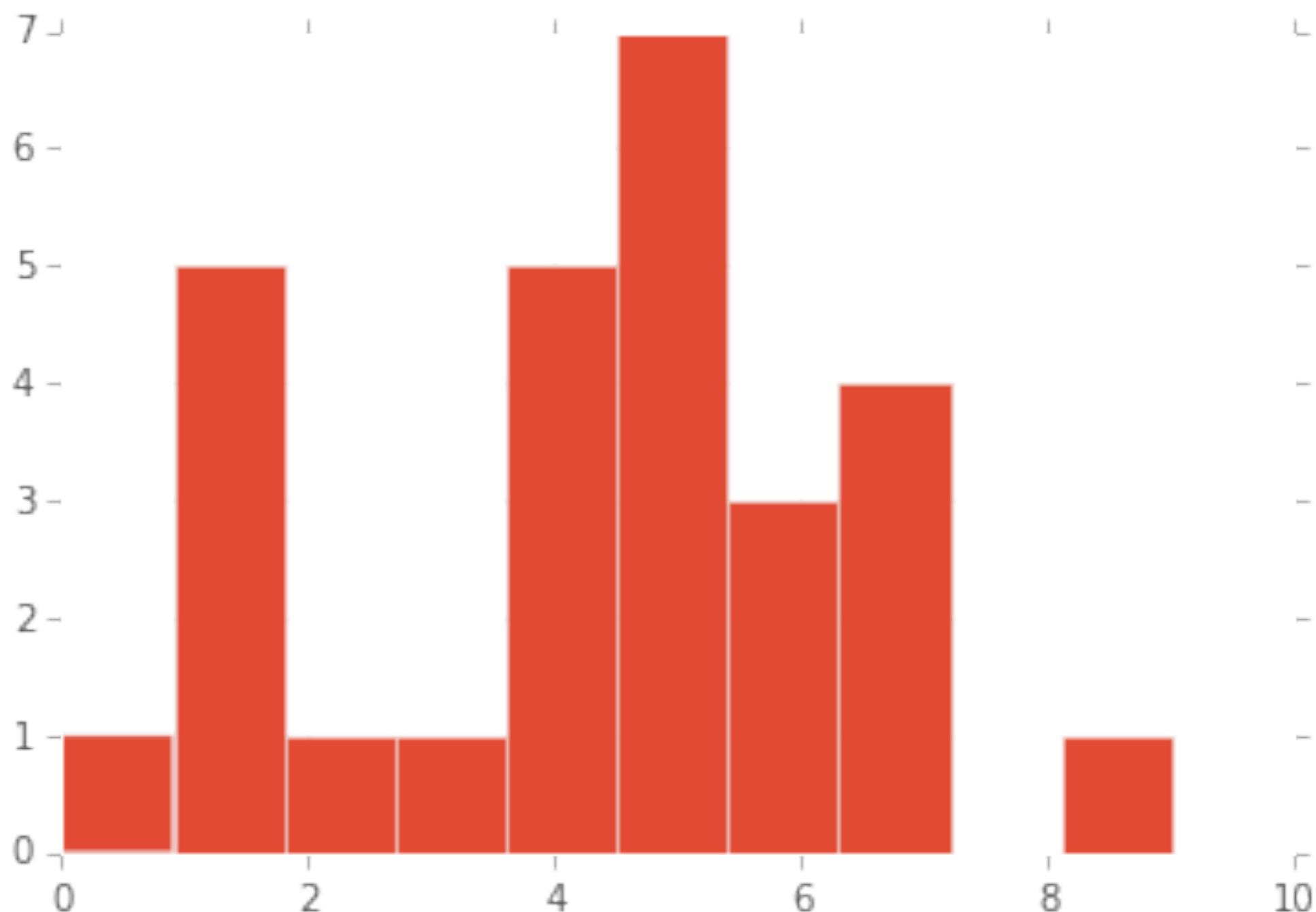
```
mySeries.unique()
```

Find Unique Value
Counts

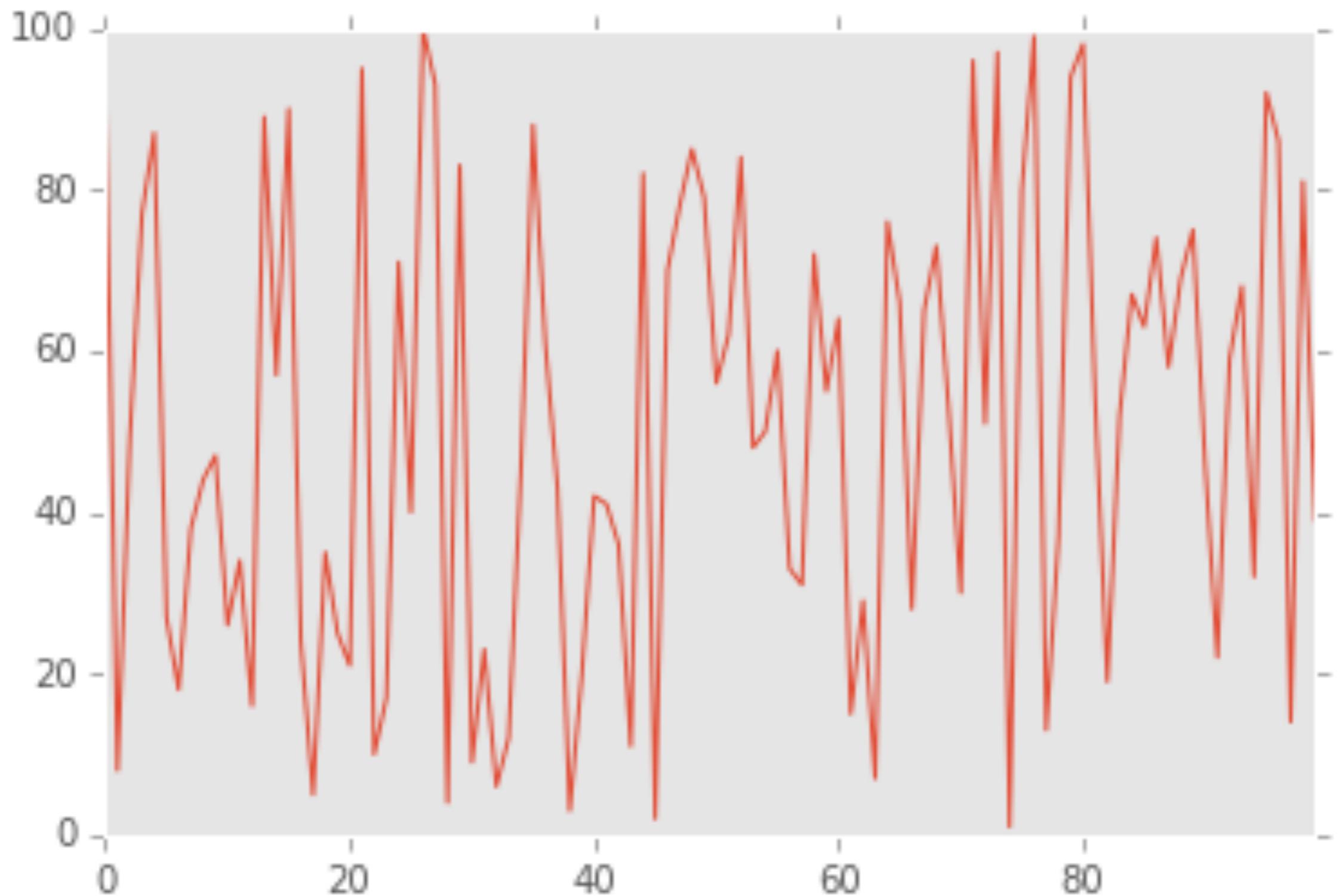
```
mySeries.value_counts()
```

```
mySeries.hist()
```

`mySeries.hist()`



```
mySeries.plot()
```



Are they related?

Are they related?

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

```
series1.corr( Series2 )
```

In Class Exercise

In Series Worksheet 3, Complete exercises 1, 2 and 3.

Exercise 1

`numlist.value_counts()`

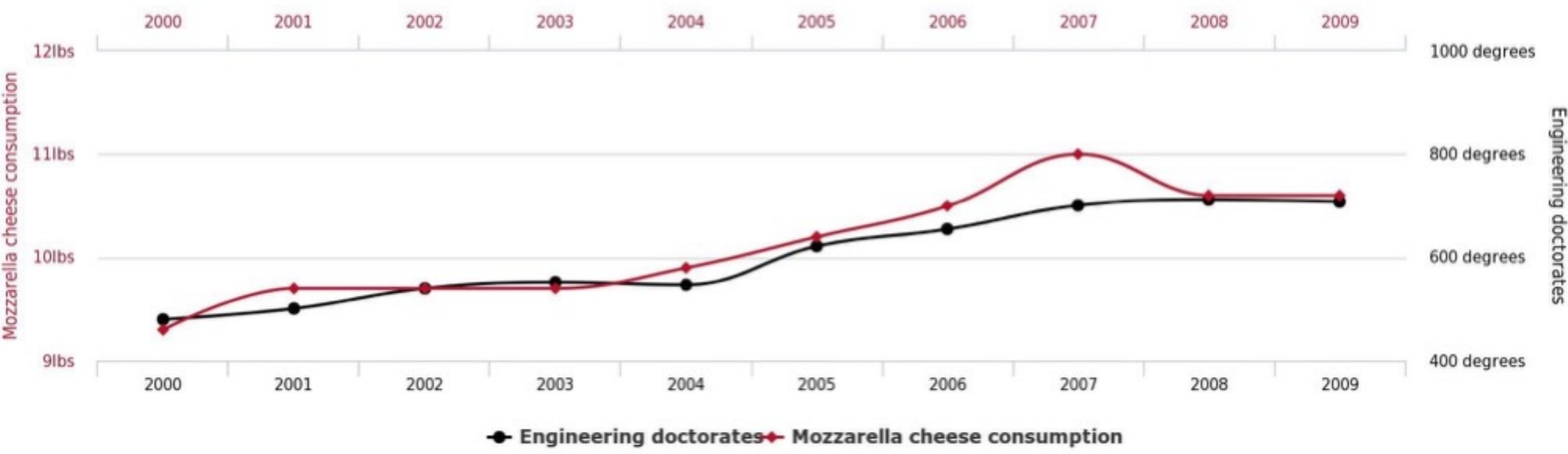
Exercise 2

`numlist.hist()`

Exercise 3

```
score = data1.corr( data2 )  
print( score )
```

Per capita consumption of mozzarella cheese
correlates with
Civil engineering doctorates awarded



correlation != causation

Questions so far?



DATA

A DataFrame can be created from any collection of lists

```
data = pd.DataFrame( <data>, <index>,  
                     <column_names> )
```

CSV

```
data = pd.read_csv( <file> )
```

Excel

```
data = pd.read_excel( <file>,  
                      sheetname=<sheetsname> )
```

```
[  
  {  
    "changed": "2015-04-0300:00:00",  
    "created": "2014-04-10 00:00:00",  
    "dnssec": "False",  
    "expires": "2016-04-10 00:00:00",  
    "isMalicious": 0,  
    "url": "nuteczki.com"  
  },  
  ...  
]
```

JSON

data = pd.read_json(<file>)
or

data = pd.read_json(<url>)

From a Database

```
data = pd.read_sql( <query>, <connection> )
```

HTML

```
data = pd.read_html( <source> )
```

In Class Exercise

Open DataFrame Worksheet 1 and complete Exercises 1 & 2

Exercise 1

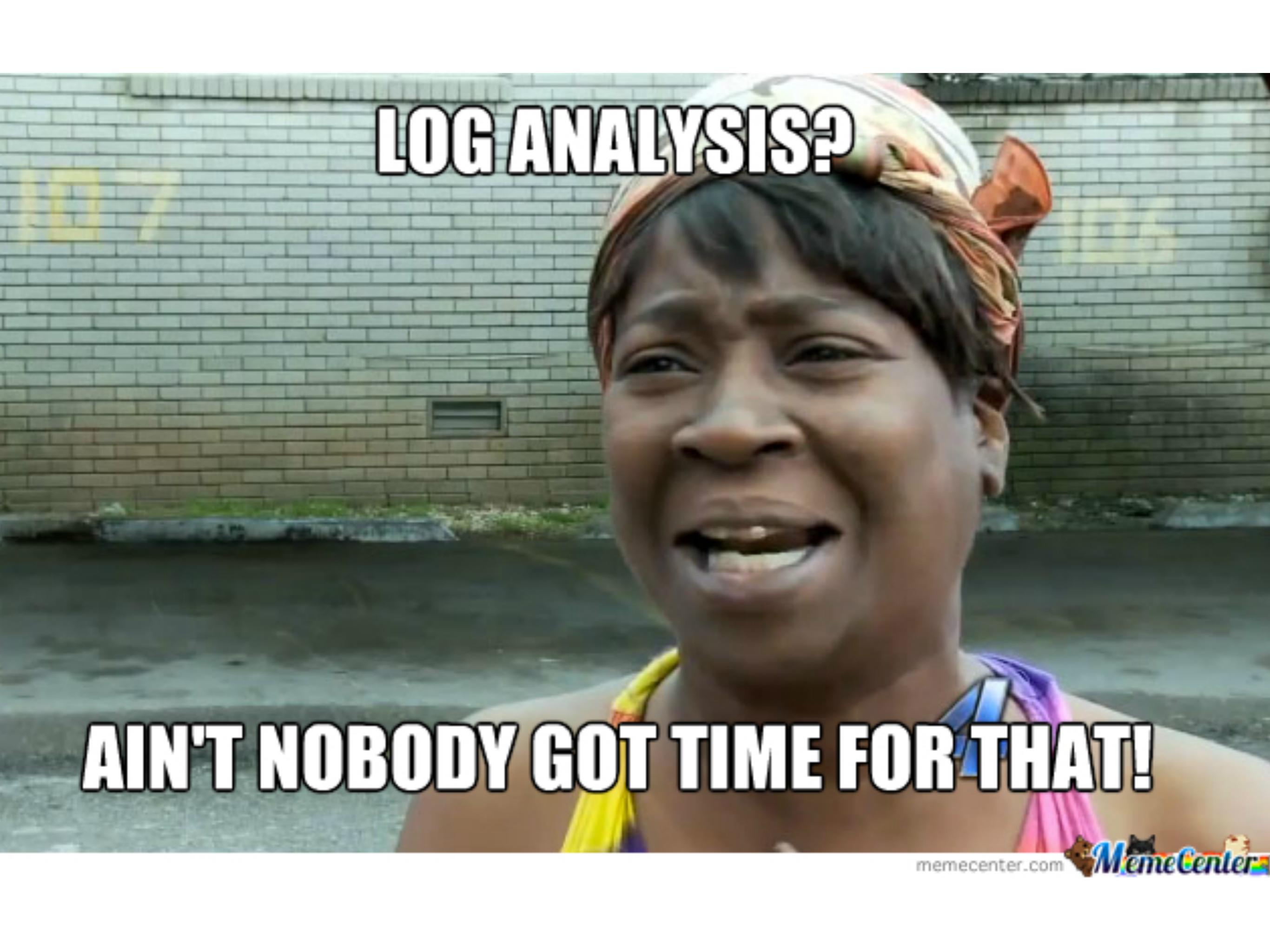
```
twitterData = pd.read_csv(  
    'twitter1.csv',  
    encoding='iso8859_15'  
)
```

Exercise 2

```
terrorData =  
pd.read_excel( 'terrorismData.xlsx' )
```

Log Files

A bit more complicated...



LOG ANALYSIS?

AIN'T NOBODY GOT TIME FOR THAT!

98.233.143.207 - - [30/Sep/2013:11:08:07
-0400] "GET /x.txt HTTP/1.1" 200 20 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X
10_7_4) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/29.0.1547.76 Safari/537.36"

```
logData1 = pd.read_csv( <file>,
    sep=" ",
    quotechar=''''
)
```

```
logData1 = pd.read_table( 'log.txt' , names=[ 'raw' ] )
```

```
logData1 = pd.read_table( 'log.txt', names=[ 'raw' ] )
```

	raw
0	220.181.108.177 - - [30/Sep/2013:08:32:25 -040...
1	46.188.47.240 - - [30/Sep/2013:08:41:34 -0400]...
2	5.10.83.92 - - [30/Sep/2013:08:54:06 -0400] "G...
3	67.222.16.252 - - [30/Sep/2013:09:03:48 -0400]...
4	123.125.71.113 - - [30/Sep/2013:09:03:11 -0400...

```
logData1 = pd.read_table( 'log.txt', names=['raw'] )
logData1['ip'] = logData1['raw'].str.extract( '(\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3})' )
```

	raw	ip
0	220.181.108.177 -- [30/Sep/2013:08:32:25 -040...	220.181.108.177
1	46.188.47.240 -- [30/Sep/2013:08:41:34 -0400]...	46.188.47.240
2	5.10.83.92 -- [30/Sep/2013:08:54:06 -0400] "G...	5.10.83.92
3	67.222.16.252 -- [30/Sep/2013:09:03:48 -0400]...	67.222.16.252
4	123.125.71.113 -- [30/Sep/2013:09:03:11 -0400...	123.125.71.113

```
pattern = r'(\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3})\s(\S+)\s(\S+)\s
\[ (\d{2}/\w+?/\d{4}:\d{2}:\d{2}:\d{2}\s-\d{4})\]\s"([A-Z]+)\s(\S
+?)\s(\S+)\s(\d{3})\s(\S+?)\s"(\S+?)"\s"(.+?)"'

columnNames = [ '', 'ipAddress', 'remoteLogname', 'username',
'dateTime', 'request', 'requestLine' , '', 'responseCode',
'bytesSent', 'referrer', 'userAgent' ]

data = read_from_regex( r=pattern, input='apache-logs.txt',
columns=columnNames )
```

```
data = read_from_regex( r=pattern,
input='apache-logs.txt',
columns=columnNames )
```

	dateTime	ipAddress	request	requestLine	responseCode	username
0	30/Sep/2013:08:32:25 -0400	220.181.108.177	GET	/printers/view-all-products.html	404	-
1	30/Sep/2013:08:41:34 -0400	46.188.47.240	GET	/x.txt	200	-
2	30/Sep/2013:08:54:06 -0400	5.10.83.92	GET	/other/view-all-products.html	404	-
3	30/Sep/2013:09:03:48 -0400	67.222.16.252	POST	/index.php	404	-
4	30/Sep/2013:09:03:11 -0400	123.125.71.113	GET	/	200	-

```
import apache_log_parser
```

pip install apache-log-parser if not installed

Determining Log File Format

- cd /etc/apache2/
- \$ grep CustomLog sites-enabled/foo.nowhere.org
- CustomLog /var/log/apache2/foo.nowhere.org-access.log combined
- \$ grep combined apache2.conf
- LogFormat "**%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i**" combined
- line_parser = apache_log_parser.make_parser("%v %h %l %u %t \"%r\" %>s %b")

Define Variables

APACHE_COMBINED= "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""

APACHE_COMMON= "%h %l %u %t \"%r\" %>s %b"

Logging

This module defines functions and classes which implement a flexible event logging system for applications and libraries.

Example

```
FORMAT = '%(asctime)-15s %(clientip)s %(user)-8s %(message)s'
logging.basicConfig(format=FORMAT)
d = {'clientip': '192.168.0.1', 'user': 'fbloggs'}
logger = logging.getLogger('tcpserver')
logger.warning('Protocol problem: %s', 'connection reset', extra=d)
```

would print something like

```
2006-02-08 22:20:02,165 192.168.0.1 fbloggs Protocol problem: connection reset
```

Glob Module

The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell.

```
>>> import glob  
>>> glob.glob('*gif')  
['card.gif']  
>>> glob.glob('.c*')  
['.card.gif']
```

Reading in the Logs

```
import glob
import logging

def gulp(log_file_path, pattern=APACHE_COMBINED):
    """ import and parse log files """
    log_data=[]
    line_parser=apache_log_parser.make_parser(pattern)
    for file_name in glob.glob(log_file_path):
        logging.info("file_name: %s" % file_name)
        file = open(file_name, 'r')
        lines = file.readlines()
        file.close()
        logging.info(" read %s lines" % len(lines))
        for line in lines:
            line_data=line_parser(line)
            log_data.append(line_data)
    logging.info("total number of events parsed: %s" % len(log_data))
    return log_data
```

In [8]:

```
logging.basicConfig(level=logging.INFO)
log_data = gulp("*.*log")
```

Verifying Data

```
print(log_data[0])
```

Displaying first record

Out[9]:

```
{'remote_host': '157.55.39.90',
 'remote_logname': '-',
 'remote_user': '-',
 'request_first_line': 'GET /robots.txt HTTP/1.1',
 'request_header_referer': '-',
 'request_header_user_agent': 'Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)',
 'request_header_user_agent__browser_family': 'bingbot',
 'request_header_user_agent__browser_version_string': '2',
 'request_header_user_agent__is_mobile': False,
 'request_header_user_agent__os_family': 'Other',
 'request_header_user_agent__os_version_string': '',
 'request_http_ver': '1.1',
 'request_method': 'GET',
 'request_url': '/robots.txt',
 'response_bytes_clf': '37',
 'status': '200',
 'time_received': '[12/May/2015:17:04:15 -0700]',
 'time_received_datetimeobj': datetime.datetime(2015, 5, 12, 17, 4, 15),
 'time_received_isofromat': '2015-05-12T17:04:15',
 'time_received_tz_datetimeobj': datetime.datetime(2015, 5, 12, 17, 4, 15, tzinfo='0700'),
 'time_received_tz_isofromat': '2015-05-12T17:04:15-07:00',
 'time_received_utc_datetimeobj': datetime.datetime(2015, 5, 13, 0, 4, 15, tzinfo='0000'),
 'time_received_utc_isofromat': '2015-05-13T00:04:15+00:00'}
```

Creating the DataFrame

- import pandas as pd
- log_dataframe = pd.DataFrame(log_data)

Between Time

```
DataFrame.between_time(start_time, end_time,  
include_start=True, include_end=True)
```

Select values between particular times of the day (e.g., 9:00-9:30 AM)

Example

```
#create new dataframe with date indexed  
apache_on_time = apache_df.set_index(log_data['time_received_datetimeobj'])
```

GOAL

Select Logs Between June 12th 2015, 11:30PM & 5:00AM the next morning

SOLUTION

```
apache_logs_between_time = apache_on_time.between_time('2015-05-12 23:30:00','2015-05-13 05:00:00')
```

→ **5952/14534** records selected from dataframe

In Class Exercise

Open DataFrame Worksheet 1 and complete Exercise 3

```
users = twitterData[ 'userName' ]
```

```
usersAndFriends =  
twitterData[ [ 'username', 'friends' ] ]
```

```
tweetCounts =  
twitterData[ 'Username' ].value_counts()  
  
tweetCounts.head(10)
```

```
twitterSummary =  
twitterData[['Username', 'Friends', 'Followers']]
```

	Username	Friends	Followers
0	_prettybrown	1042	1538
1	CarlyManning24	278	304
2	madzLuvzLakers	619	1039
3	_AyyJayy	203	204
4	Akeemoneale	165	27

`data.loc[<index>]`

`data.loc[<list of indexes>]`

`data.sample(<n>)`

1	2	3
4	5	6
7	8	9

data.T

1	4	7
2	5	8
3	6	9

```
rowAvg = [ ]
for x, row in studentData.iterrows():
    rowAvg.append( ( row[ 'Test1' ] +
row[ 'Test2' ] + row[ 'Test3' ] + row[ 'Test4' ] +
row[ 'Test5' ] ) / 5 )
studentData[ 'average2' ] = rowAvg
```

#Gets you the sum of columns
data.sum(axis=0)

#Gets you the sum of the rows
data.sum(**axis=1**)

In Class Exercise

Open DataFrame Worksheet 1 and complete Exercise 4 & 5
and Worksheet 2, Exercise 1

Worksheet 1, Exercise 4

```
tweetCounts = twitterData['Username'].value_counts()  
tweetCounts.head(10)
```

Worksheet 1, Exercise 5

```
twitterSummary = twitterData[['Username', 'Friends', 'Followers']]  
twitterSummary['ffratio'] = twitterSummary['Friends'] / twitterSummary['Followers']  
  
twitterSummary.head()
```

Worksheet 2, Exercise 1

```
studentData = pd.read_csv('..../Data/studentData.csv')

studentData[ 'average' ] =
    studentData[ ['Test1', 'Test2', 'Test3', 'Test4', 'Test5'] ].mean( axis=1 )

studentData.sort( columns='average', ascending=False )
```

data[<logical condition>]

```
twitterData[ 'Friends' ] > 5
```

0	True
1	True
2	True
3	True
4	True
5	False
6	True
7	True

```
twitterData[ twitterData['Friends'] > 5 ]
```

```
Series.dropna()  
Series.fillna(value="<something>")
```

In Class Exercise

Open DataFrame Worksheet 1 and complete Exercise 6

Worksheet 1, Exercise 6

```
newData =  
twitterData[  
    twitterData['URL'].fillna("").str.contains('facebook')  
]  
  
newData['FacebookID'] =  
    newData['URL'].str.extract('profile.php\?id=(\d+)')  
  
newData.dropna(inplace=True)
```

Applying a Function

```
data.apply( <function> )
```

Rule #1 of Web Security

Don't let Google index your web logs.



LEGENDARY FAIL

For when an Epic Fail wasn't enough

[https://www.google.com/
search?q=inurl%3Aaccess.log
+filetype%3Alog&gws_rd=ssl](https://www.google.com/search?q=inurl%3Aaccess.log+filetype%3Alog&gws_rd=ssl)

DataFrame Challenge 1

Scenario: You have an Apache Web Server log—called `apache-log-sample2.txt`—which can be found in the `/Data/` folder of your course materials.

Using this data, and what you have learned thus far, answer the following questions:

1. How many different bots have queried this site and which one has hit the site the most?
2. What is the breakdown of mobile users vs. non-mobile users?
3. What is the breakdown of browser usage?

```
from user_agents import parse
ua_string = 'Mozilla/5.0 (iPhone; CPU iPhone OS 5_1 like Mac OS X)
AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9B179 Safari/
7534.48.3'
user_agent = parse(ua_string)

# Accessing user agent's browser attributes
user_agent.browser # returns Browser(family=u'Mobile Safari', version=(5,
1), version_string='5.1')
user_agent.browser.family # returns 'Mobile Safari'
user_agent.browser.version # returns (5, 1)
user_agent.browser.version_string # returns '5.1'

# Accessing user agent's operating system properties
user_agent.os # returns OperatingSystem(family=u'iOS', version=(5, 1),
version_string='5.1')
user_agent.os.family # returns 'iOS'
user_agent.os.version # returns (5, 1)
user_agent.os.version_string # returns '5.1'
```

The module also includes:

- **is_pc**: Returns True if the user agent is a PC
- **is_mobile**: Returns True if the user agent is a mobile phone
- **is_tablet**: Returns True if the user agent is a tablet device
- **is_touch_capable**: Returns True if the user agent has touch capabilities
- **is_bot**: Returns true if the user agent is a search engine crawler

day1blackhat

Merging Data Sets

Union

Union

Series 1

Index	Value
0	6
1	4
2	2
3	3

Union

Series 1

Index	Value
0	6
1	4
2	2
3	3

Series 2

Index	Value
0	7
1	5
2	3
3	4

Union

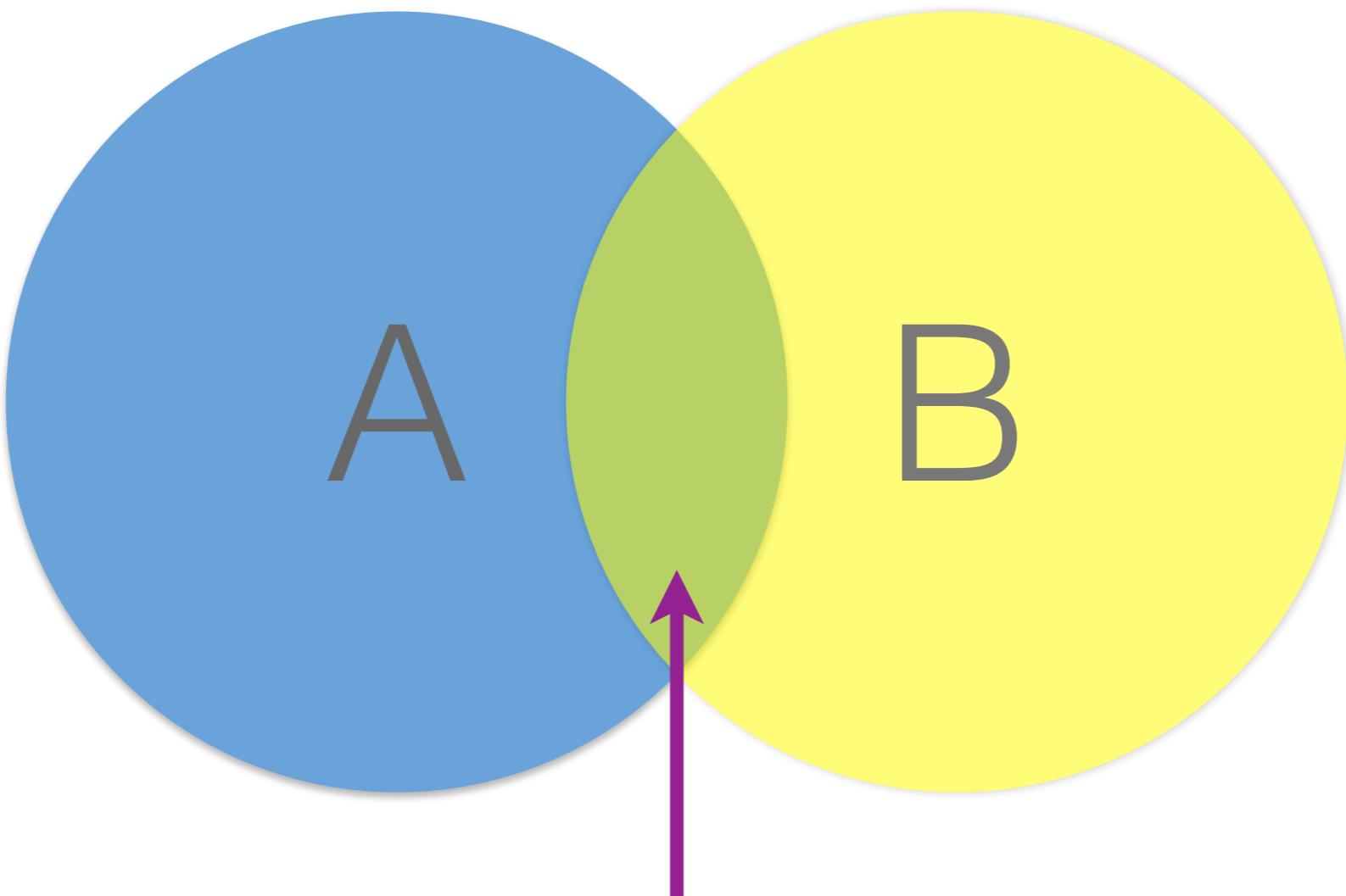
combinedSeries

Index	Value
0	6
1	4
2	2
3	3
4	7
5	5
6	3
7	4

```
combinedSeries = pd.concat( [series1, series2], ignore_index=True )
```

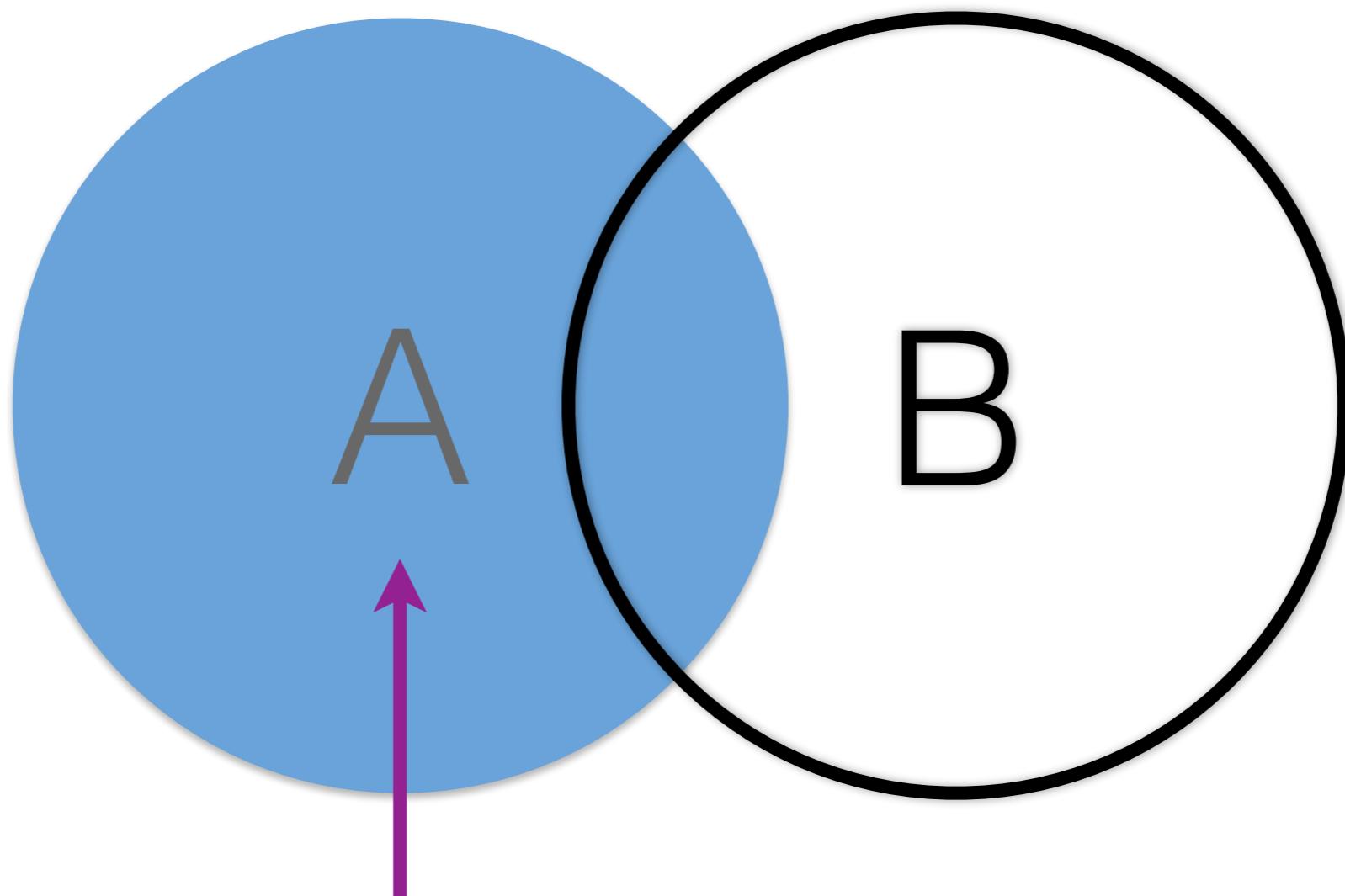
Joins

Inner Join (Intersection)



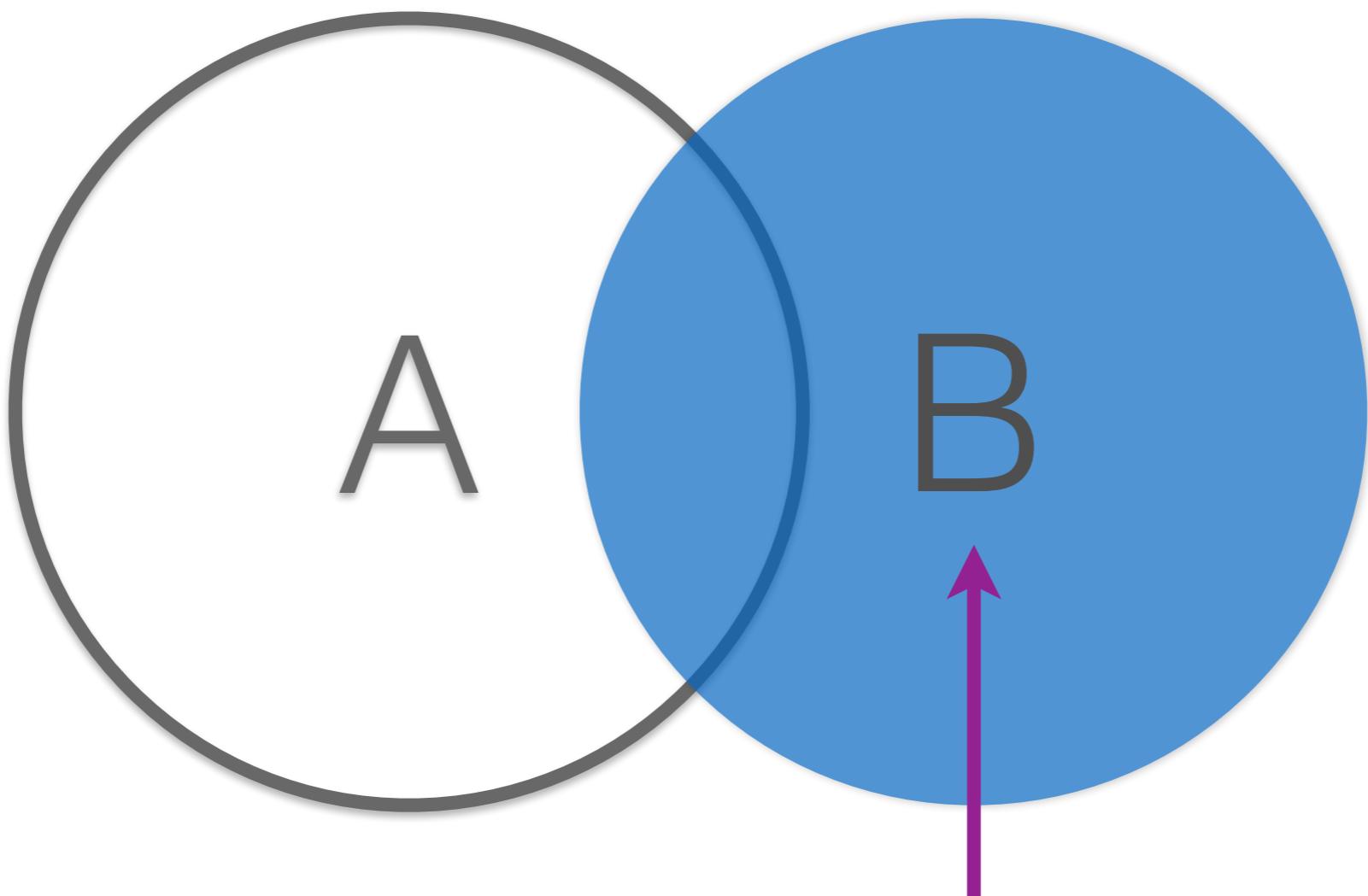
Inner Join

Left Join



Left Join

Right Join



Right Join

```
pd.merge( leftData, rightData )
```

```
pd.merge( leftData, rightData,  
         how=<join type> )
```

```
pd.merge( leftData, rightData,  
         how=<join type> )
```

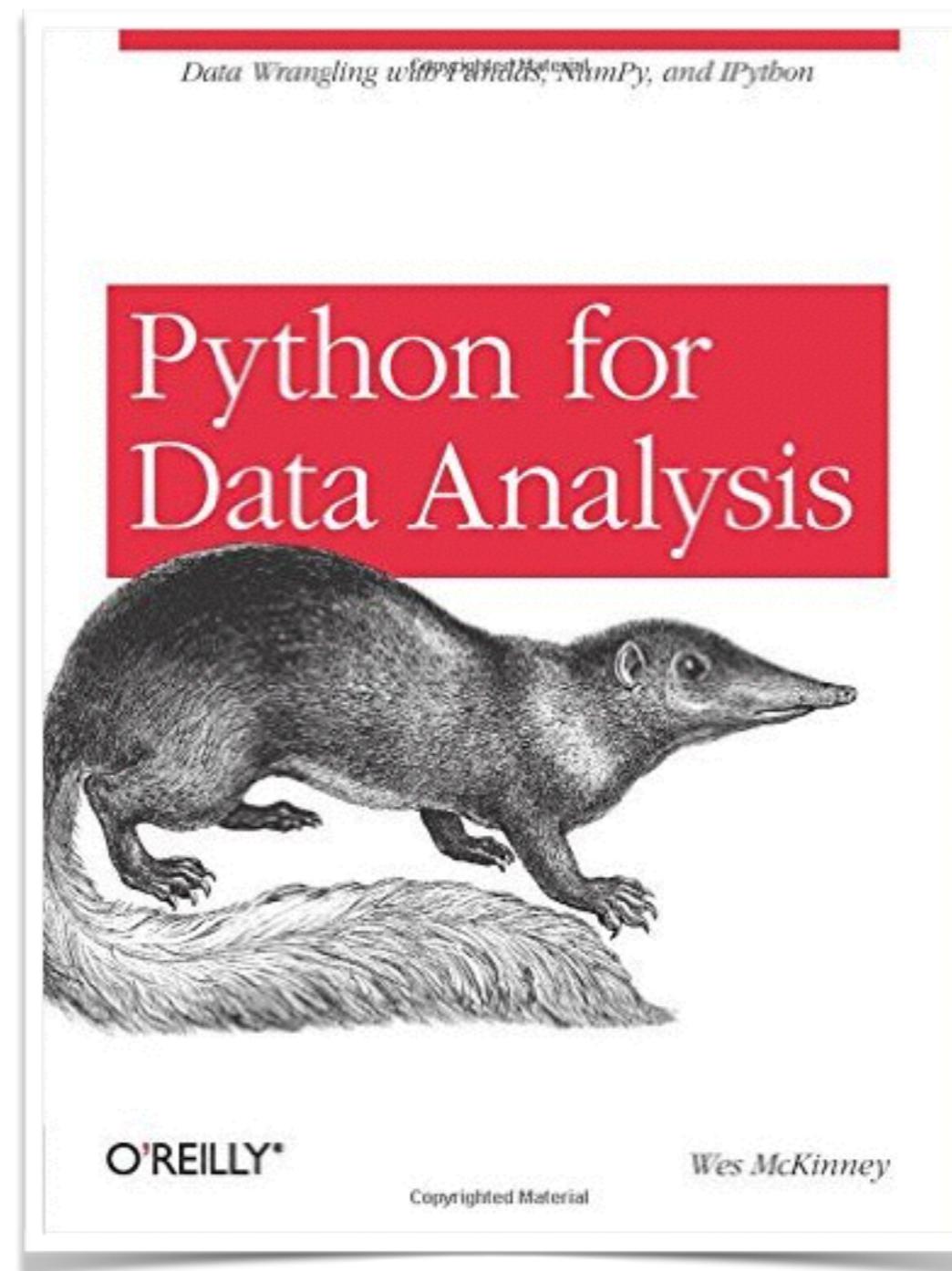
Option	SQL Equivalent	Description
inner	INNER JOIN	Intersection
left	LEFT OUTER JOIN	Returns items in Set A, but not in Set B
right	RIGHT OUTER JOIN	Returns items in Set B, but not in Set A
outer	FULL OUTER JOIN	Returns the union of both sets

```
pd.merge( leftData, rightData,  
         how=<join type>,  
on=<field list> )
```

In Class Exercise

Complete DataFrame Worksheet - Building the Dataset

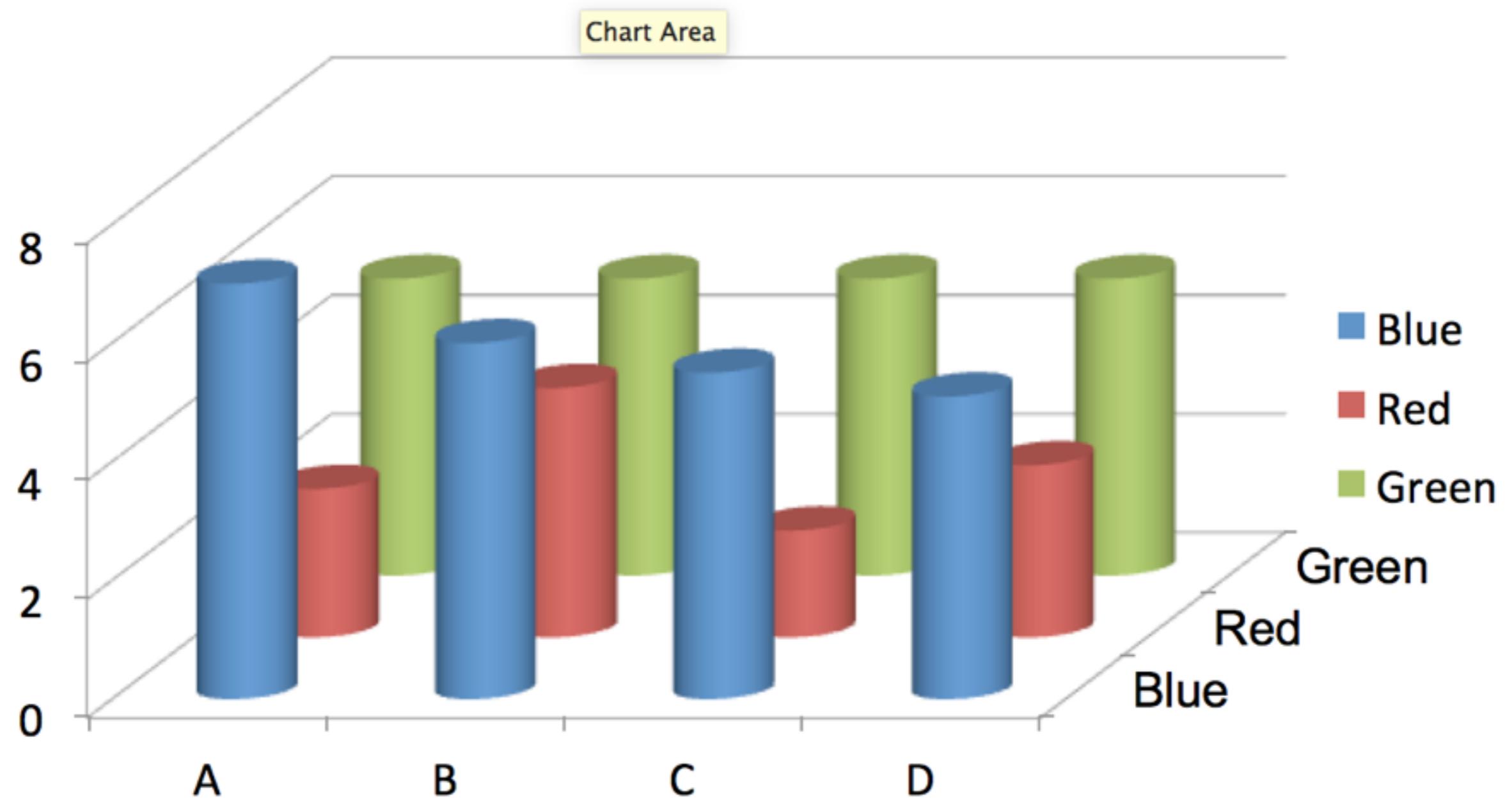
Recommended Reading



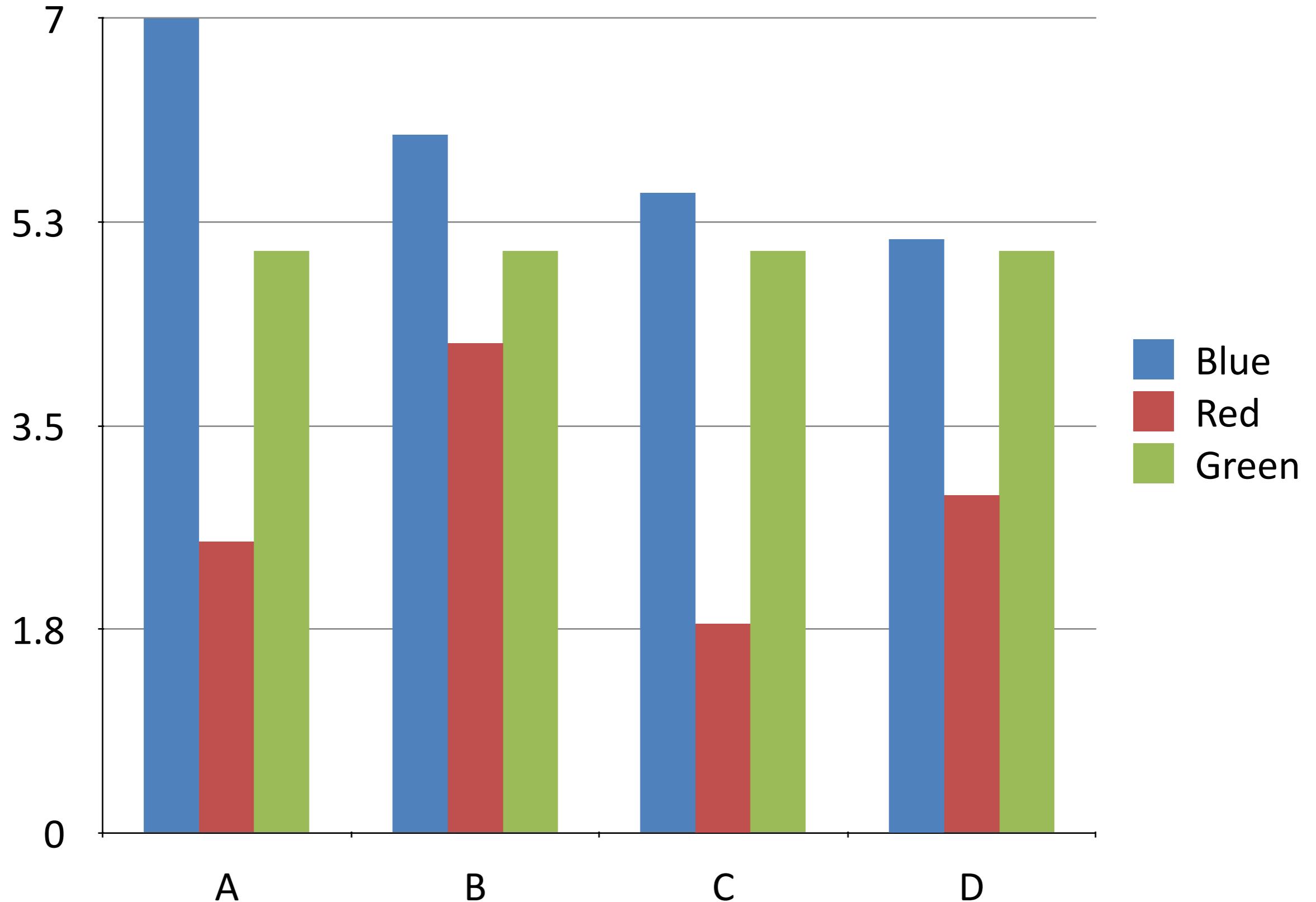
Visualizing Data

Graph Challenge

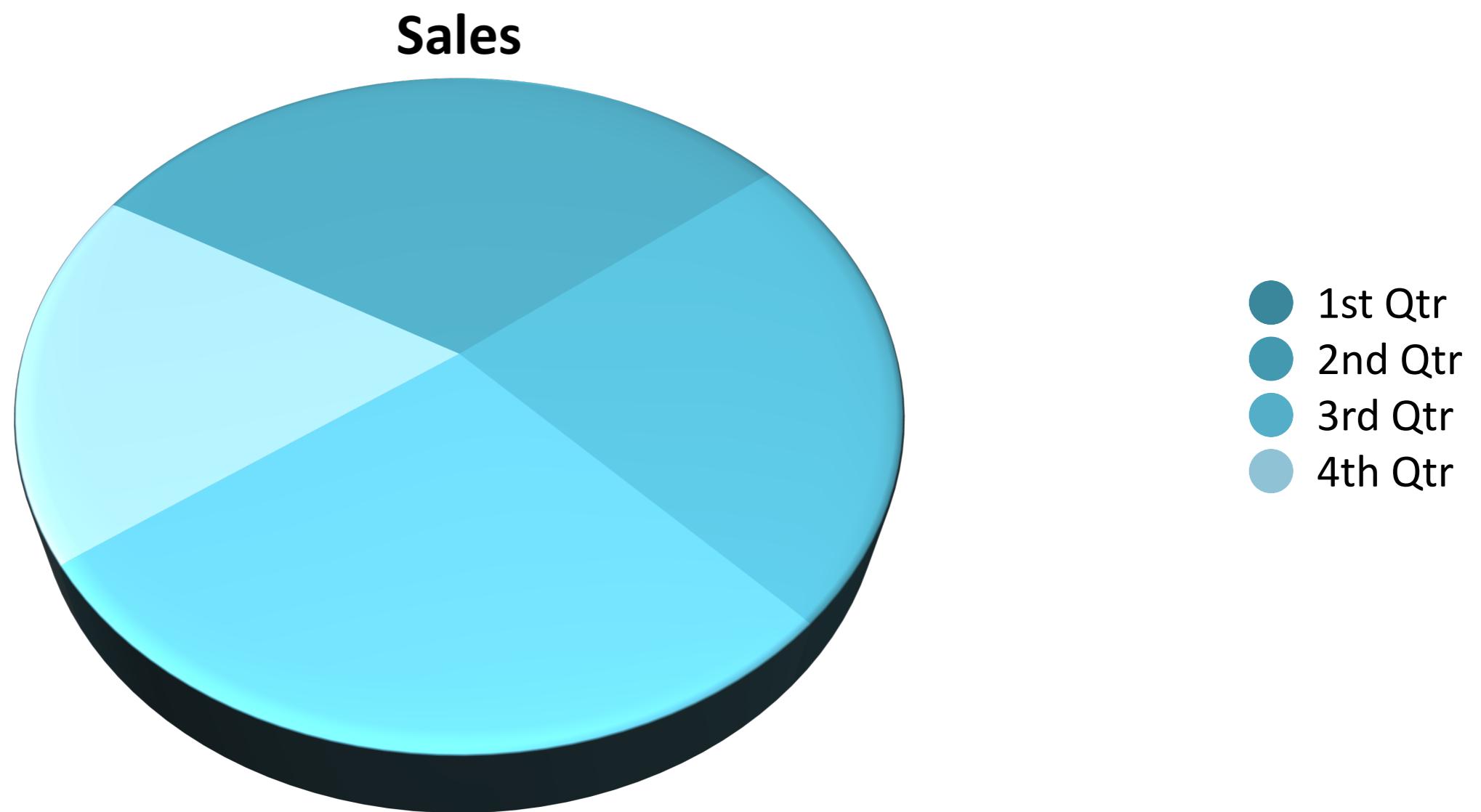
Questions: What is the height of the green columns? For which categories (A,B,C,D) are the blue columns taller than the green columns?



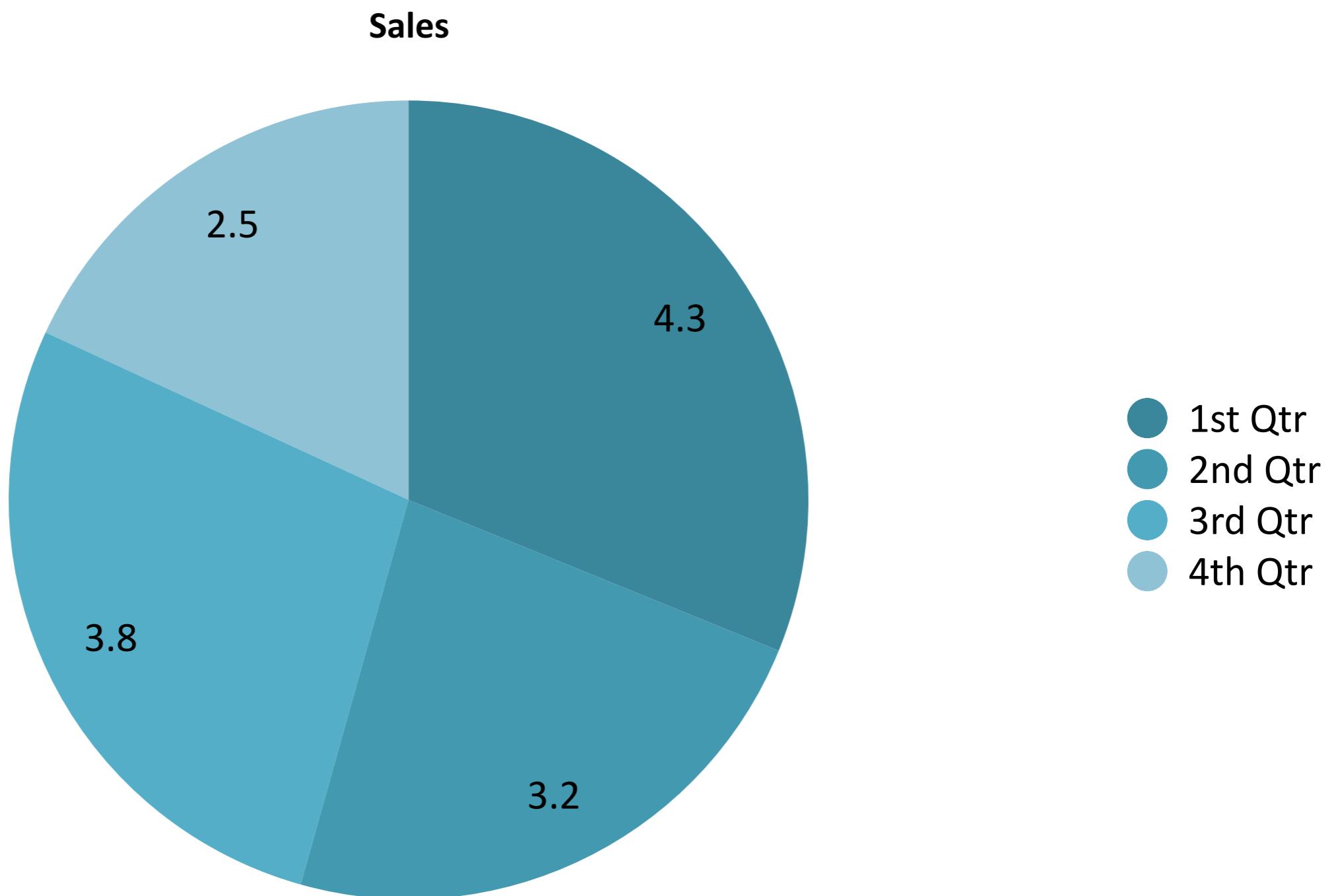
Graph Challenge



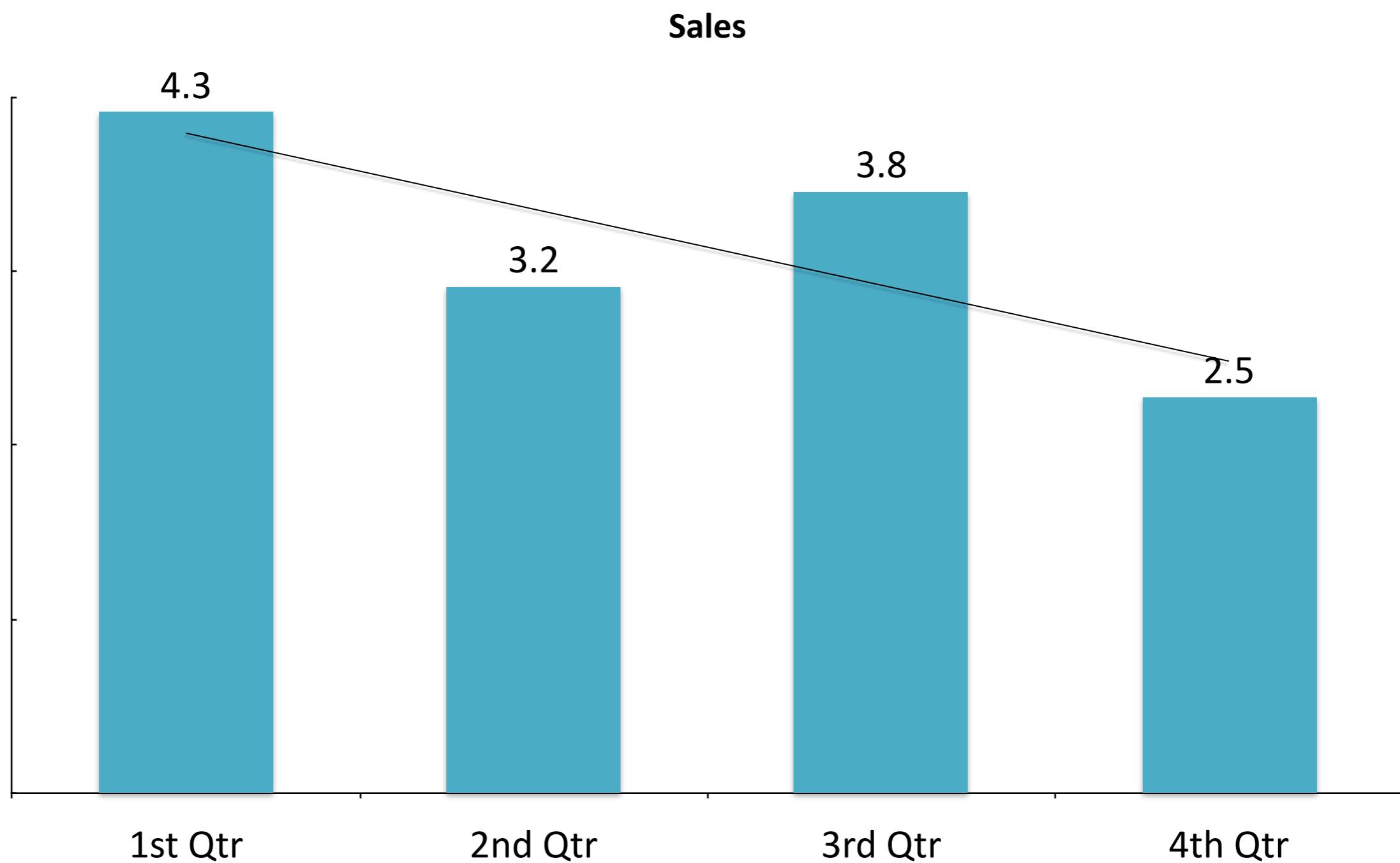
What's wrong with this?



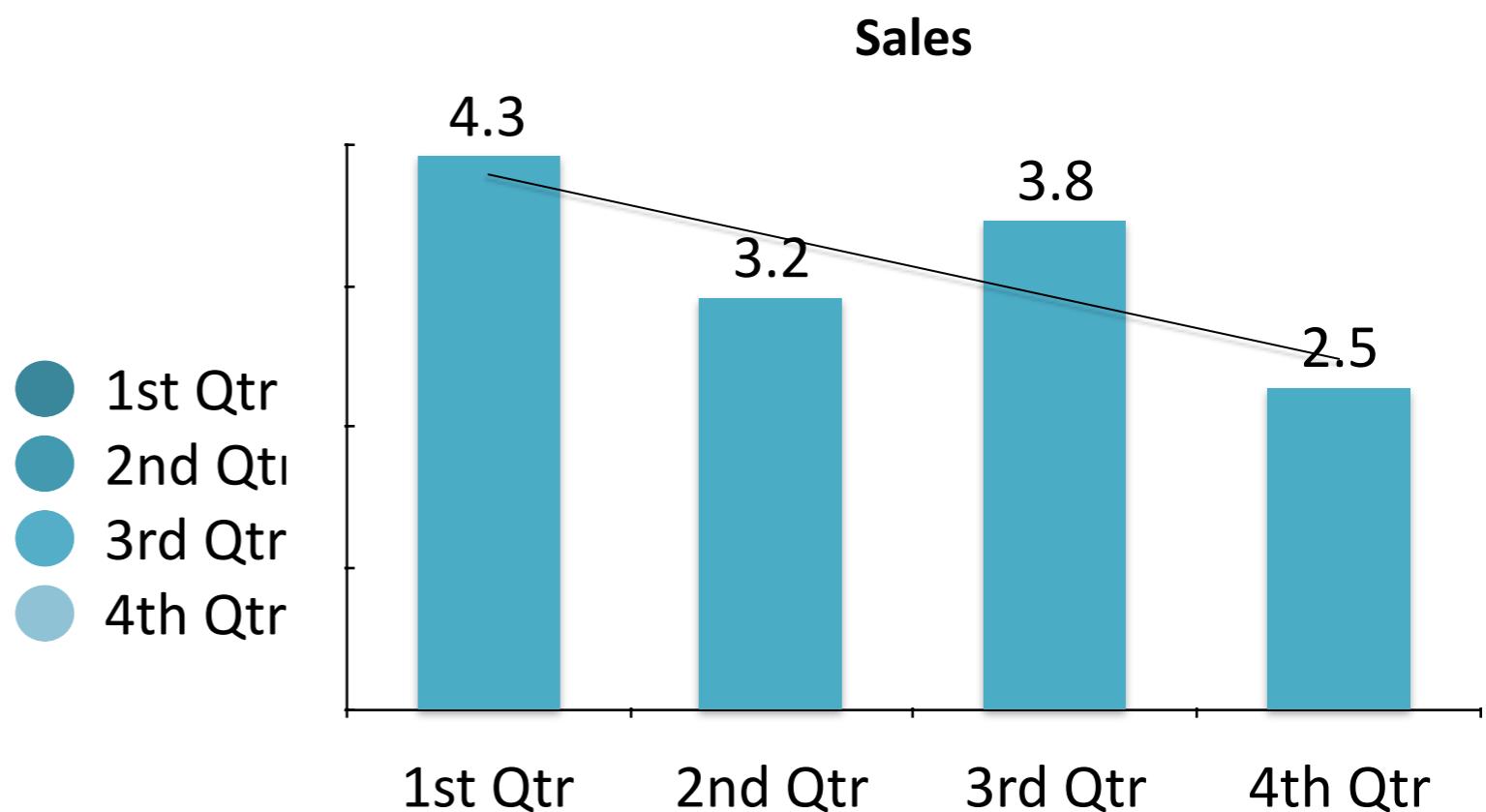
What about this?



How about this?



Why is the Bar Chart Better?



Exploratory Visualizations

Explanatory Visualizations

Why Visualize Data?

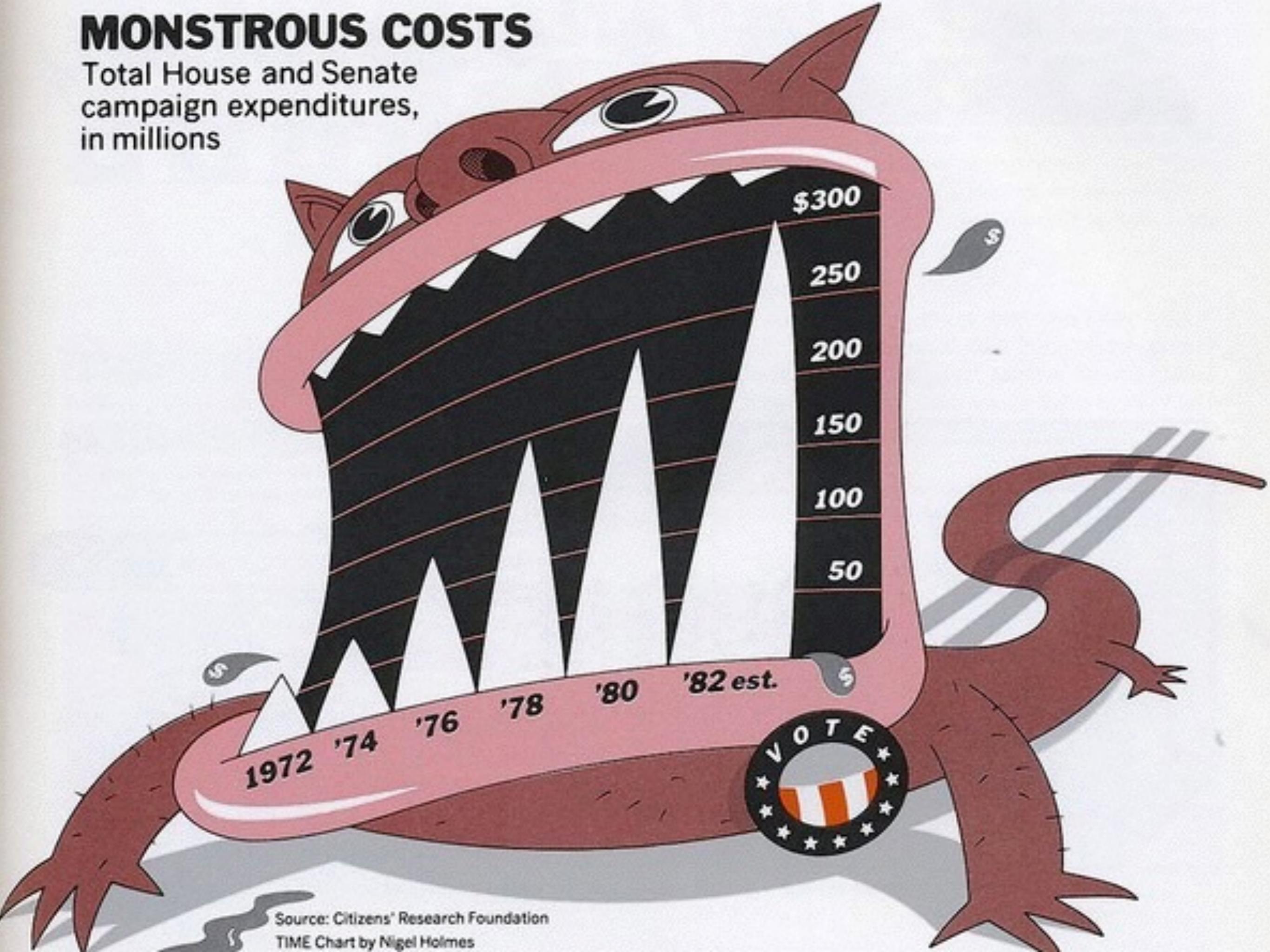
Visualizing data can inspire you to ask new and more refined questions of your data and ultimately lead to better analysis.

The power of visualization comes from illustrating relationships, contrasts and comparisons between many different dimensions of data.

Remove
to improve
(the **data-ink** ratio)

MONSTROUS COSTS

Total House and Senate campaign expenditures, in millions



Source: Citizens' Research Foundation
TIME Chart by Nigel Holmes

Show Comparisons,
Contrasts and
Differences

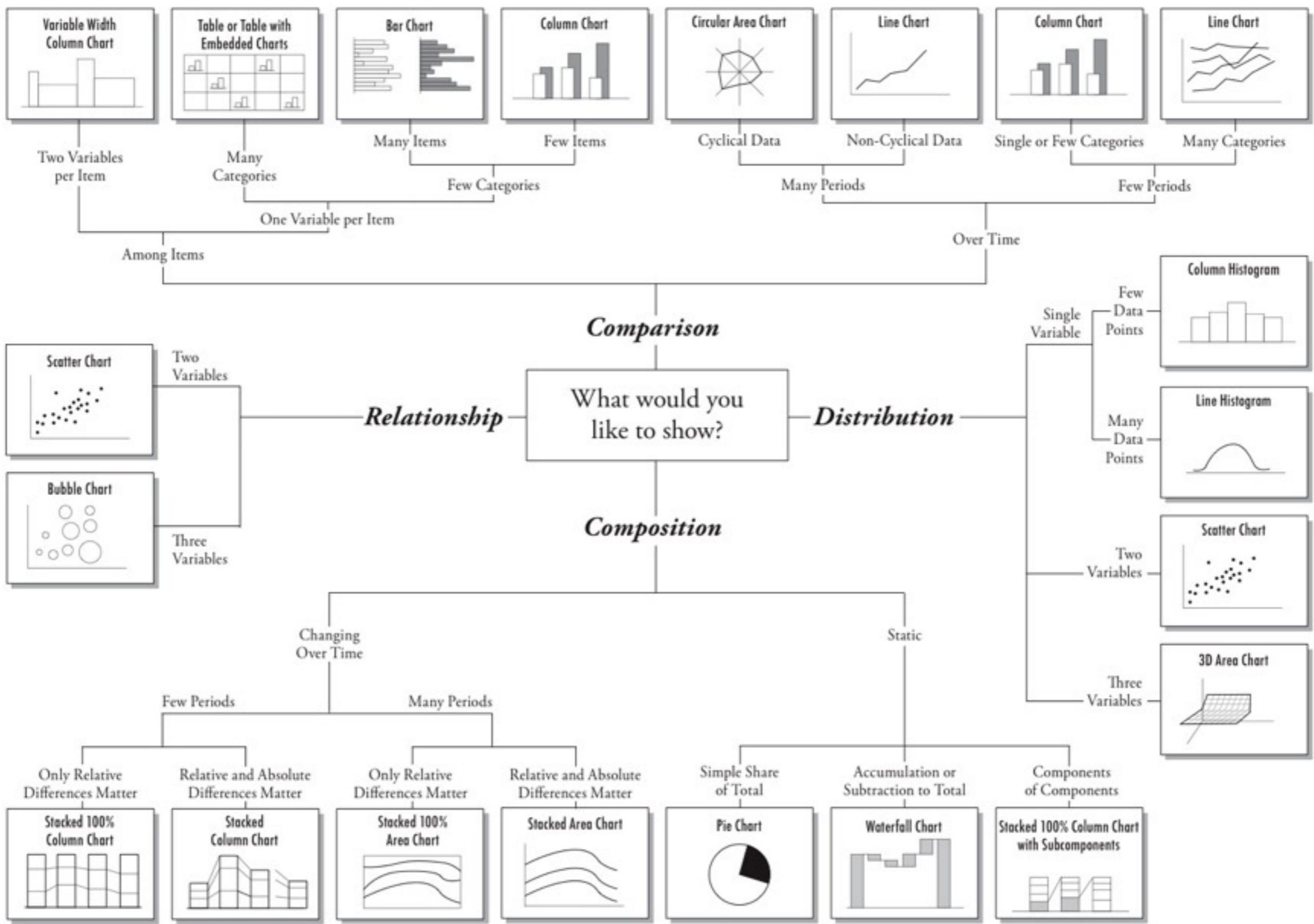
Show Multivariate
Data

Integrate words,
numbers, images and
diagrams

Document your
Evidence

Ultimately, the quality, relevance and integrity of the content is most important.

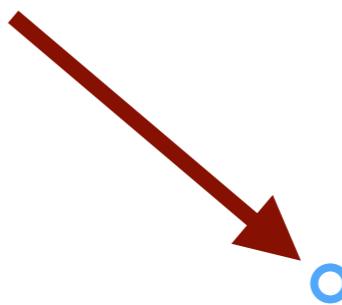
Chart Suggestions—A Thought-Starter



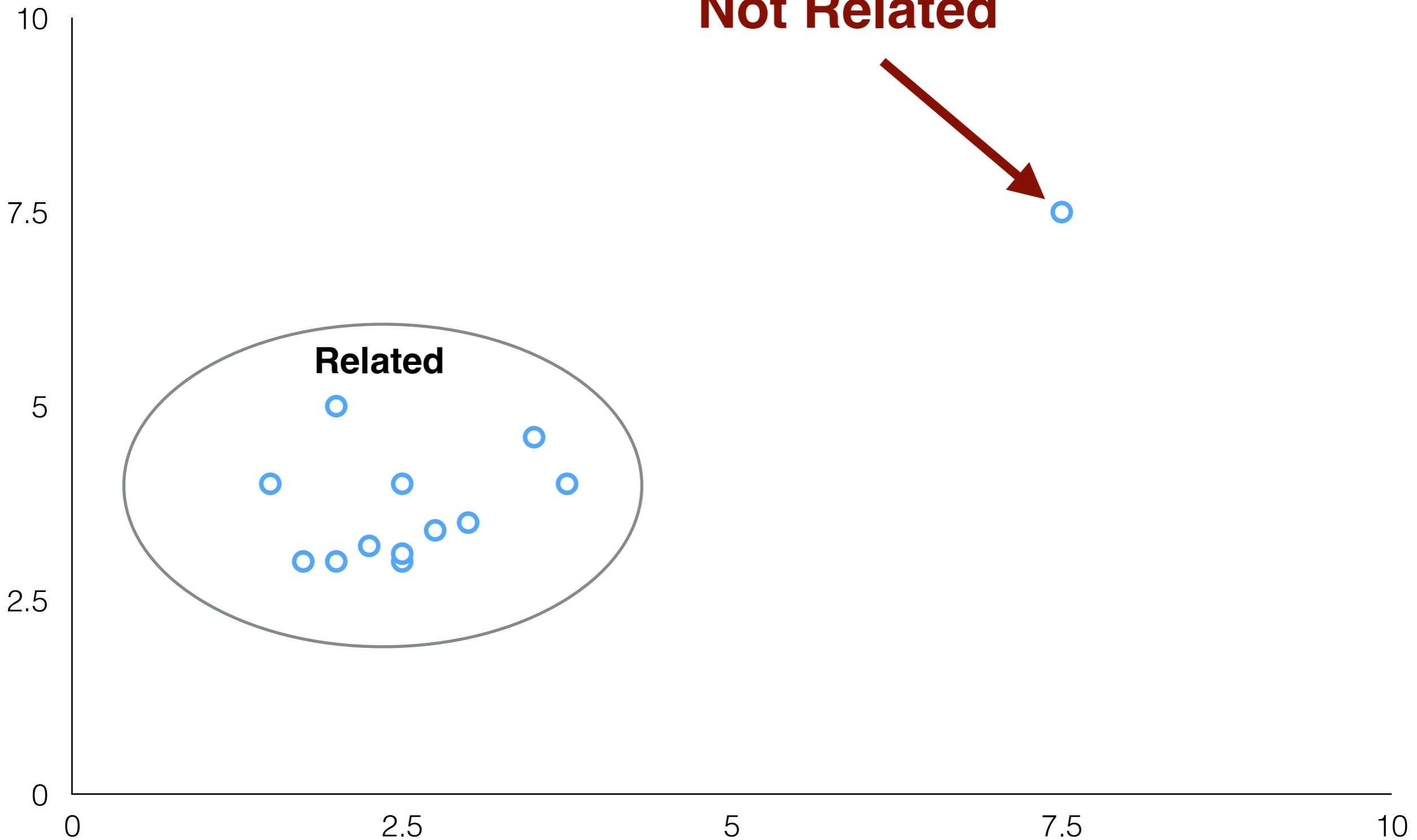
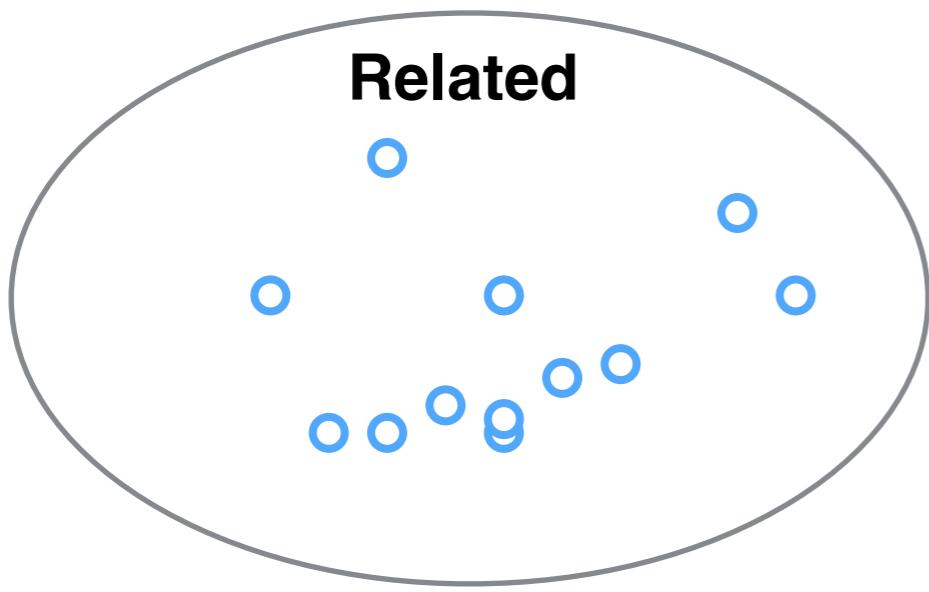
Visual Encodings

Visual Encodings: **Position**

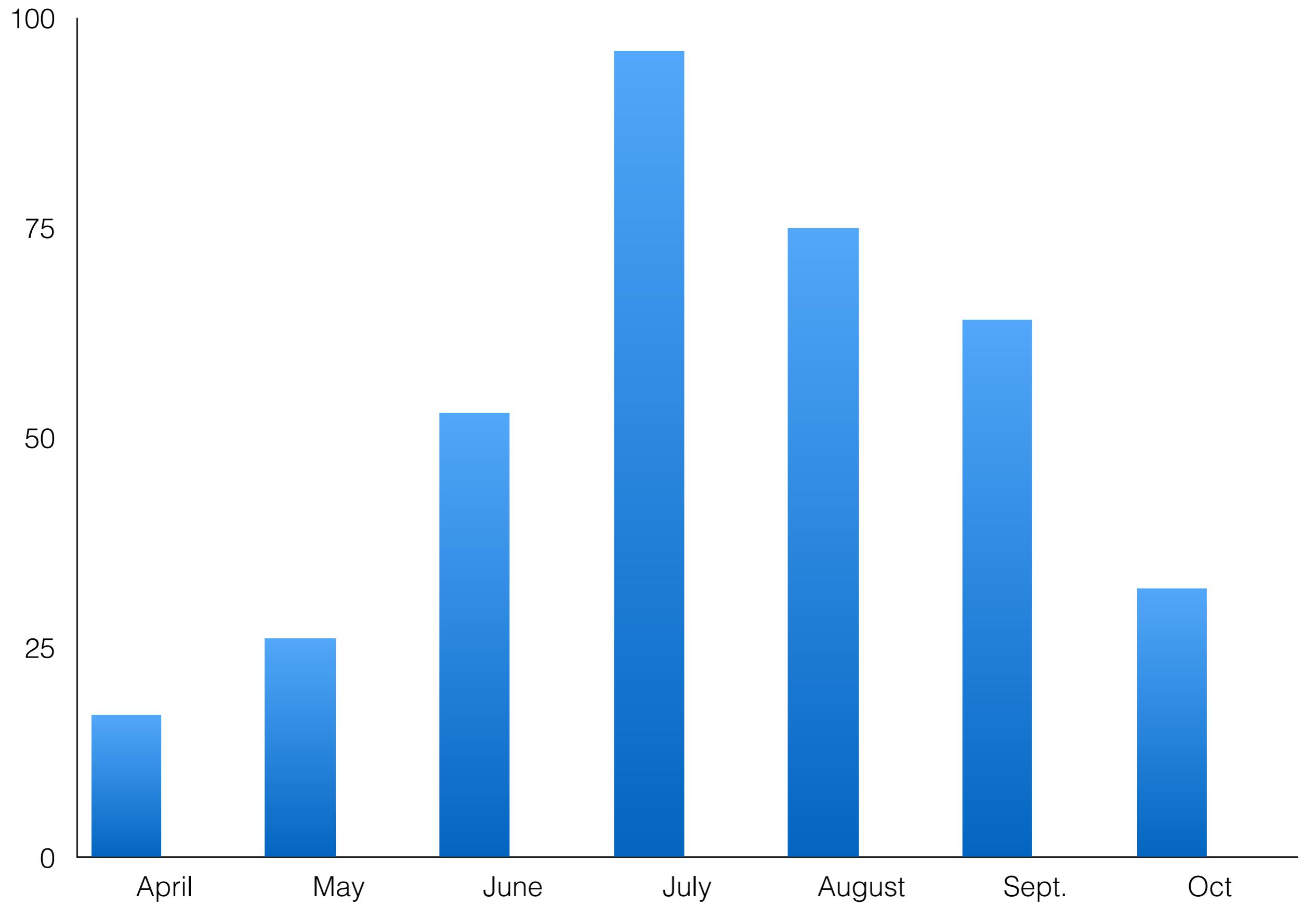
Not Related



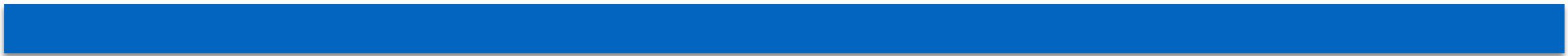
Related



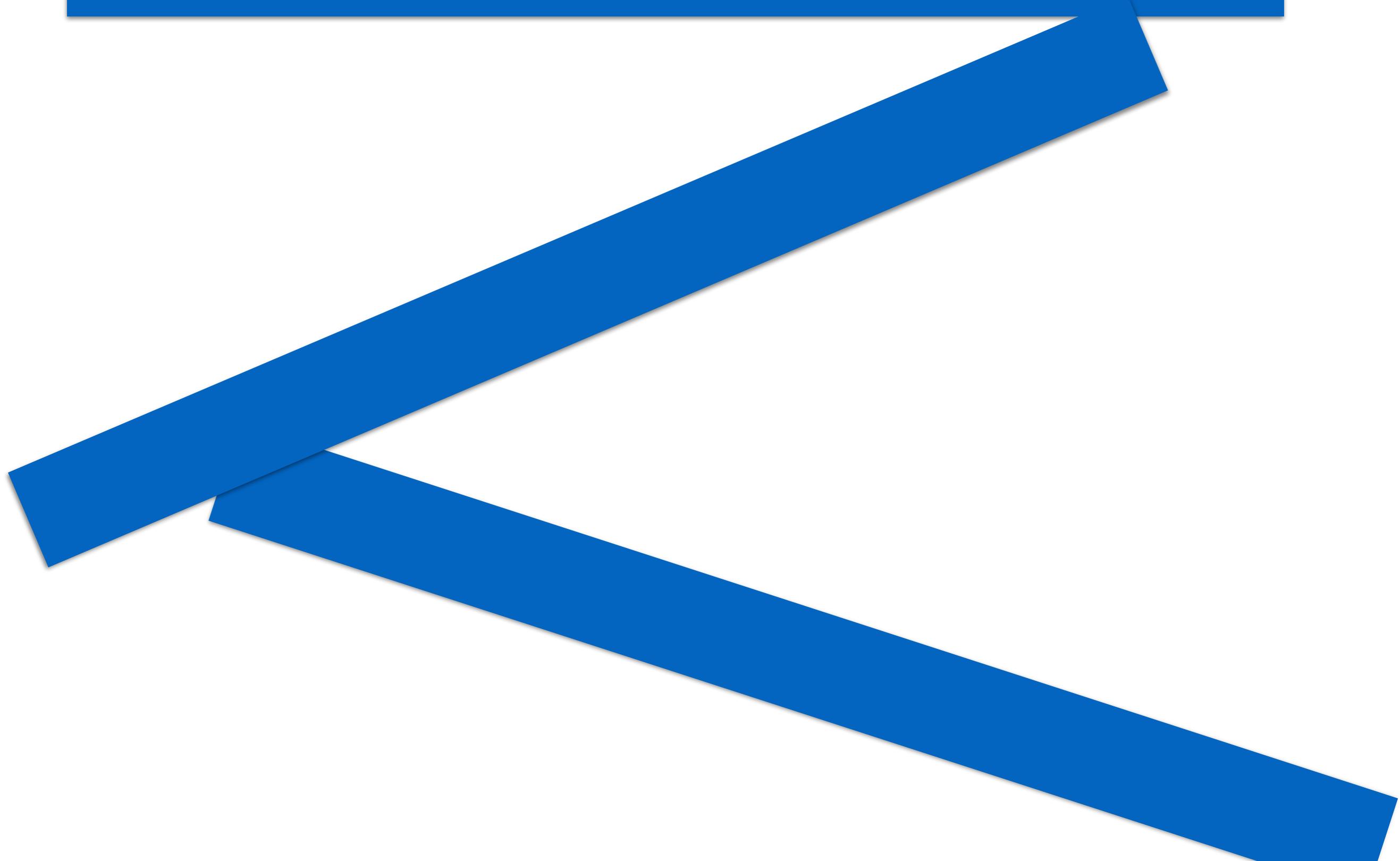
Visual Encodings: **Length**

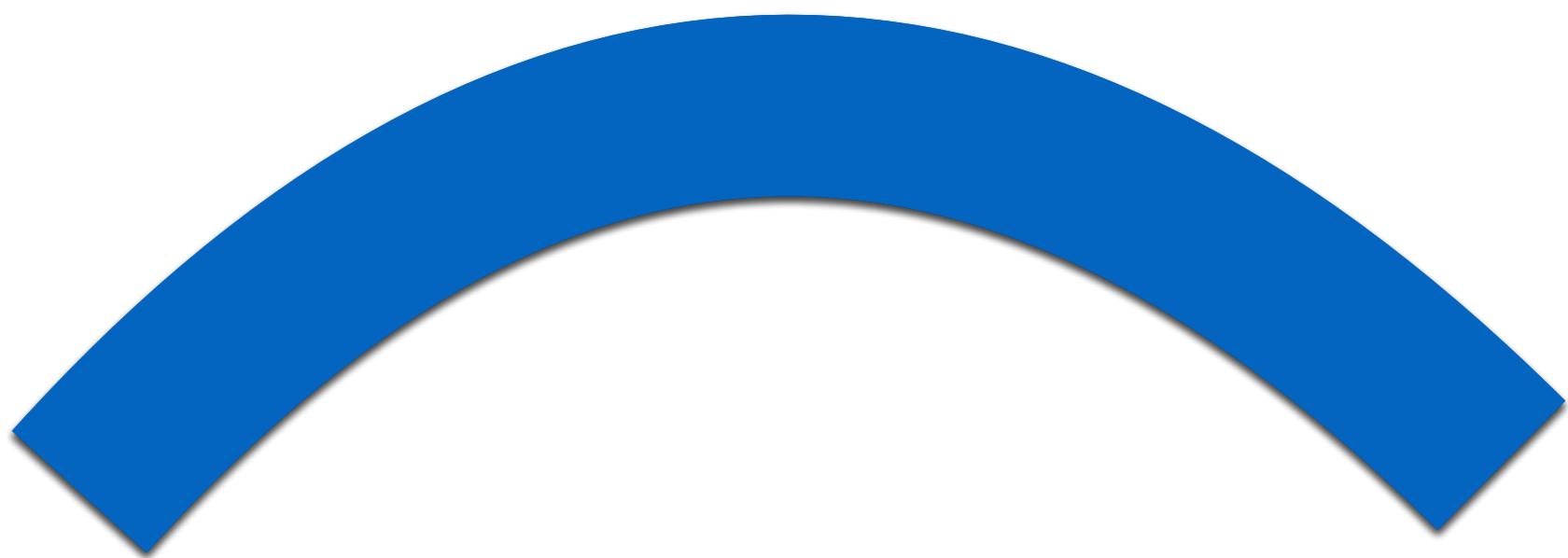




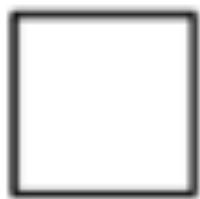








Visual Encodings: **Size**



Length 1
Area 1



Length 1.4
Area 2



Length 2
Area 2



Length 2
Area 4



Length 4
Area 4



Length 3
Area 9



Length 9
Area 9

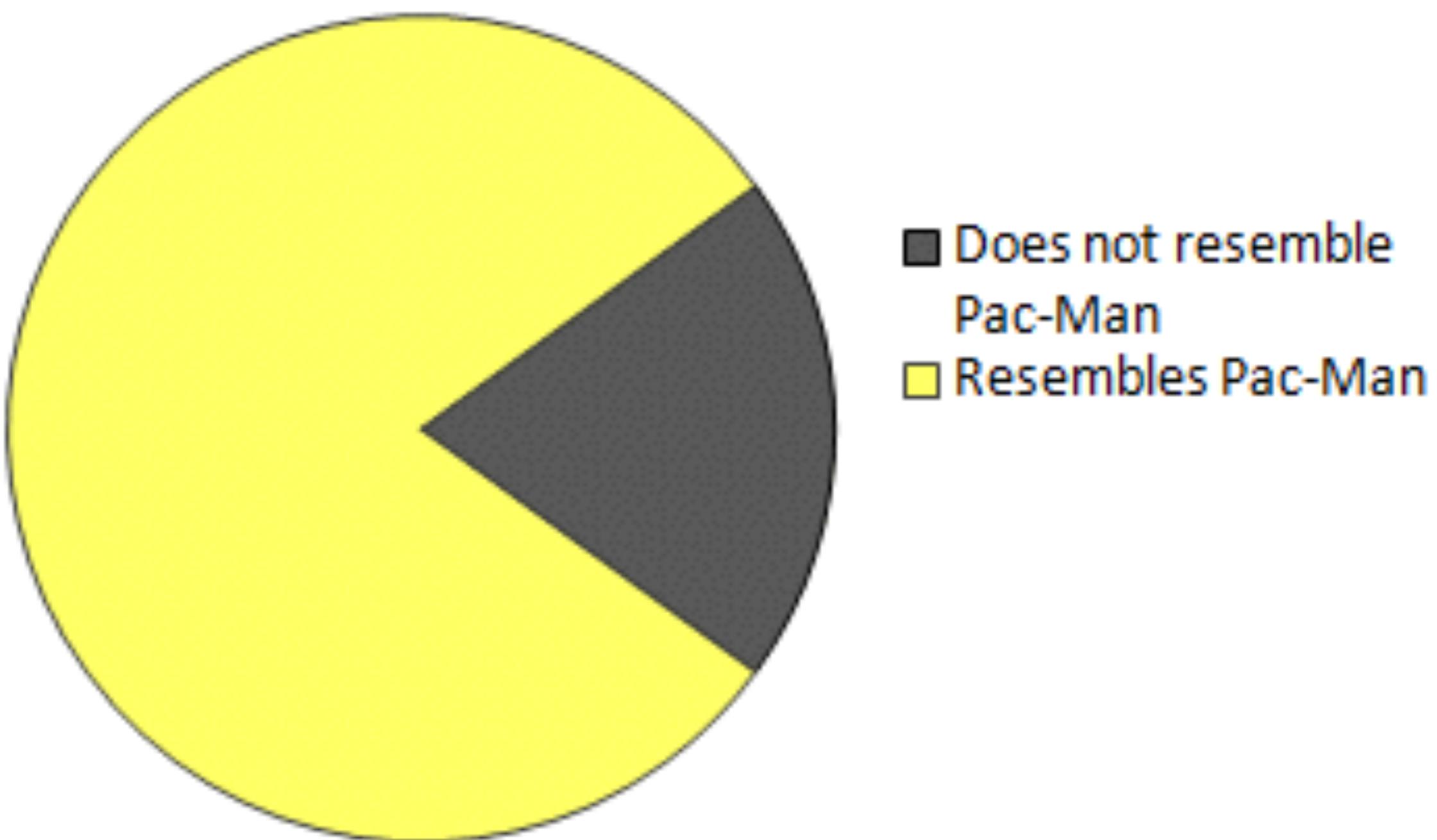


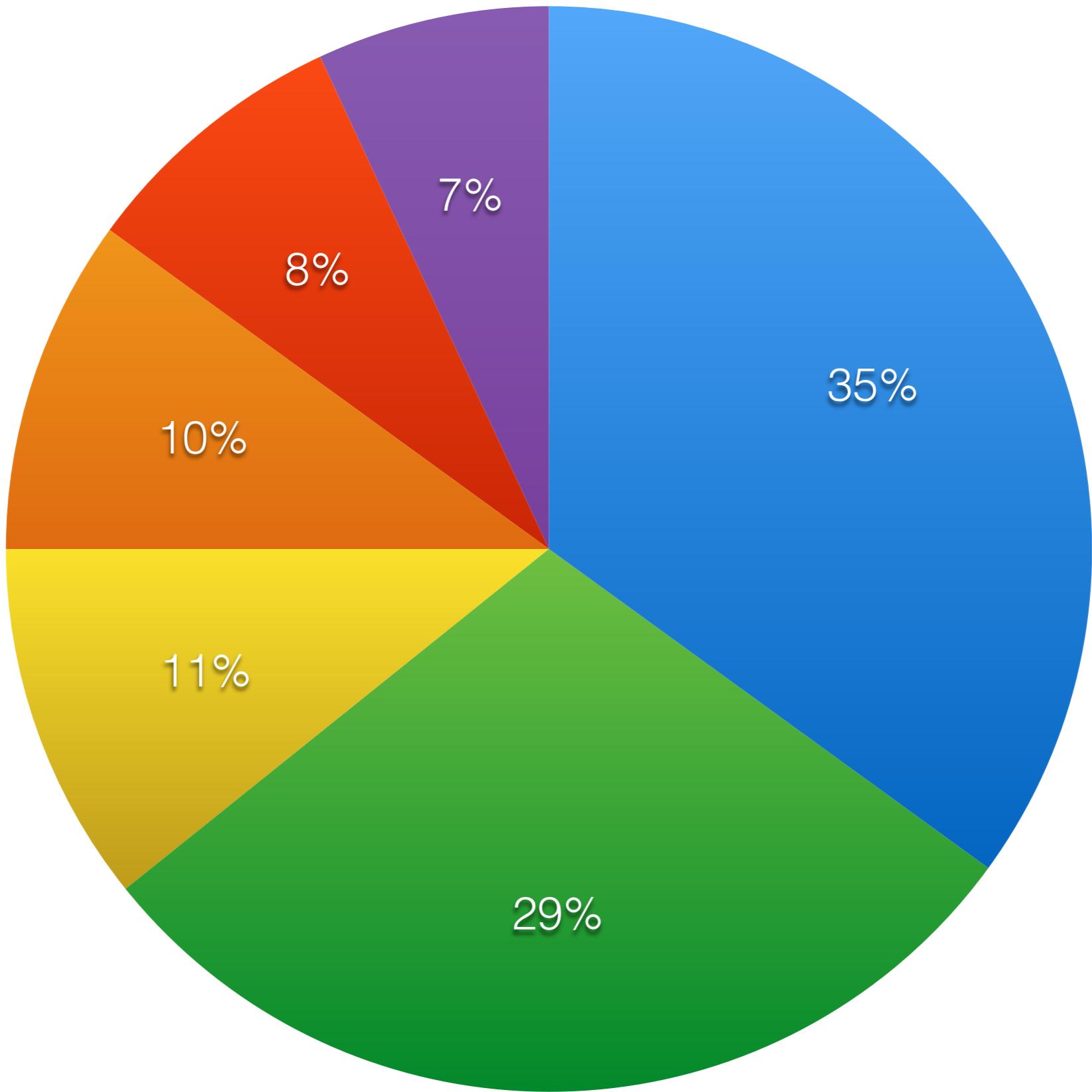


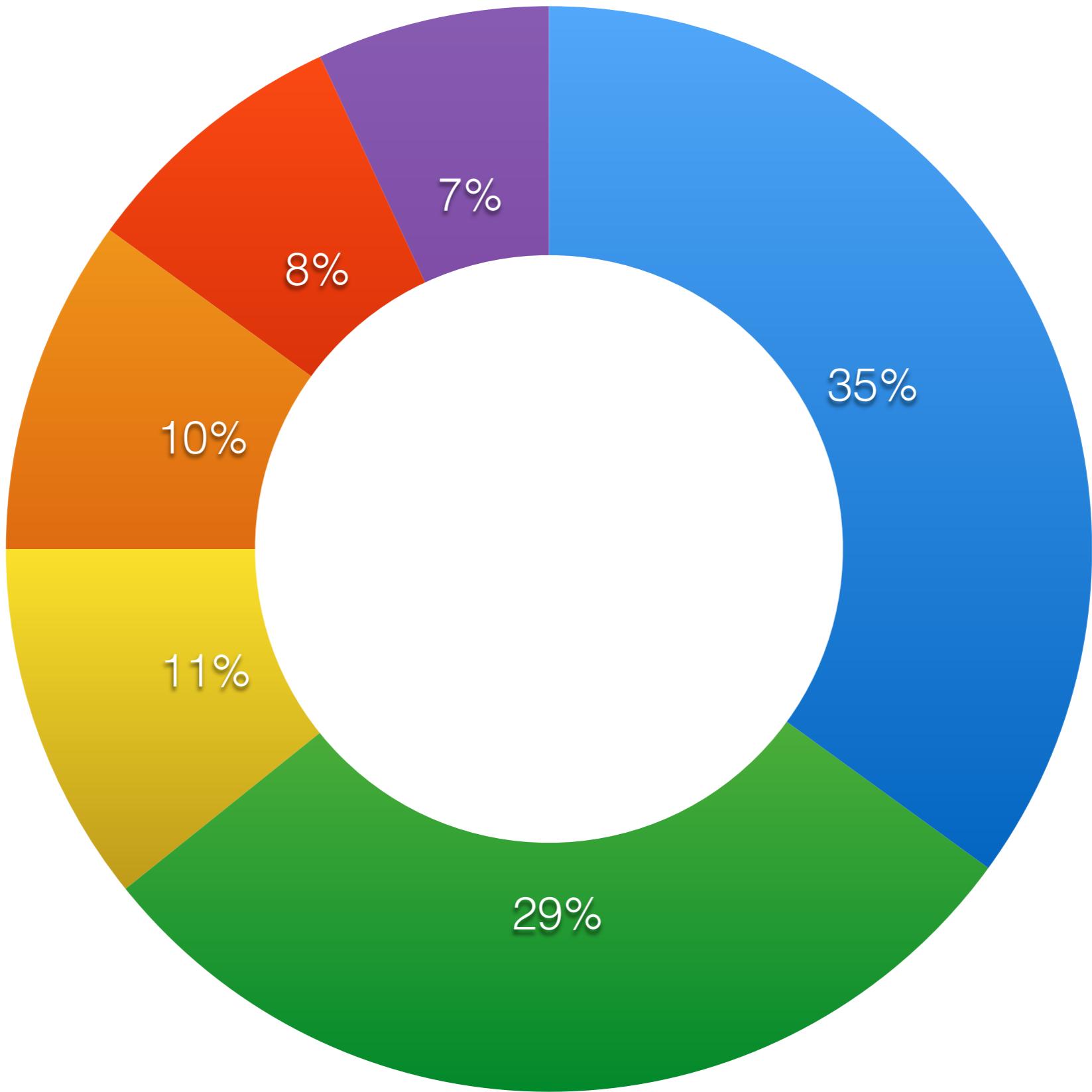
A brief interlude: Why
never to use a Pie Chart

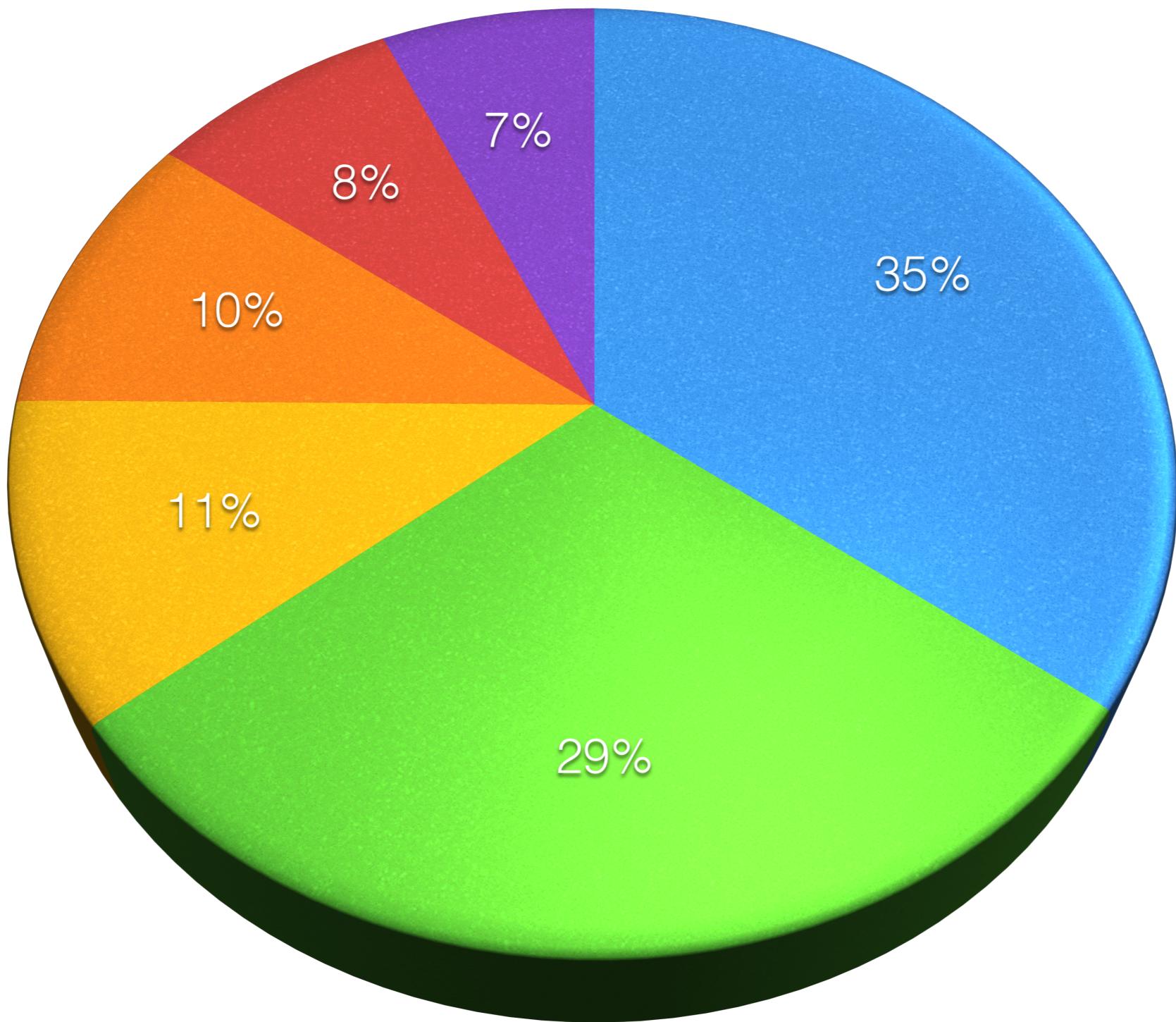
- 
- Pie I have eaten
 - Pie I have not yet eaten

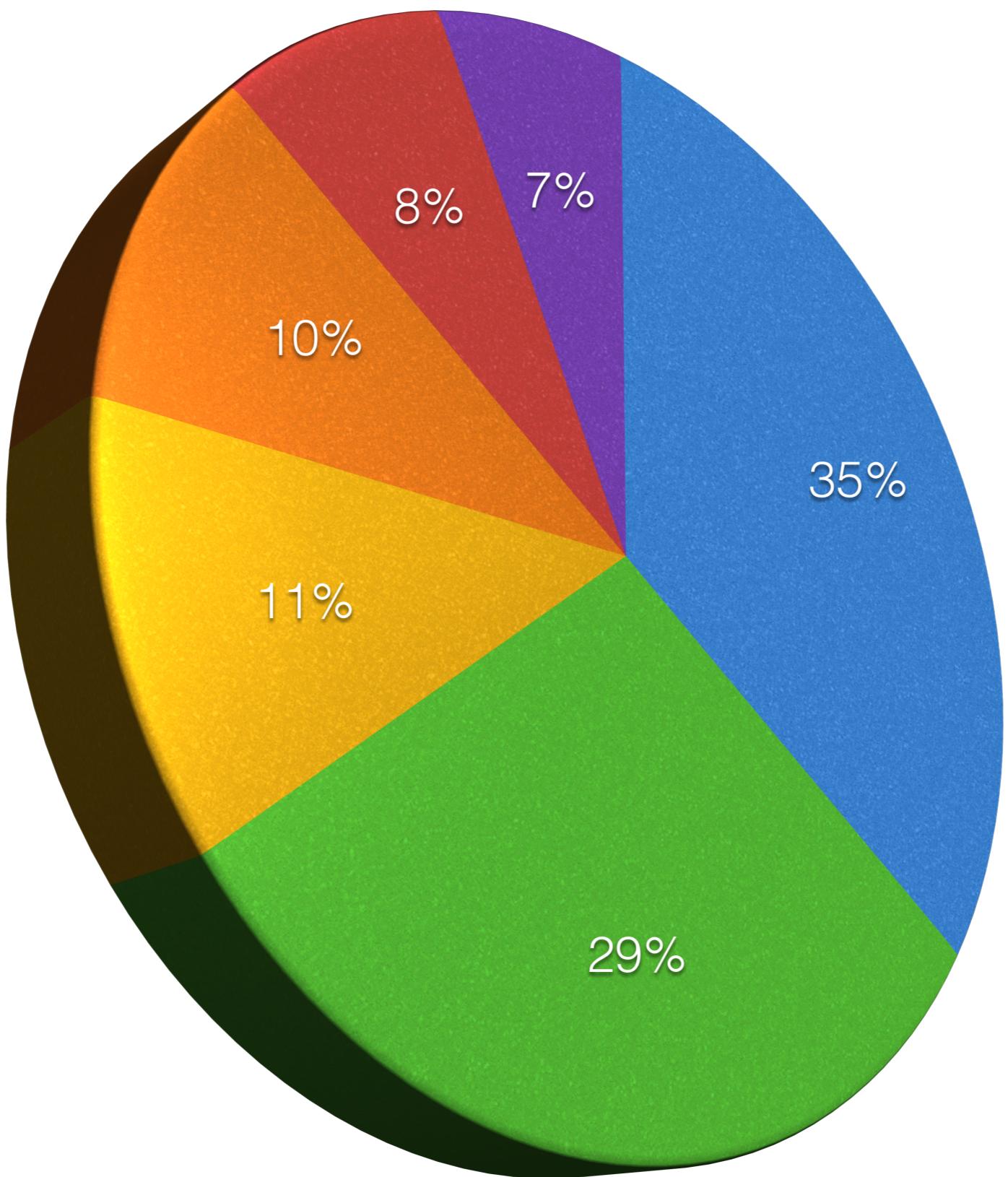
Percentage of Chart Which Resembles Pac-Man





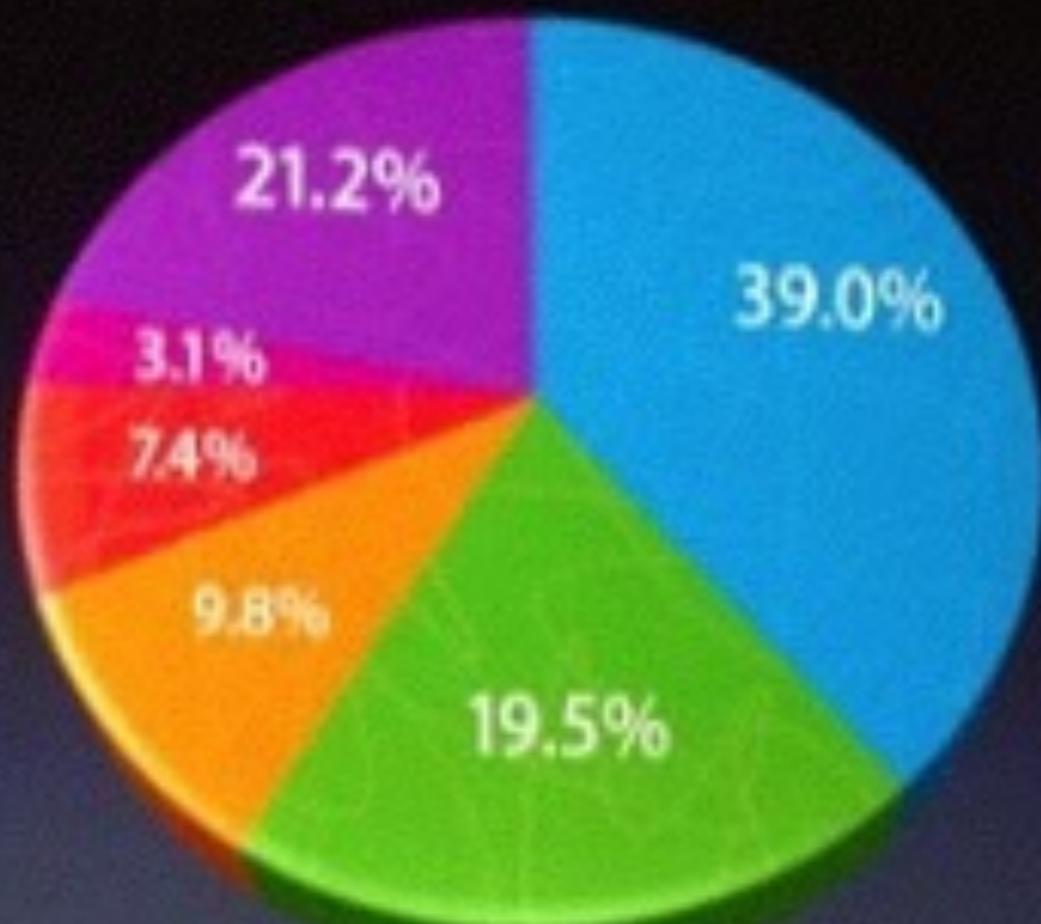






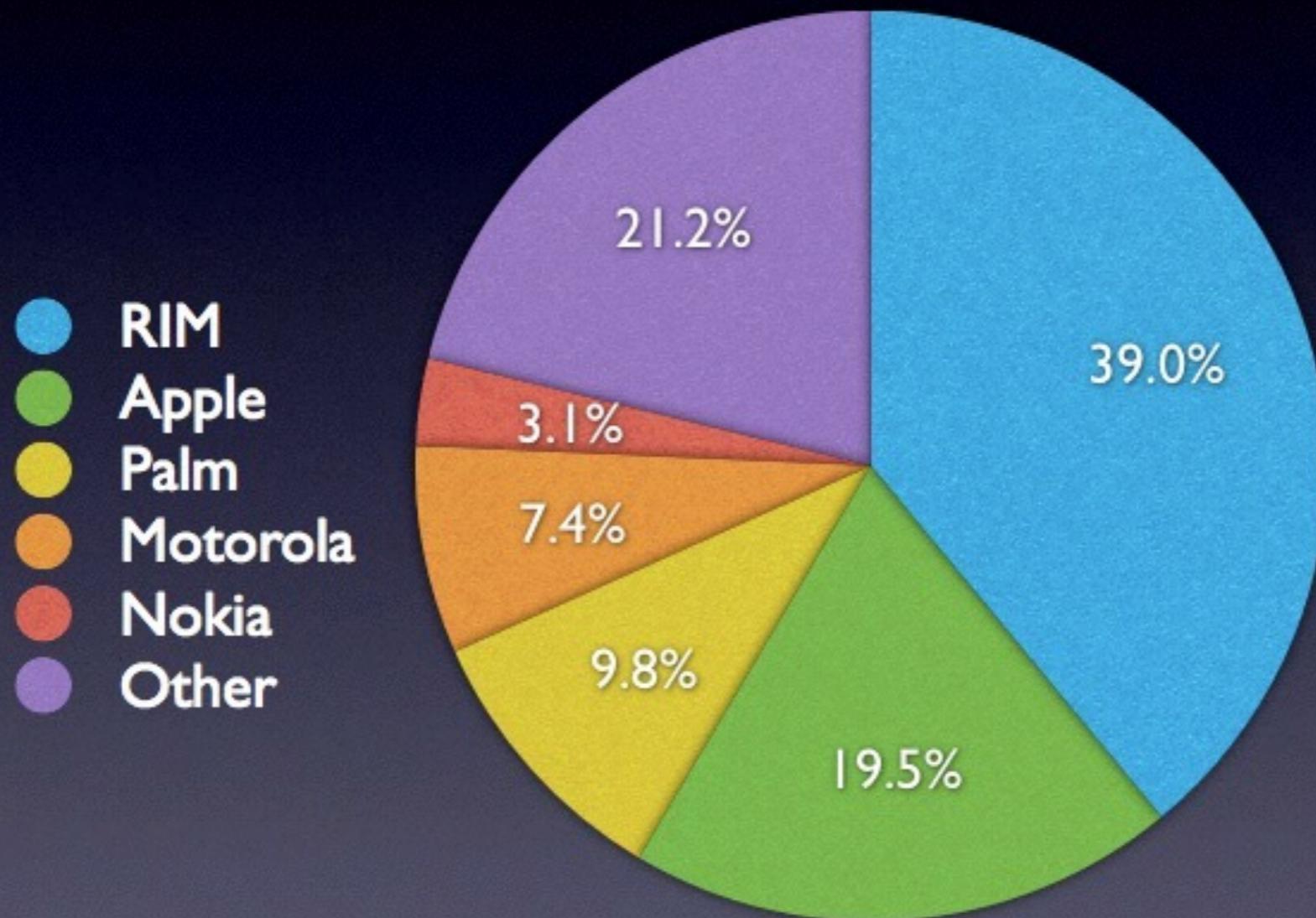
U.S. SmartPhone Marketshare

- RIM
- Apple
- Palm
- Motorola
- Nokia
- Other



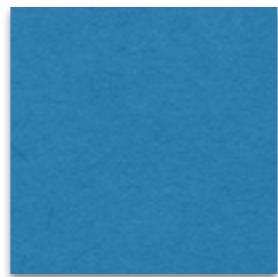
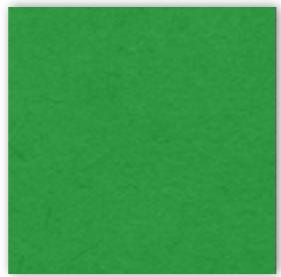
Gartner fo

U.S. SmartPhone Marketshare

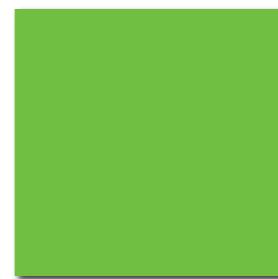
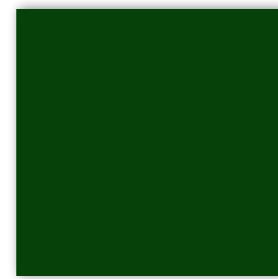


Visual Encodings: **Color**

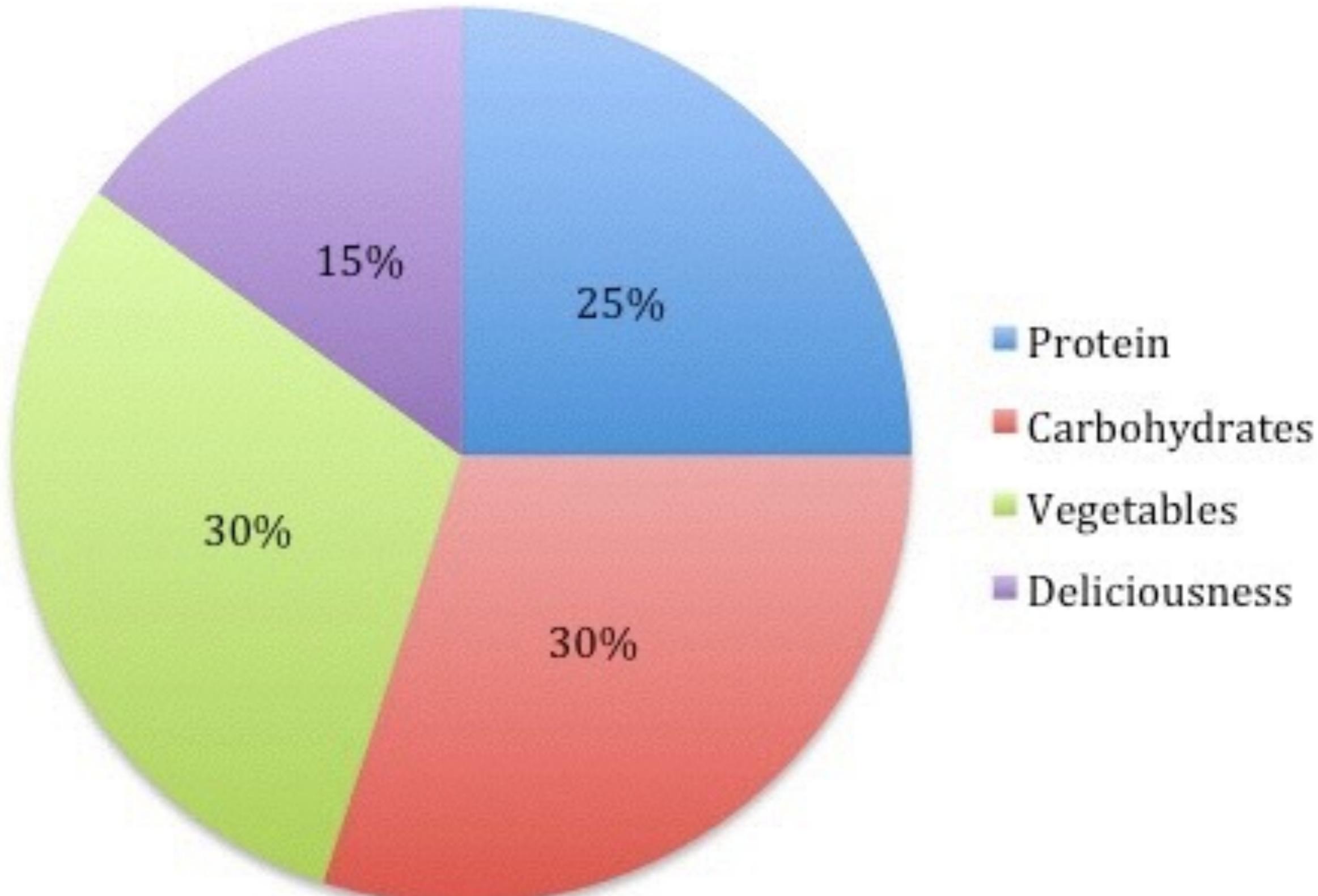
Hue



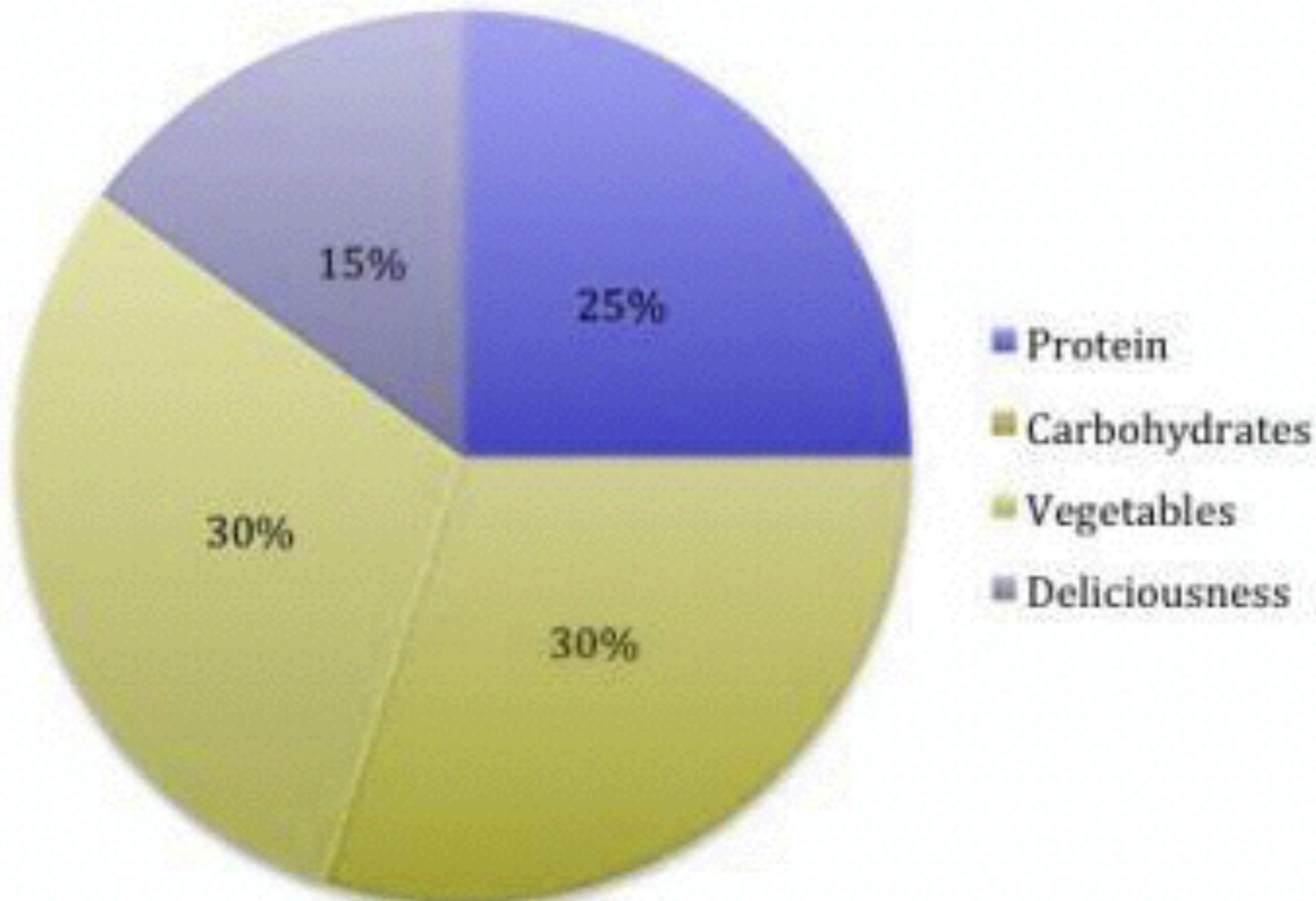
Saturation



A Healthy Meal



A Healthy Meal



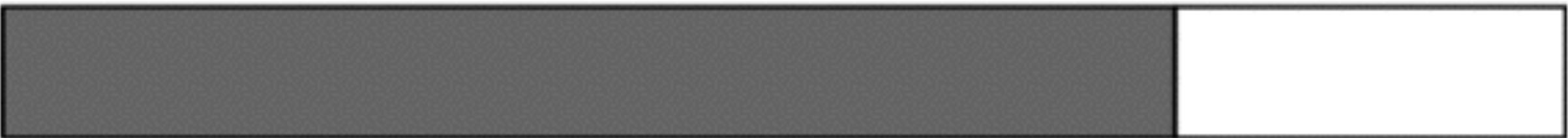
<http://www.color-blindness.com/coblis-color-blindness-simulator/>

Color	HTML/CSS Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
Black	Black	#000000	(0,0,0)
	White	#FFFFFF	(255,255,255)
Red	Red	#FF0000	(255,0,0)
Lime	Lime	#00FF00	(0,255,0)
Blue	Blue	#0000FF	(0,0,255)
Yellow	Yellow	#FFFF00	(255,255,0)
Cyan / Aqua	Cyan / Aqua	#00FFFF	(255,255)
Magenta / Fuchsia	Magenta / Fuchsia	#FF00FF	(255,0,255)
Silver	Silver	#C0C0C0	(192,192,192)
Gray	Gray	#808080	(128,128,128)
Maroon	Maroon	#800000	(128,0,0)
Olive	Olive	#808000	(128,128,0)
Green	Green	#008000	(0,128,0)
Purple	Purple	#800080	(128,0,128)
Teal	Teal	#008080	(128,128)
Navy	Navy	#000080	(0,0,128)

Picking a color palette

colorbrewer2.org

**Brainpower used
for decoding**



Total brainpower available



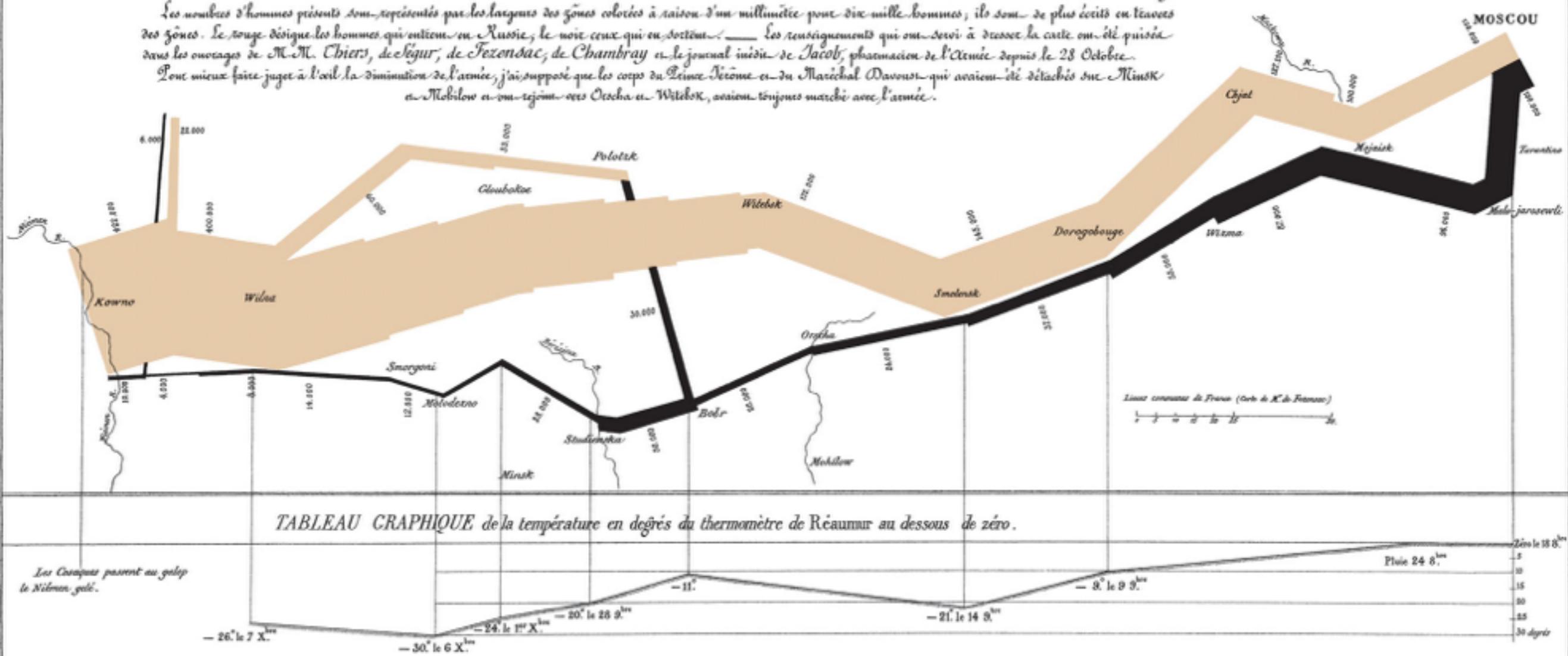
Carte Figurative des pertes successives en hommes de l'Armée Française dans la Campagne de Russie 1812-1813.
Dessiné par M. Minard, Ancien Lieutenant Général des Ponts et Chaussées en retraite.

Précisé par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.

Paris, le 20 Novembre 1869.

Les nombres d'hommes présents sont représentés par les larges des zones colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en lettres des zones. Le rouge désigne les hommes qui entrent en Russie; le noir ceux qui en sortent. — Les renseignements qui ont servi à dresser la carte ont été pris dans les ouvrages de M. M. Chiers, de Séguir, de Fézensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Académie depuis le 28 Octobre.

Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Nérome et du Maréchal Davout qui avaient été détachés sur Minsk au Mobiliori et qui rejoignaient Osscha et Witlobk, avaient toujours marché avec l'armée.

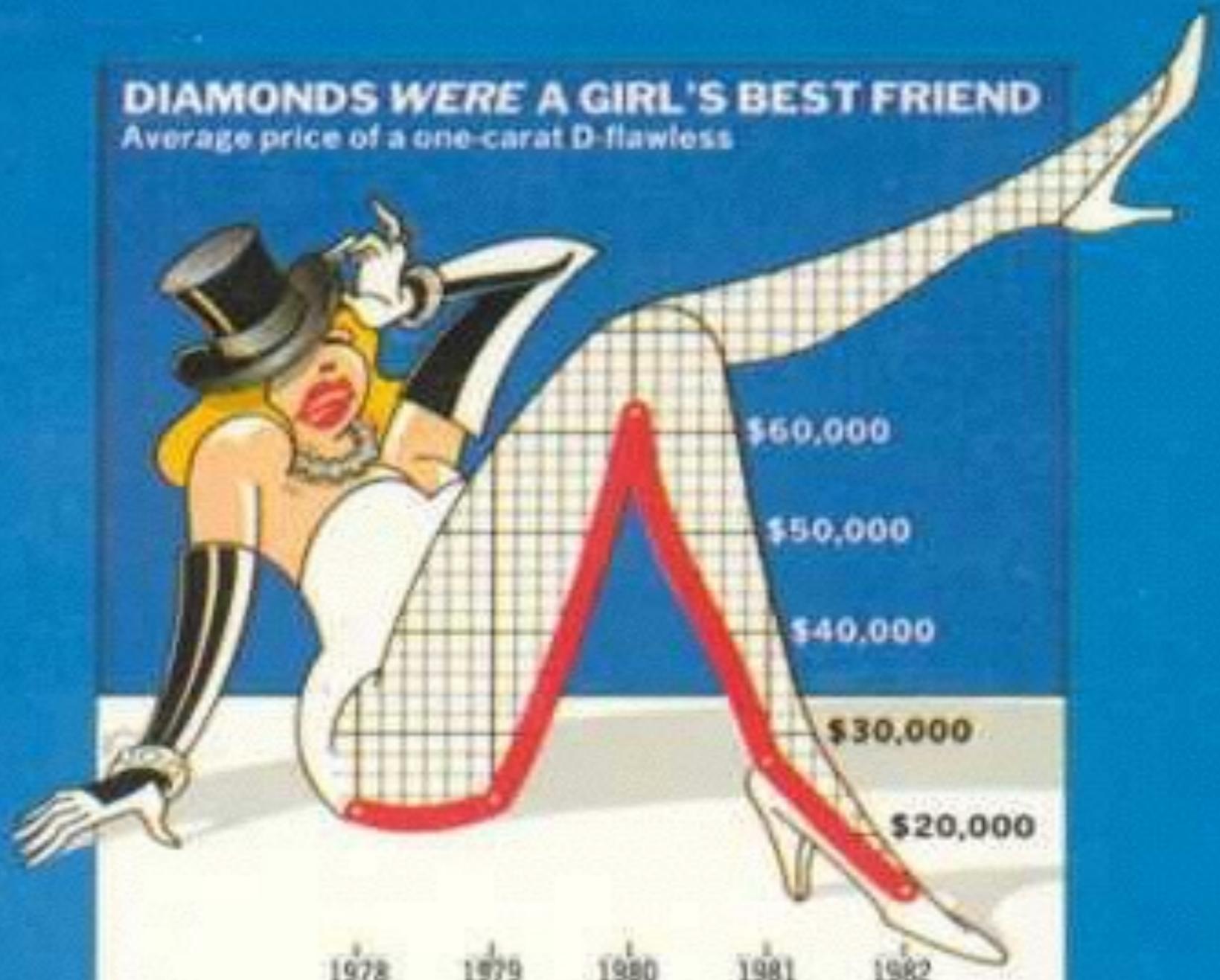


Actas del Reunión de Pcs. de Méjico, 21-22-23 d'Agosto

Am. Ind. Review at Bremen

DIAMONDS WERE A GIRL'S BEST FRIEND

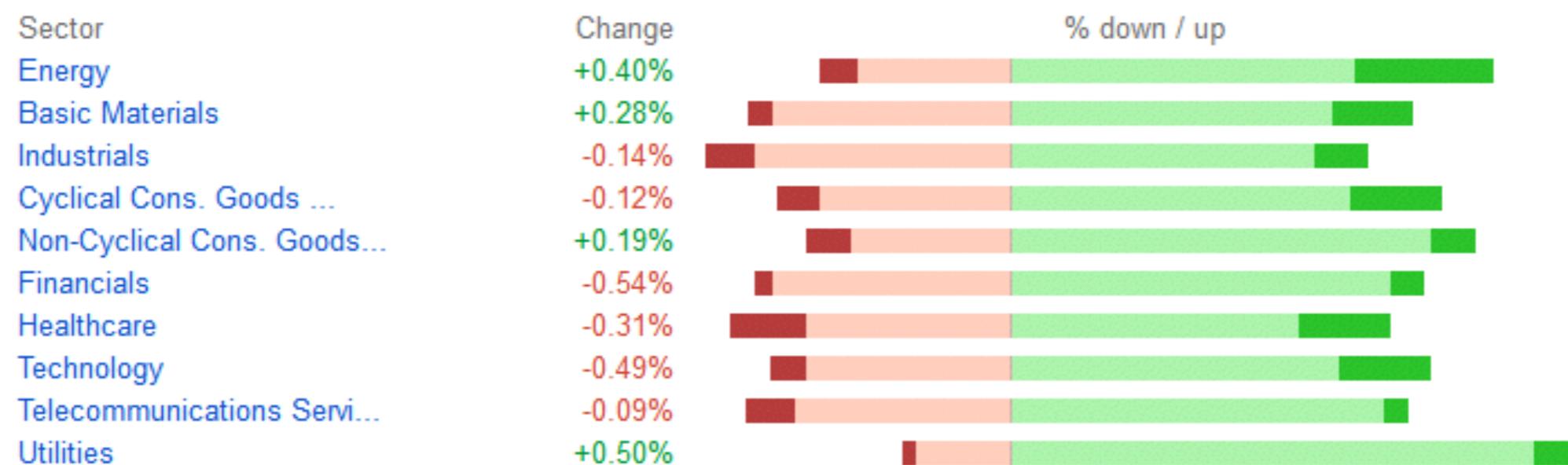
Average price of a one-carat D-flawless



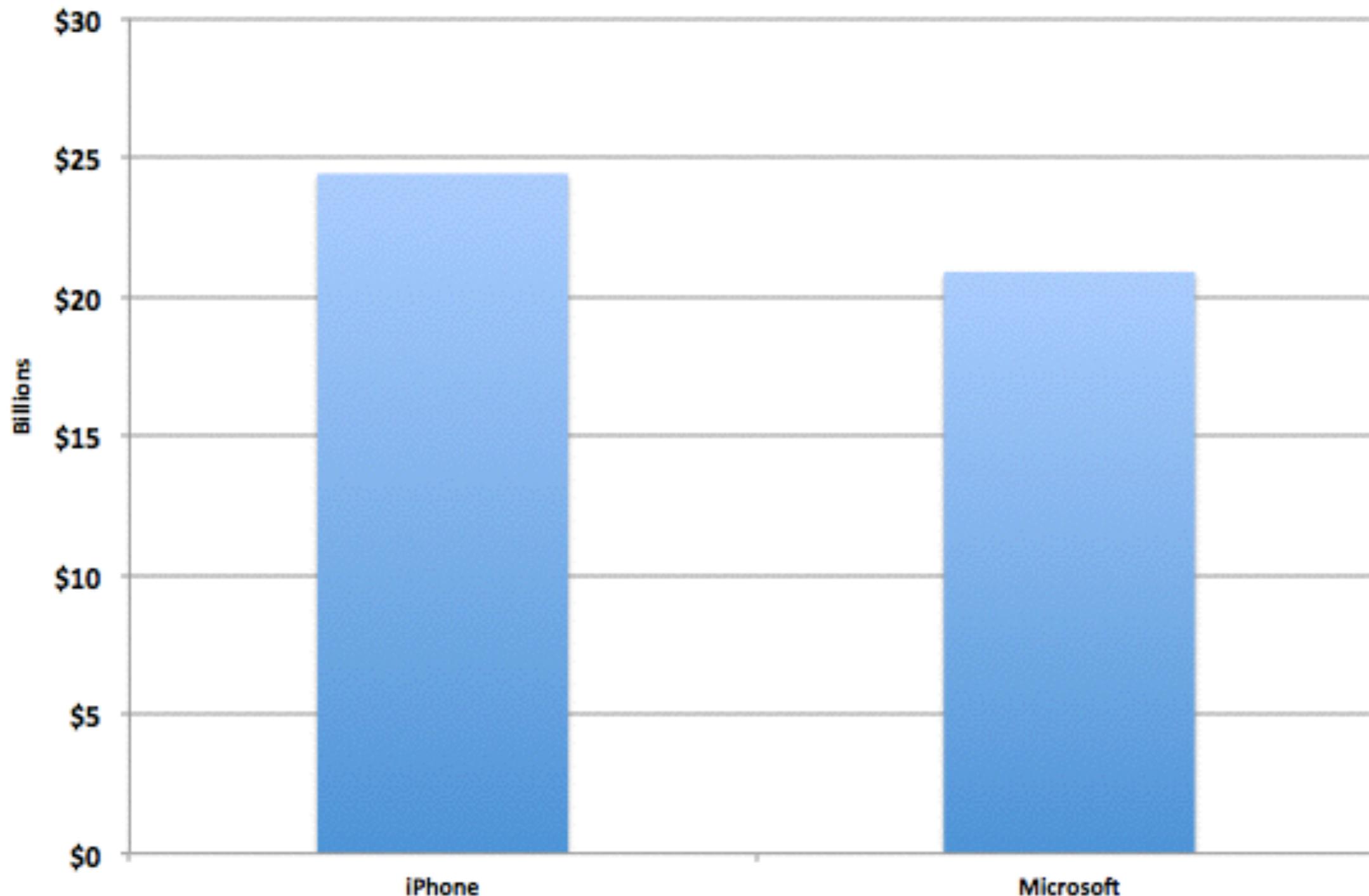
TIME Chart by Roger Hargreaves

Source: The Diamond Registry

Sector summary



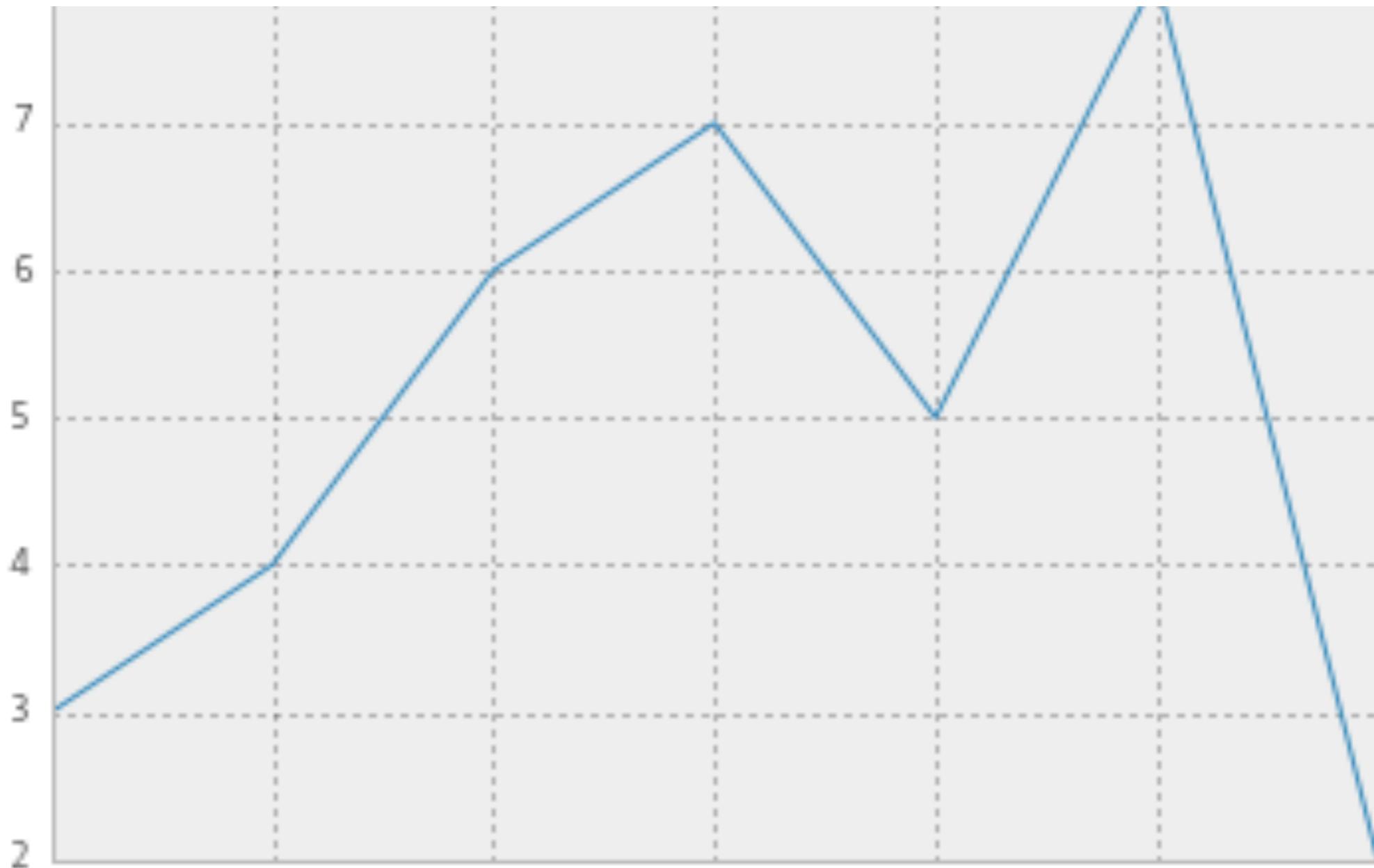
iPhone vs. Microsoft (Revenue)



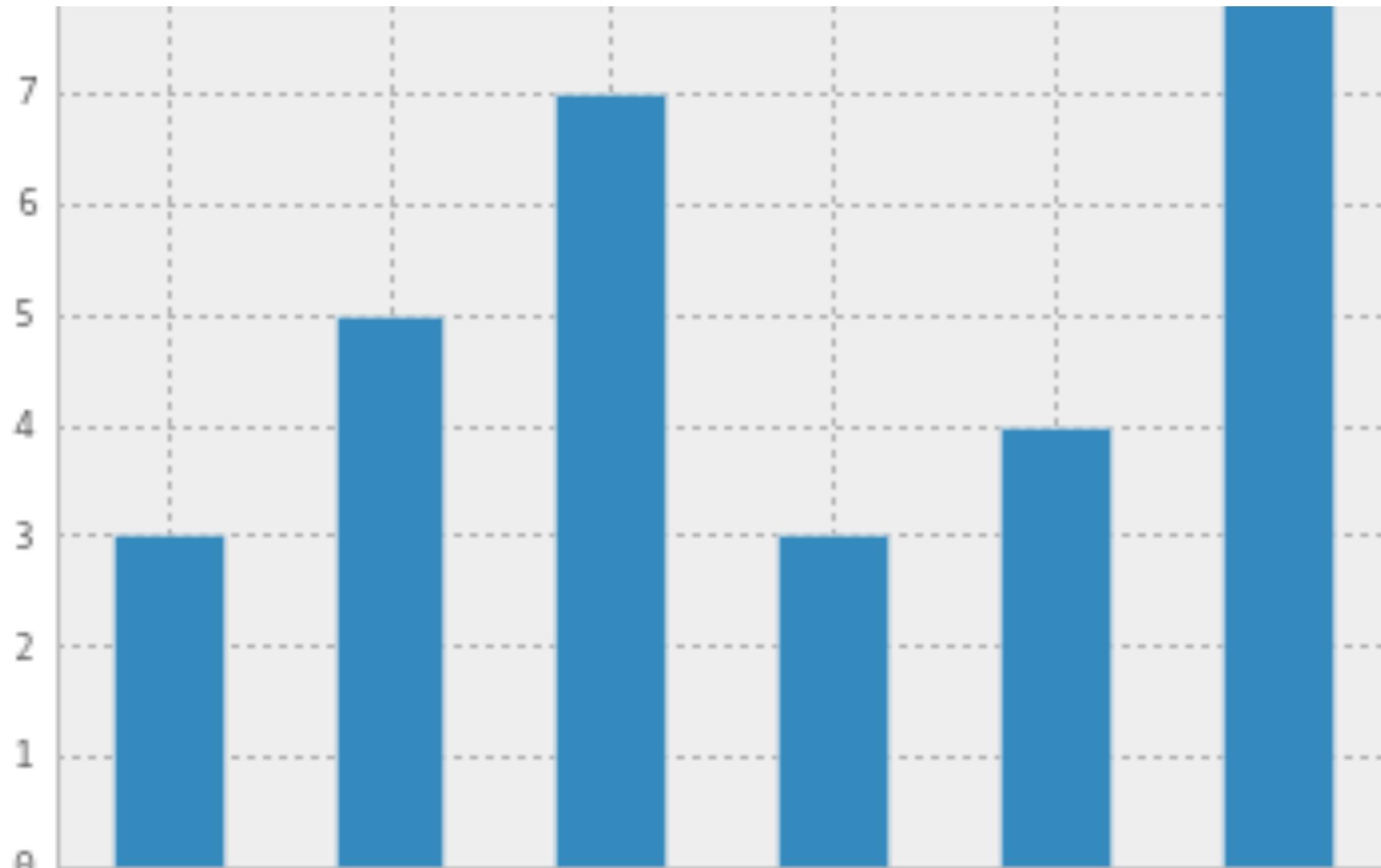
Recommended Reading



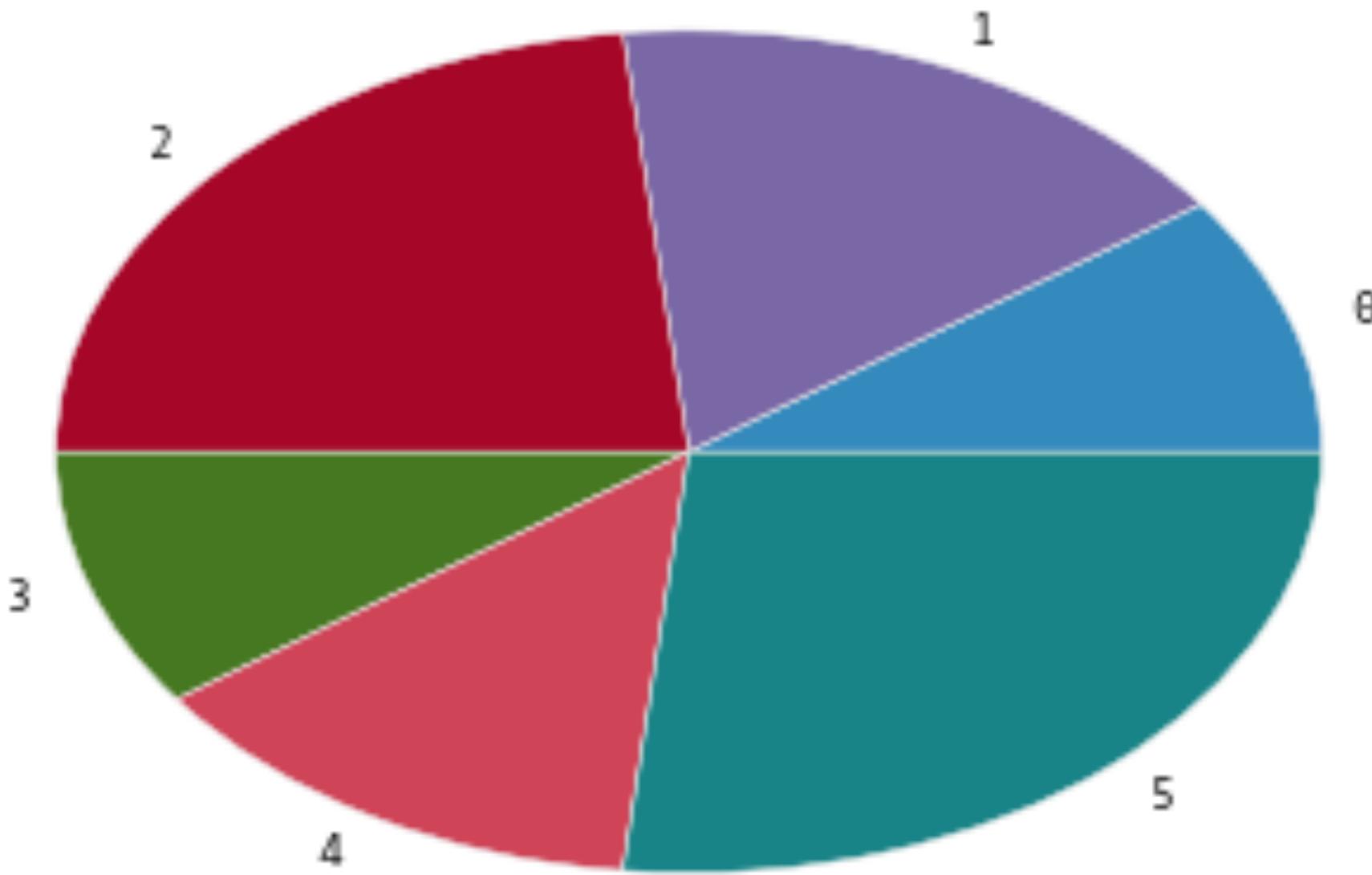
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
pd.options.display.mpl_style = 'default'
```



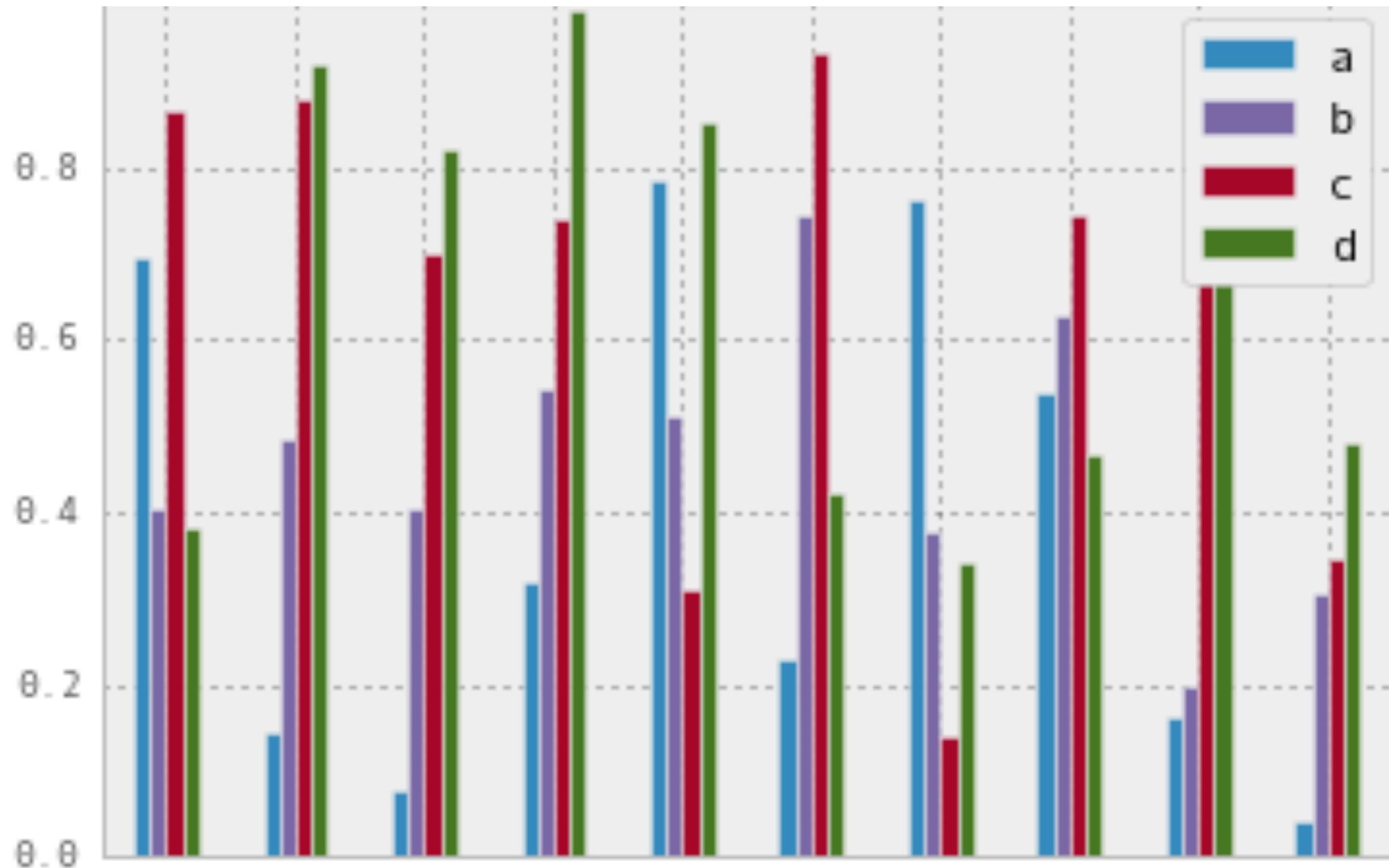
```
data = pd.Series( [3,4,6,7,5,8,2] )
graph = data.plot()
```



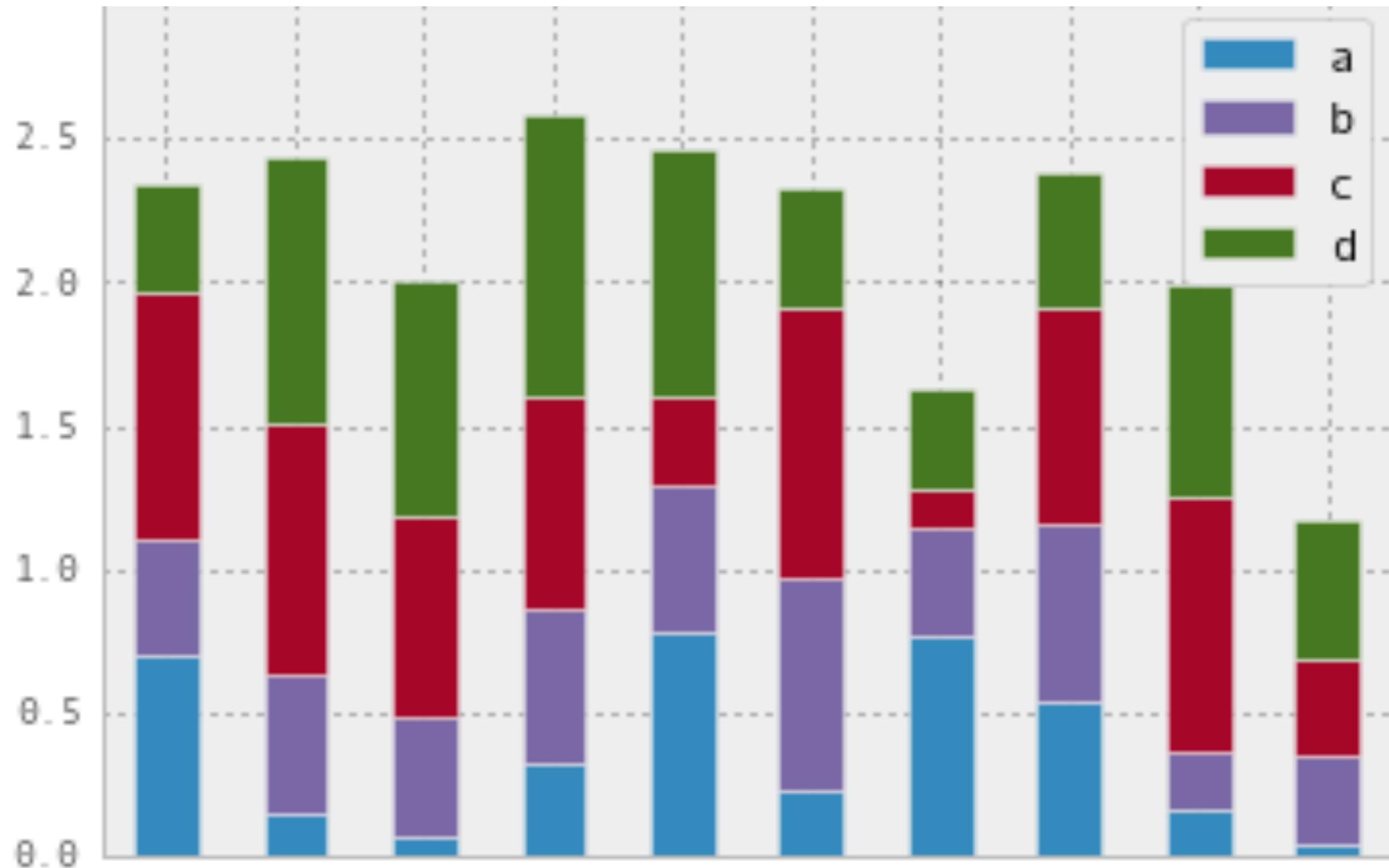
```
barchart = data.plot( kind="bar" )
```



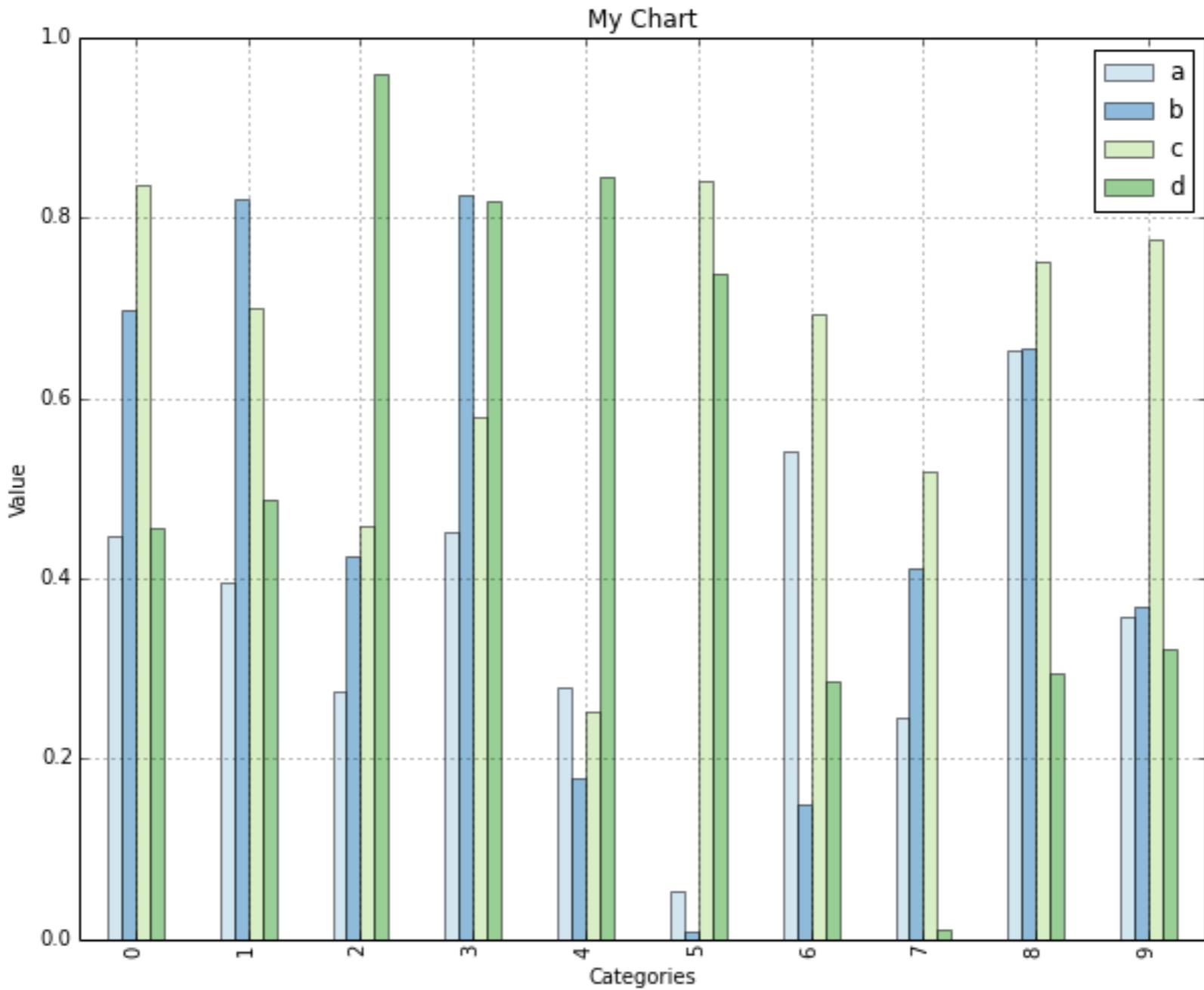
```
piechart = data.plot( kind="pie" )
```



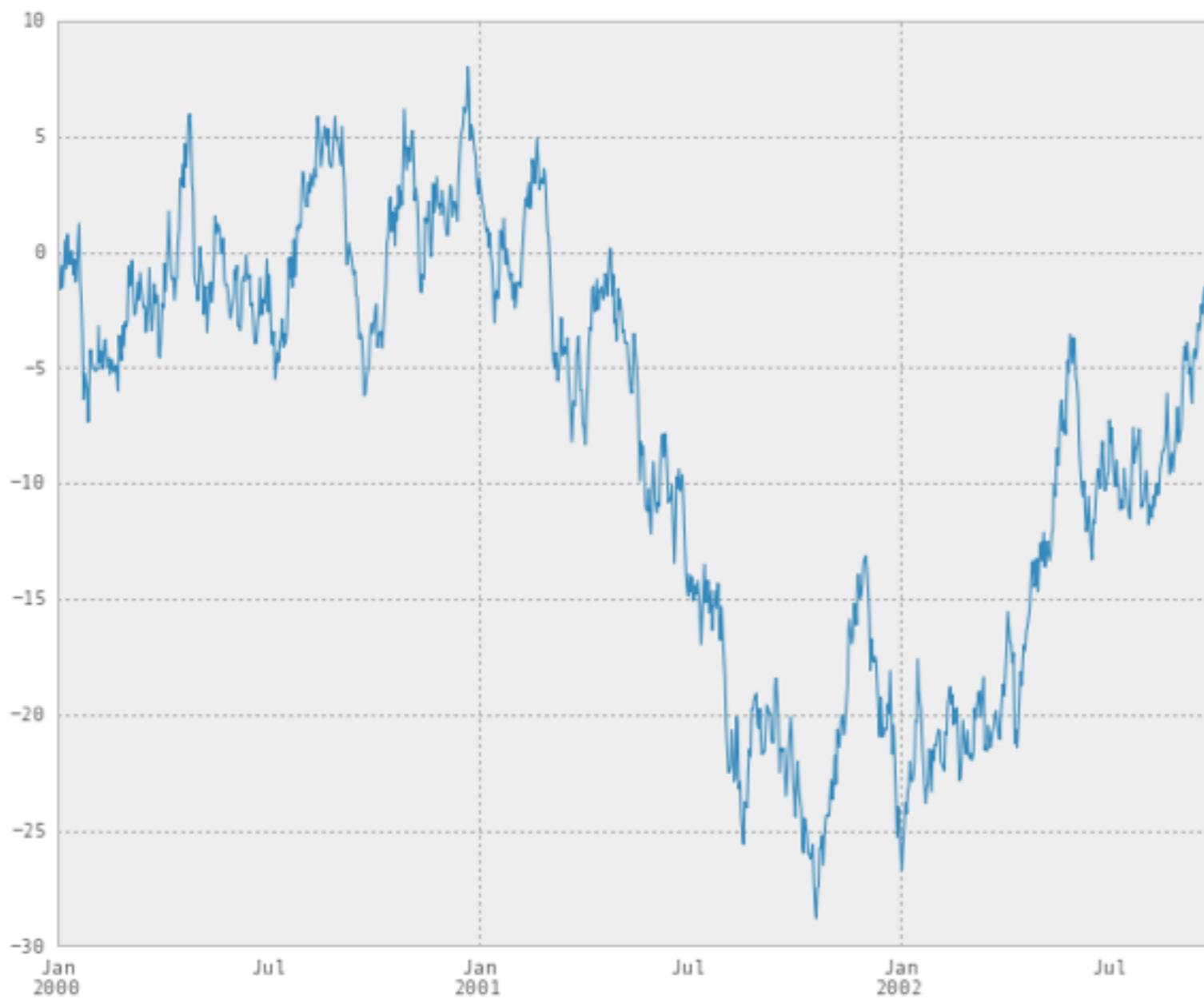
```
df2 = pd.DataFrame(np.random.rand(10, 4),
columns=[ 'a' , 'b' , 'c' , 'd' ])
df2.plot( kind='bar' )
```



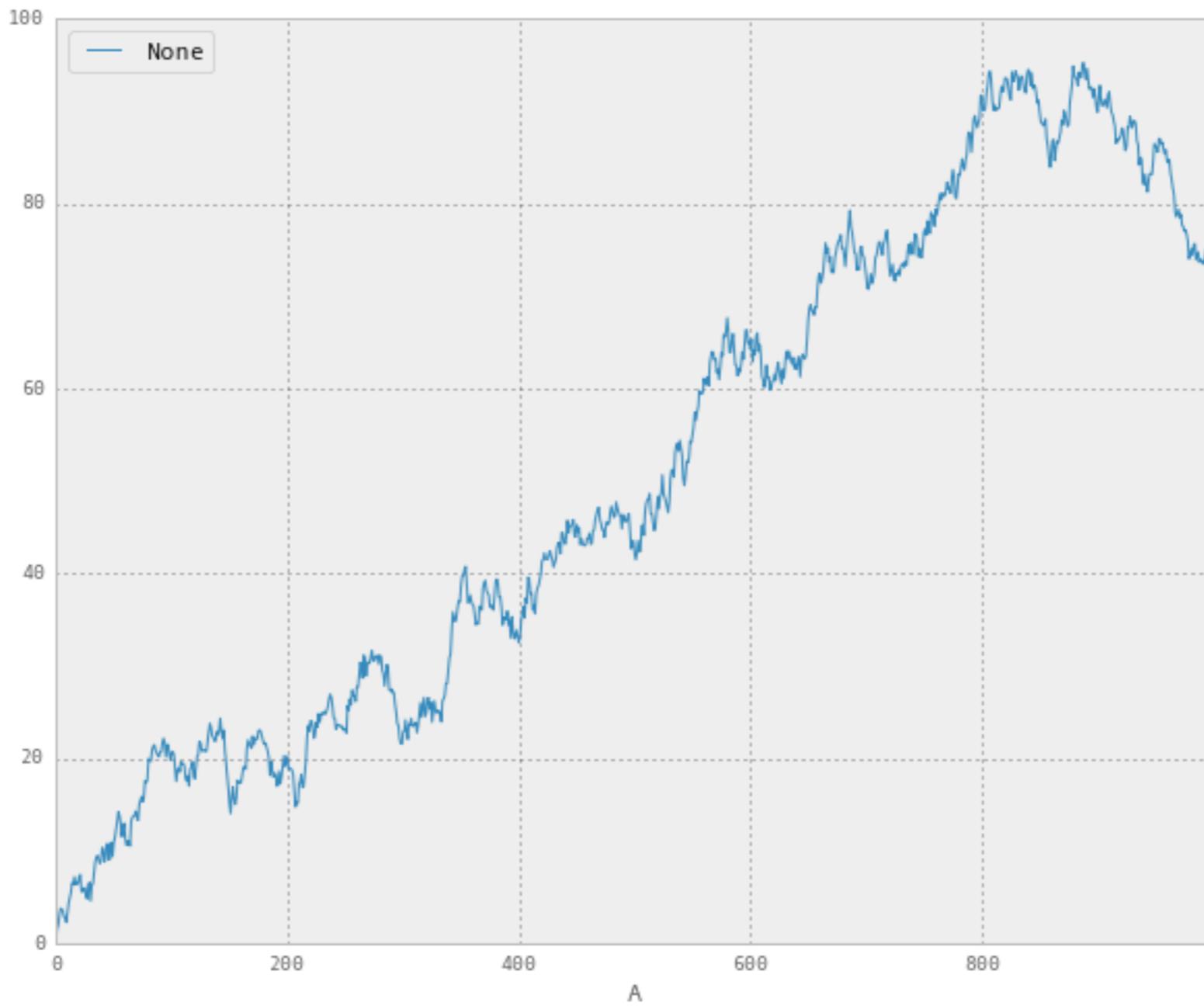
```
df2.plot( kind='bar' , stacked=True )
```



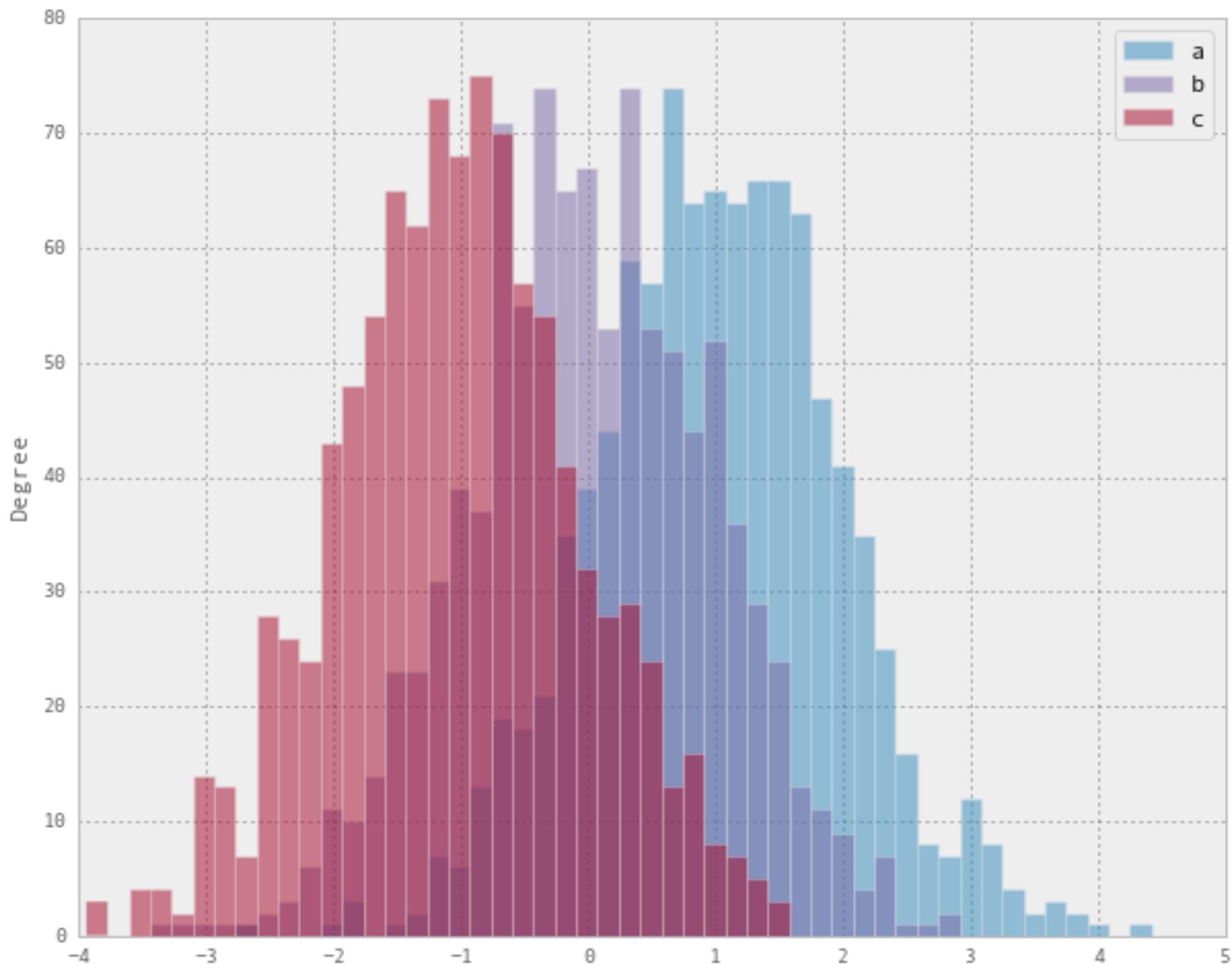
```
df2.plot( kind='bar' ,  
          color=( '#a6cee3' , '#1f78b4' , '#b2df8a' , '#33a02c' ) ,  
          alpha=0.5 ,  
          width=0.5 ,  
          figsize=( 10 , 8 ))  
plt.title( "My Chart" )  
plt.xlabel( "Categories" )  
plt.ylabel( "Value" )
```



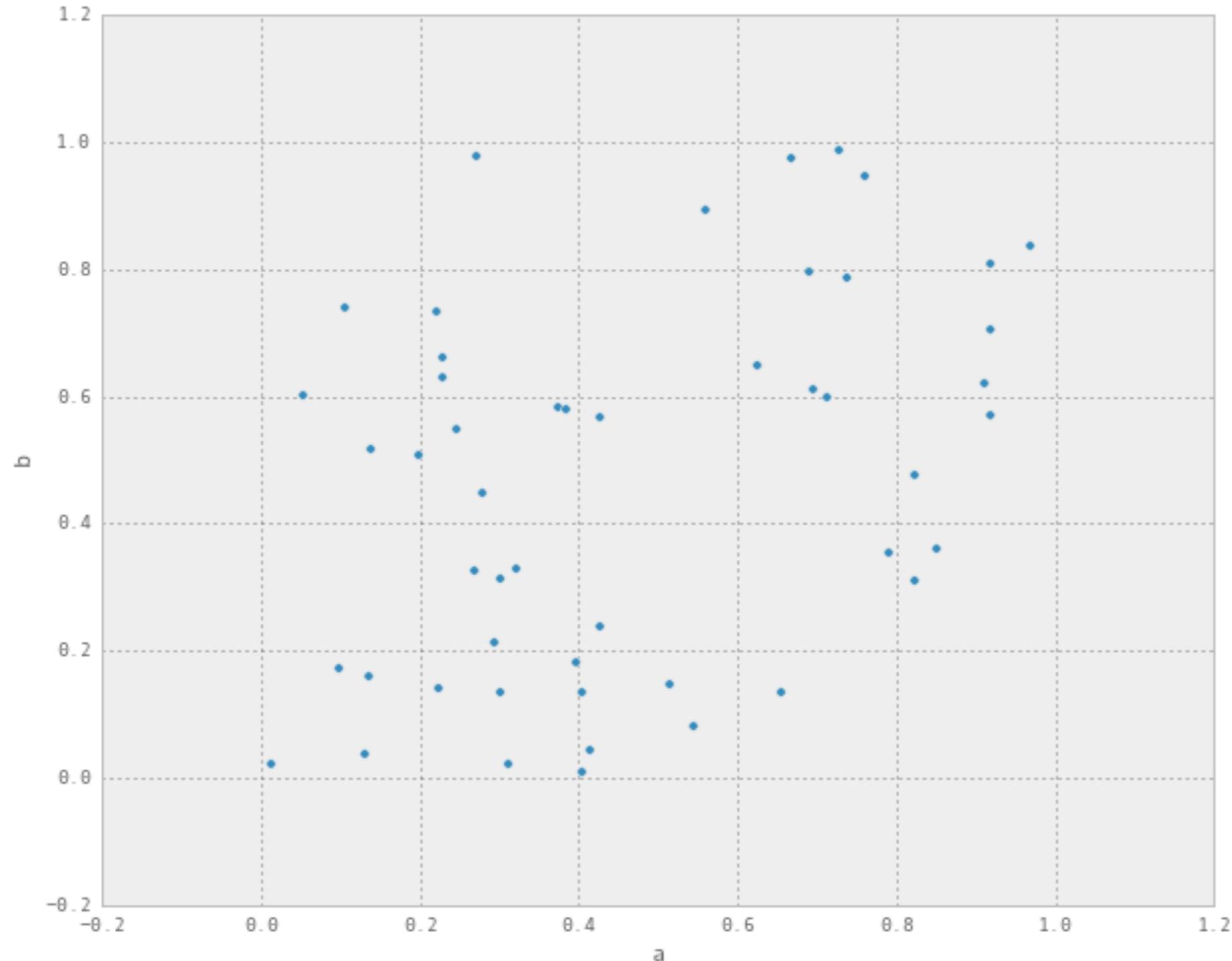
```
ts = pd.Series(np.random.randn( 1000 ),  
index=pd.date_range('1/1/2000', periods=1000))  
ts = ts.cumsum()  
timeseriesChart = ts.plot( figsize=(10, 8) )
```



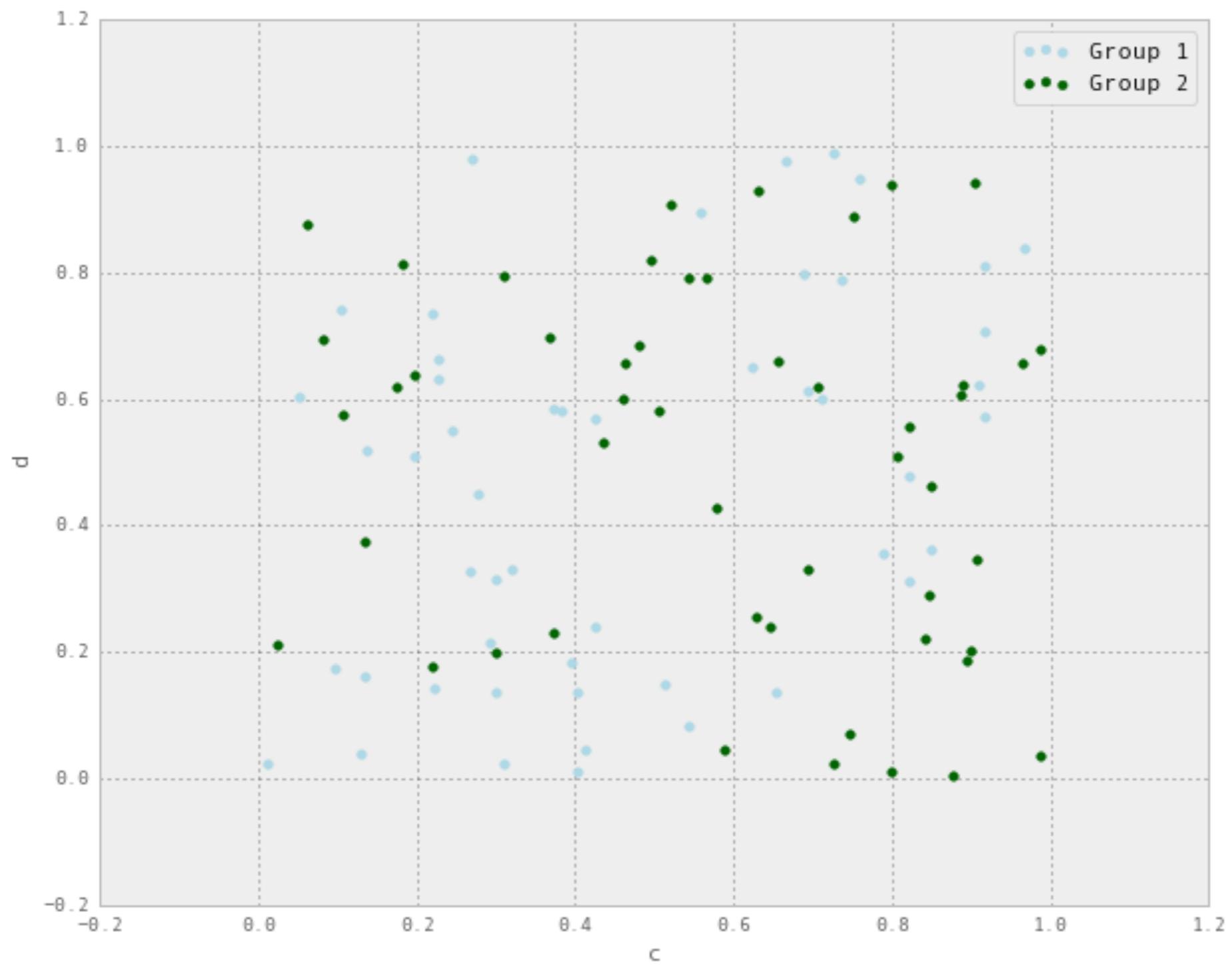
```
df3 = pd.DataFrame(np.random.randn(1000, 2),  
columns=[ 'B' , 'C' ]).cumsum()  
df3[ 'A' ] = pd.Series(list(range(len(df3))))  
  
df3.plot( x='A' , y='B' )
```



```
df4.plot(kind='hist',  
         alpha=0.5,  
         bins=50 )
```

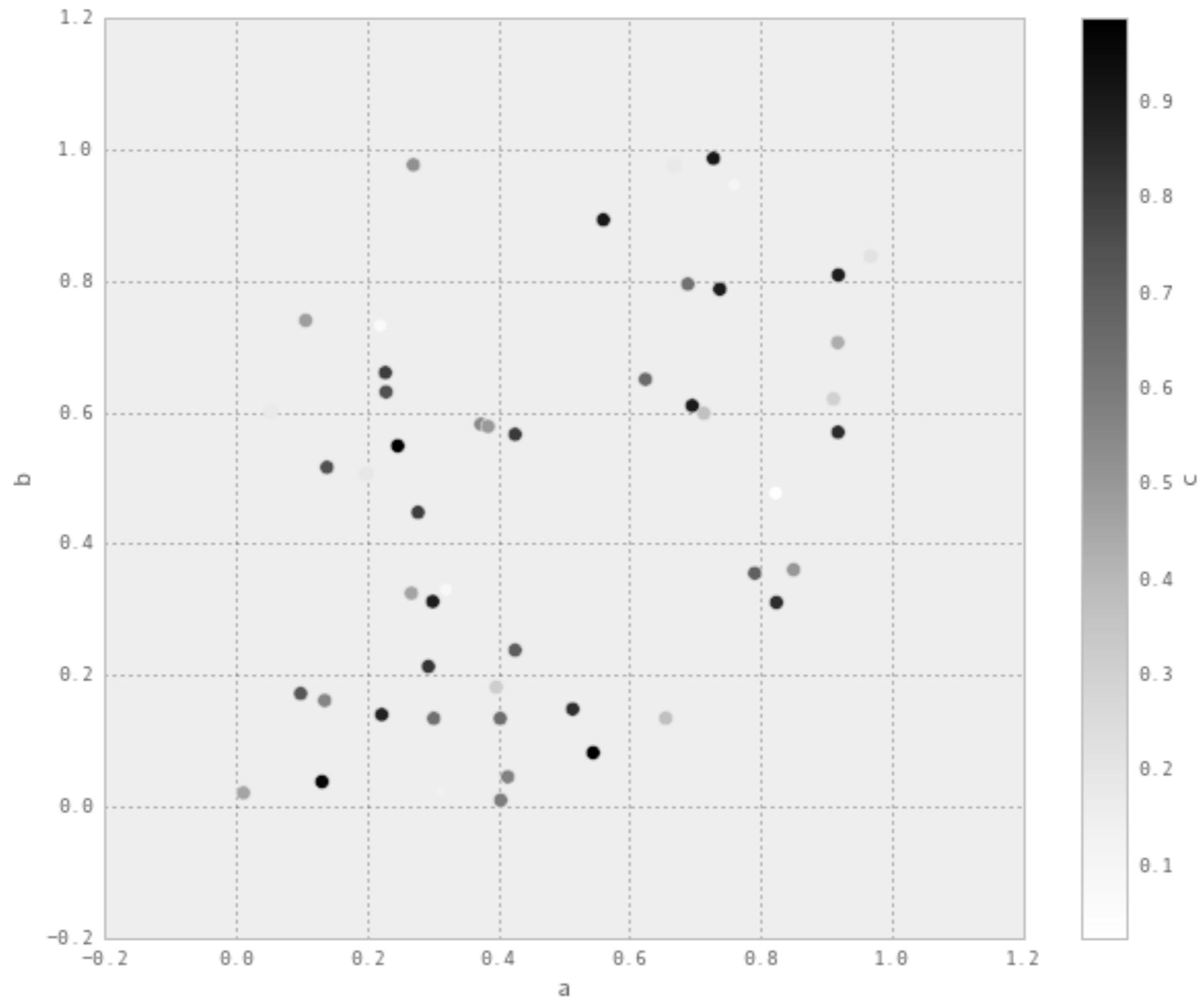


```
df5 = pd.DataFrame(np.random.rand(50, 4),  
columns=[ 'a' , 'b' , 'c' , 'd' ] )  
df5.plot(kind='scatter', x='a', y='b' )
```

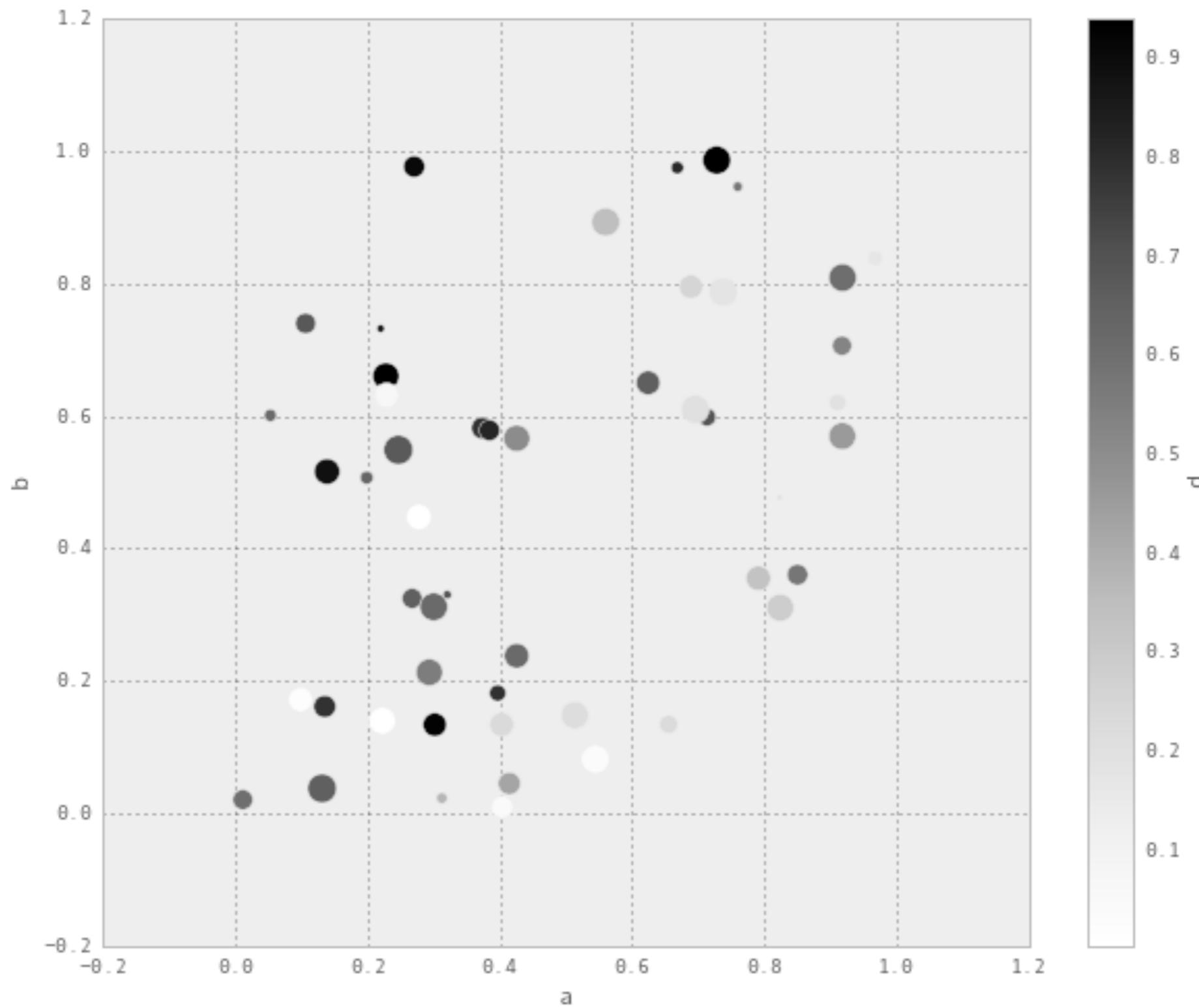


```
ax = df5.plot(kind='scatter', x='a', y='b',
               color='LightBlue',
               label='Group 1',
               figsize=(10, 8) )
```

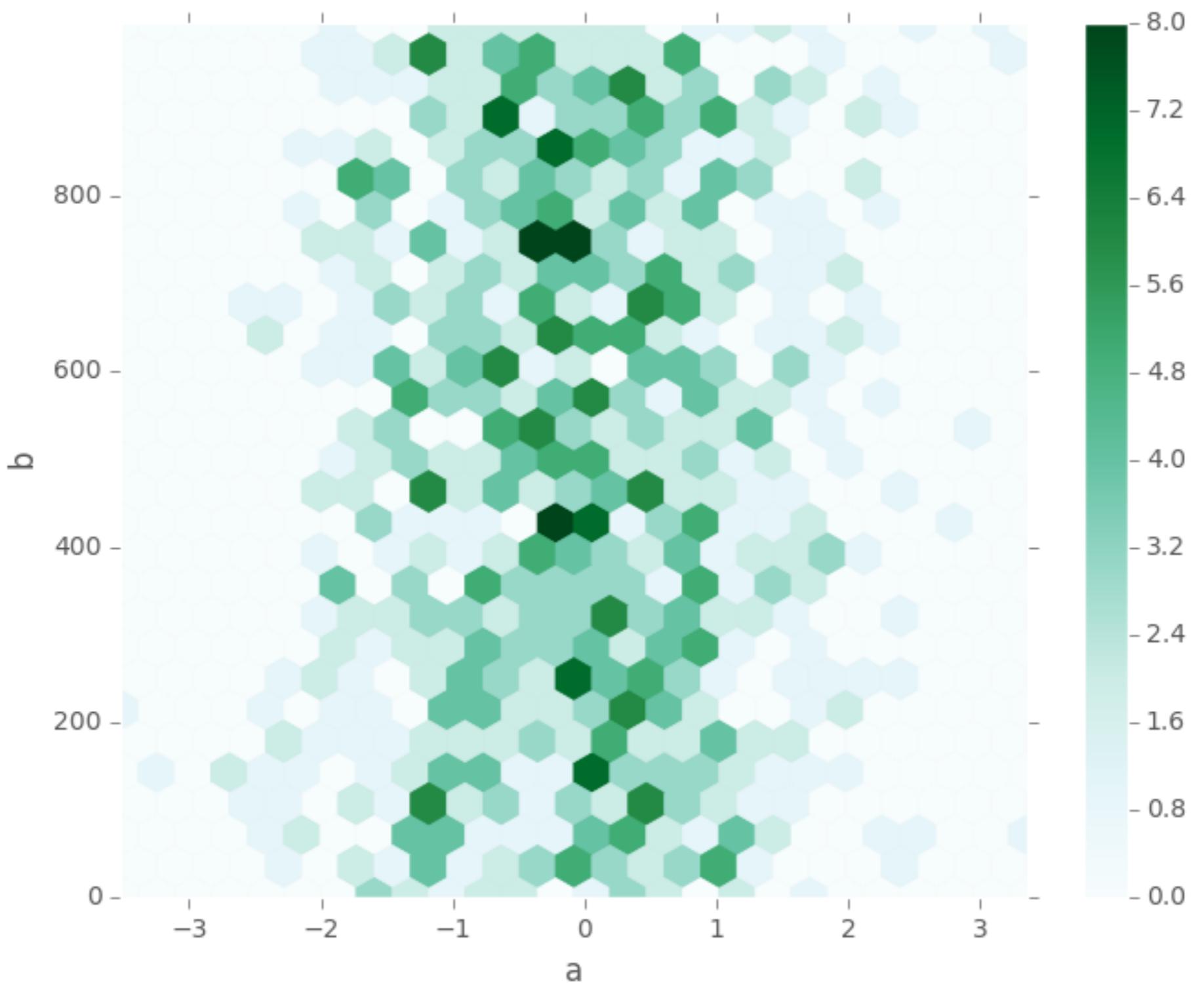
```
df5.plot(kind='scatter', x='c', y='d',
          color='DarkGreen',
          label='Group 2',
          ax=ax)
```



```
df5.plot(kind='scatter', x='a', y='b',
         c='c', s=50 )
```



```
scatter = df5.plot(kind='scatter',
                    x='a',
                    y='b',
                    s=df5['c']*200,
                    c='d')
```



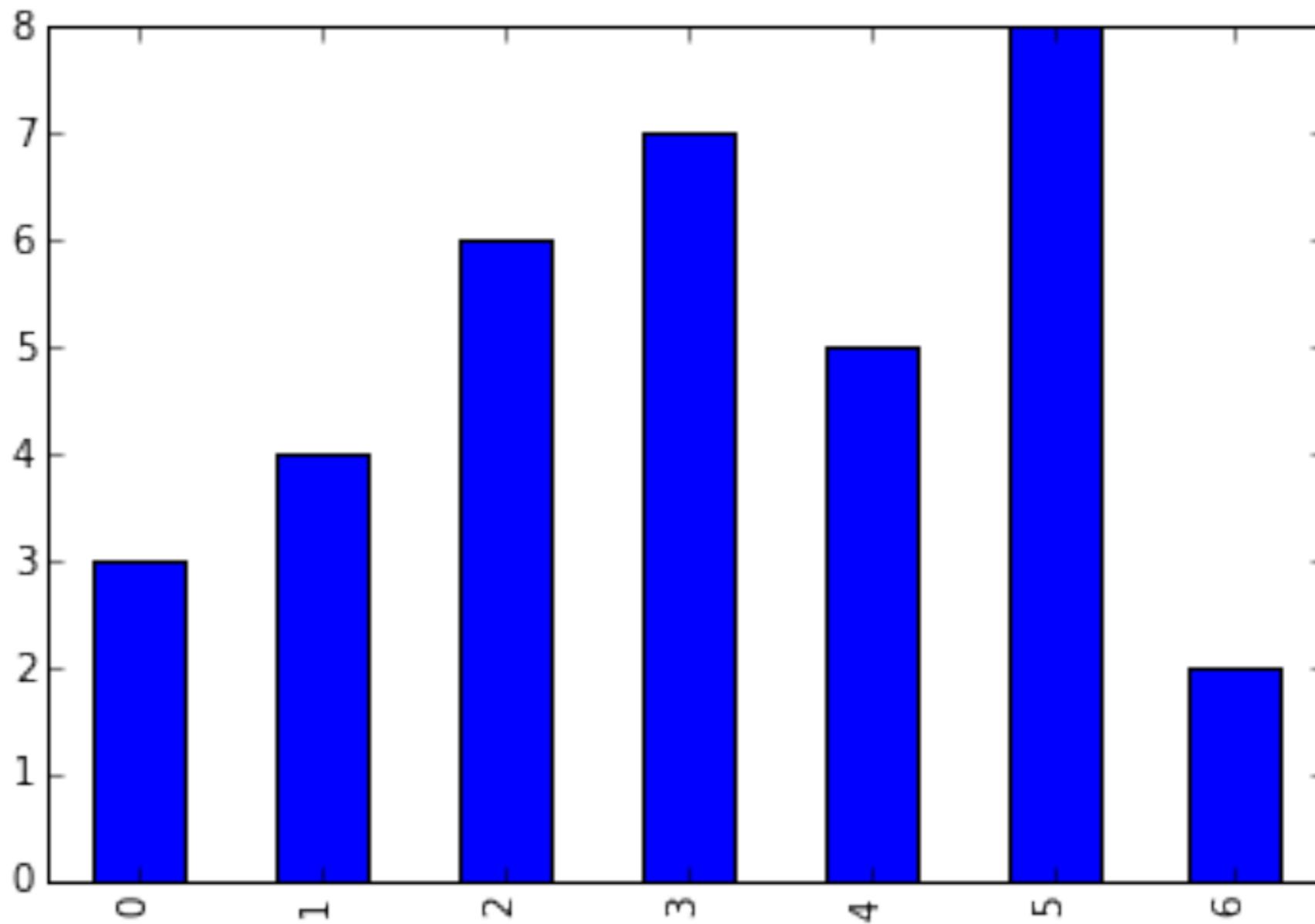
```
df.plot(kind='hexbin', x='a', y='b',  
gridsize=25)
```

```
scatterPlot.get_figure().savefig( "scatterPlot.png" )
```

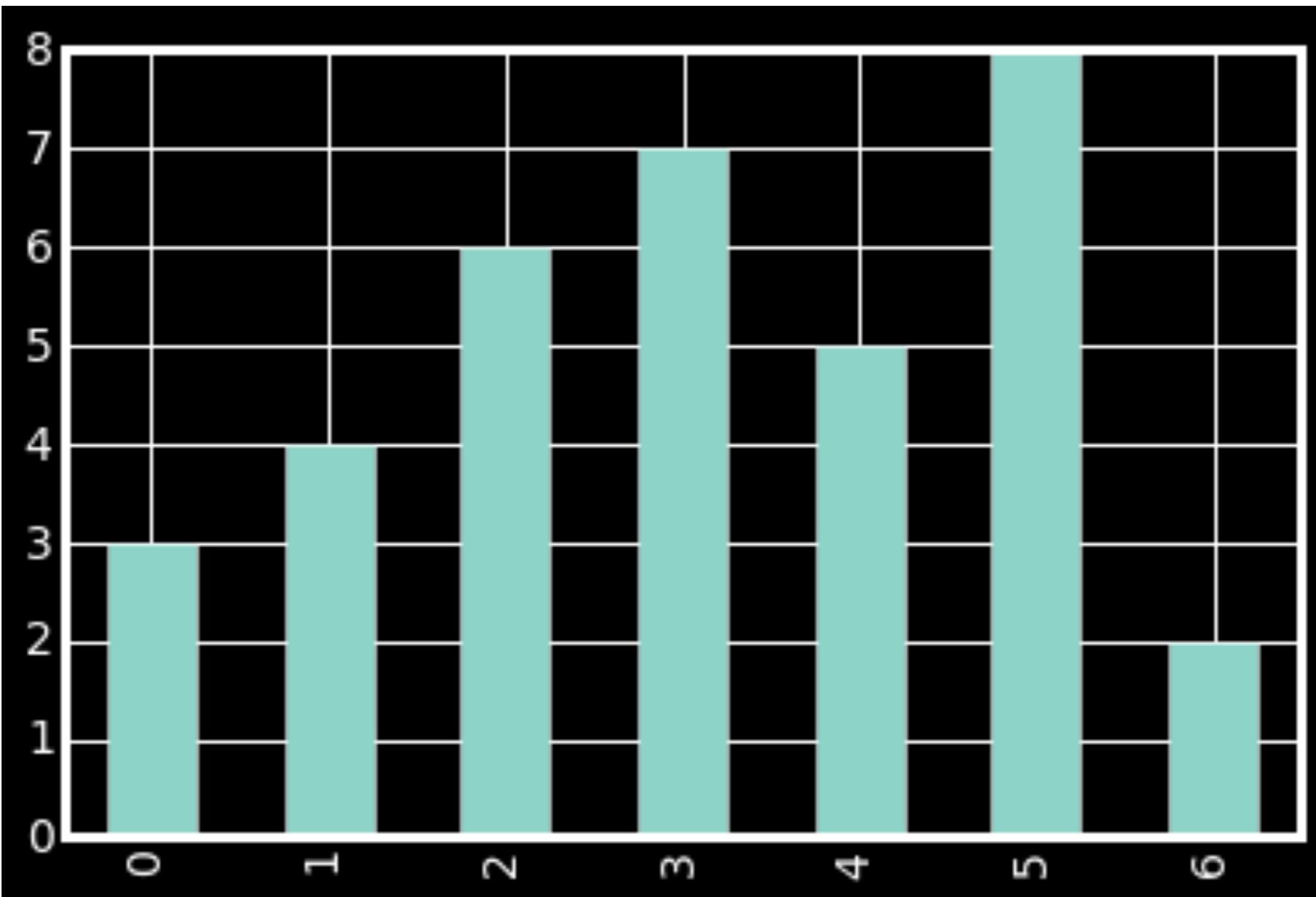
UGLY

```
print(plt.style.available)

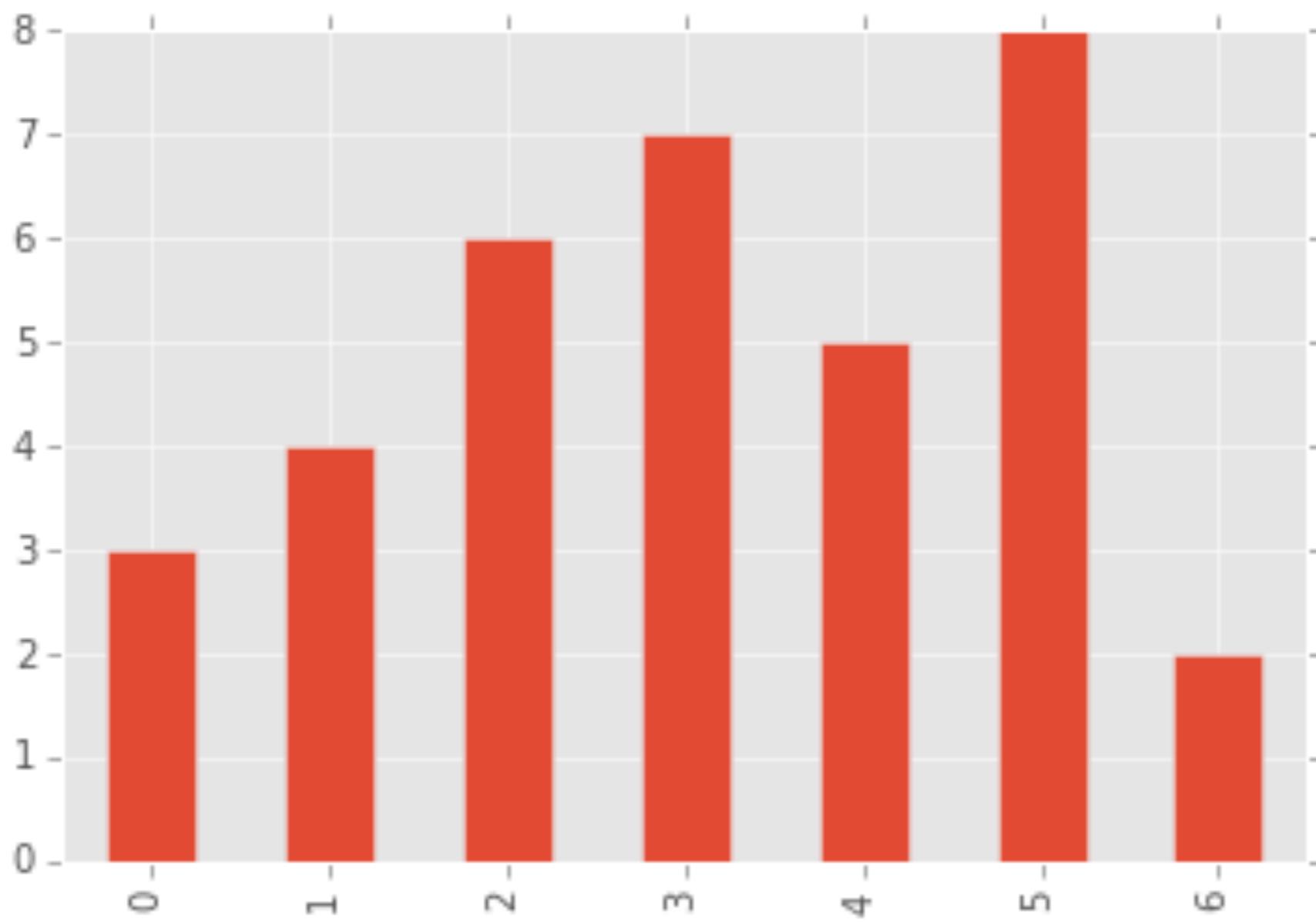
[ 'dark_background' , 'grayscale' , 'ggplot' ,
  'bmh' , 'fivethirtyeight' ]
```



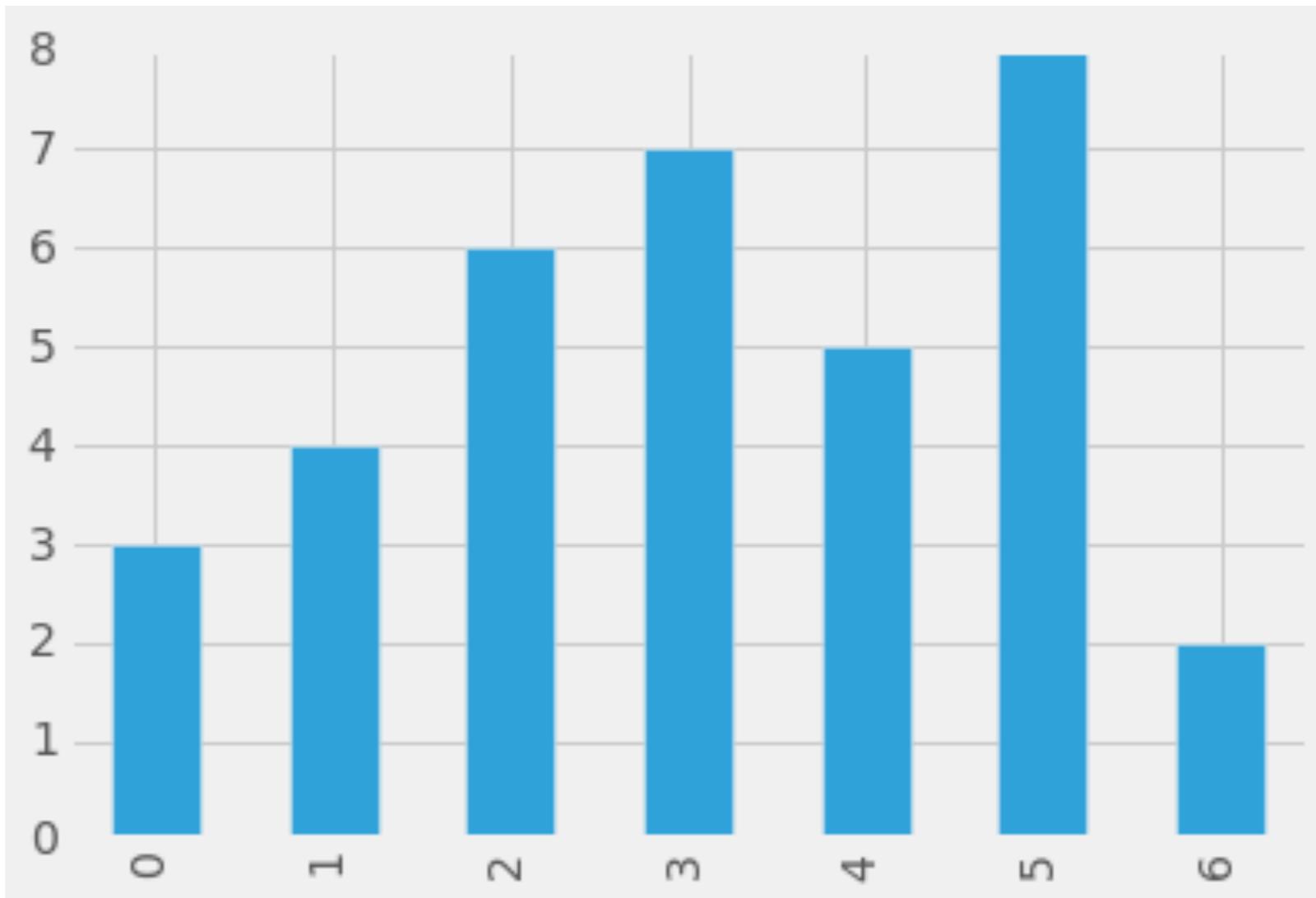
```
data = pd.Series( [3,4,6,7,5,8,2] )
barchart = data.plot( kind="bar" )
```



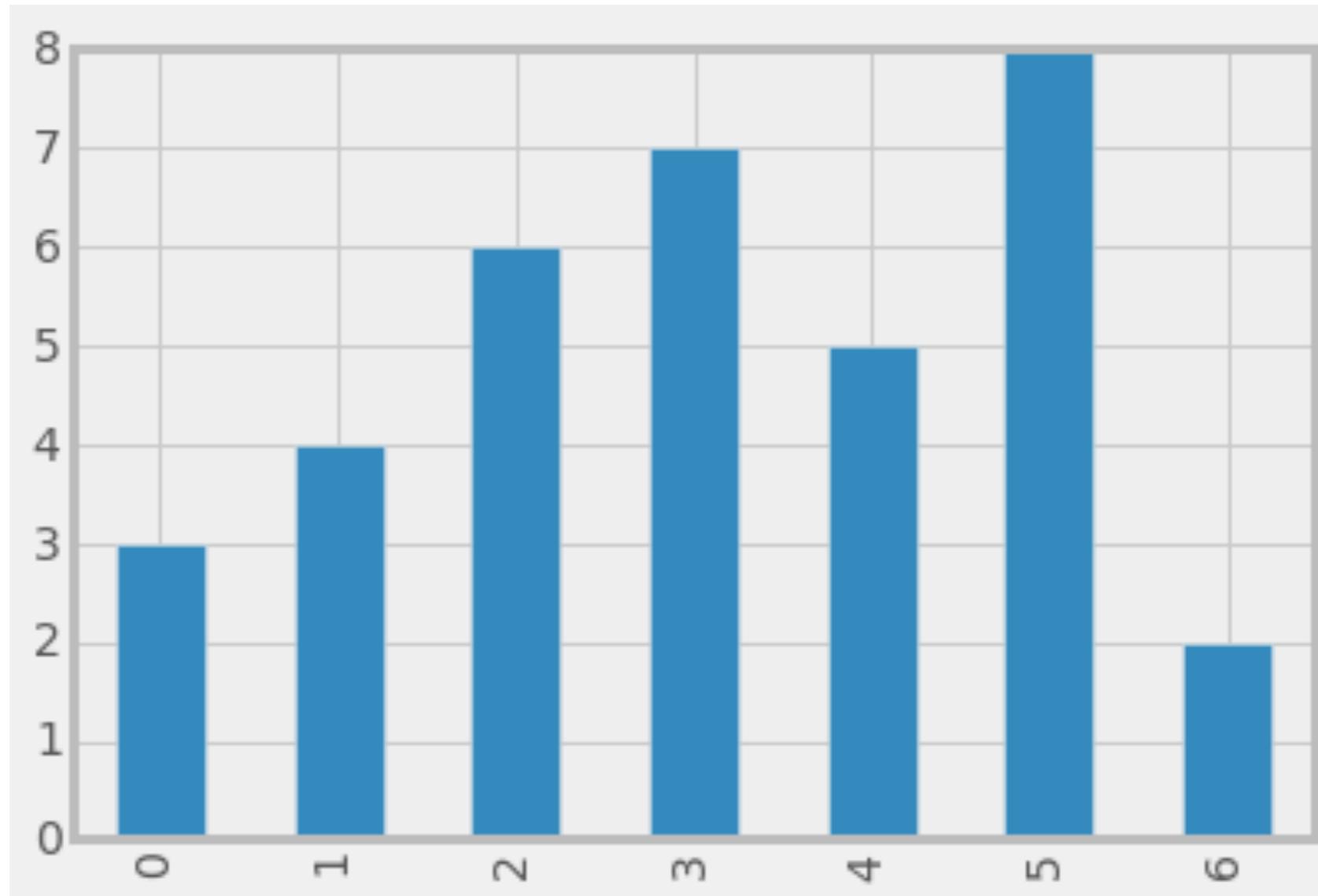
```
plt.style.use('dark_background')
barchart = data.plot( kind="bar" )
```



```
plt.style.use('ggplot')
barchart = data.plot( kind="bar" )
```



```
plt.style.use('fivethirtyeight')
barchart = data.plot( kind="bar" )
```



```
plt.style.use('bmh')
barchart = data.plot(kind="bar")
```

Interactive Visualizations



03



JavaScript







Easy to use... if you know R

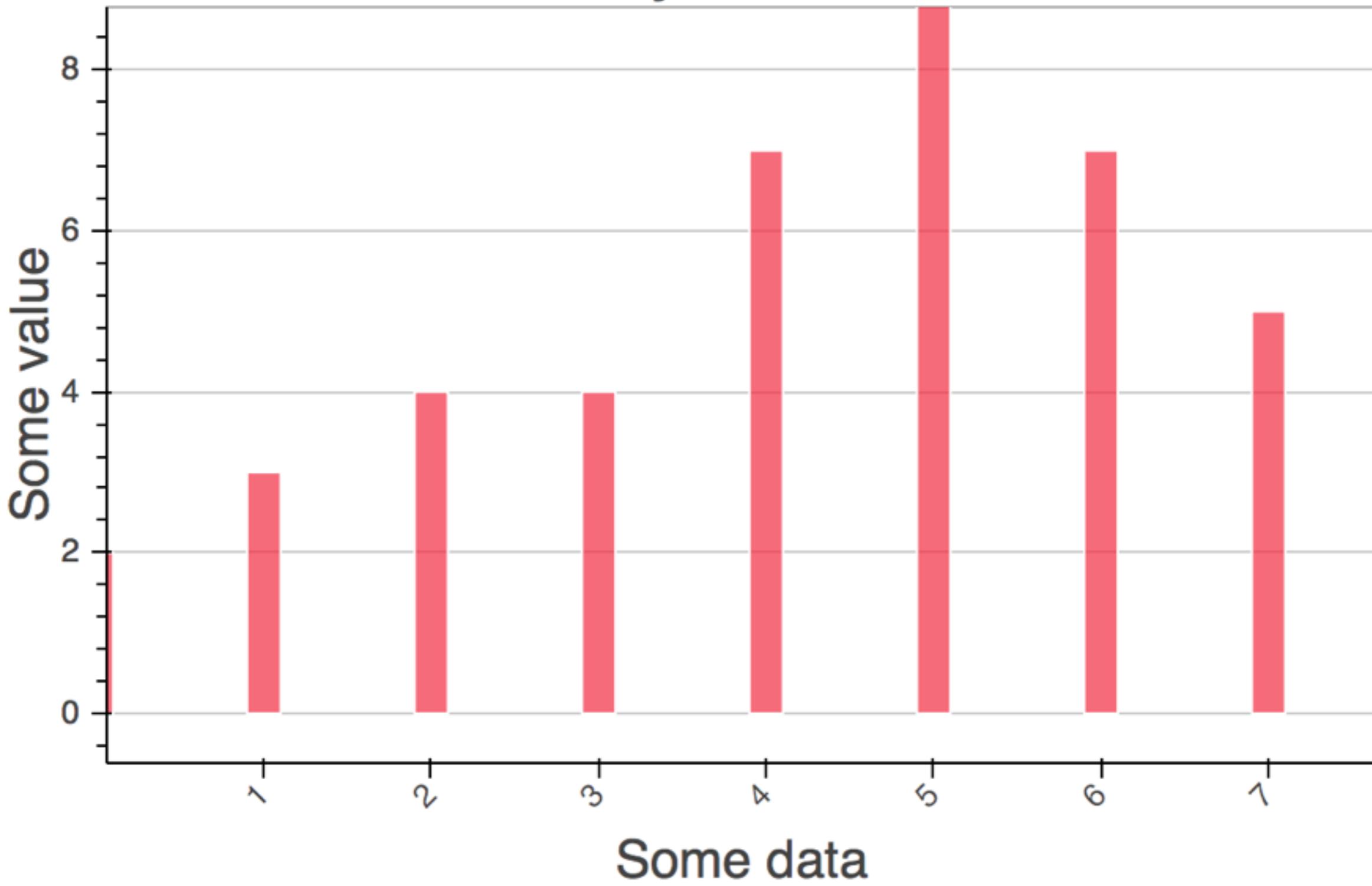
Not Free



Introducing Bokeh

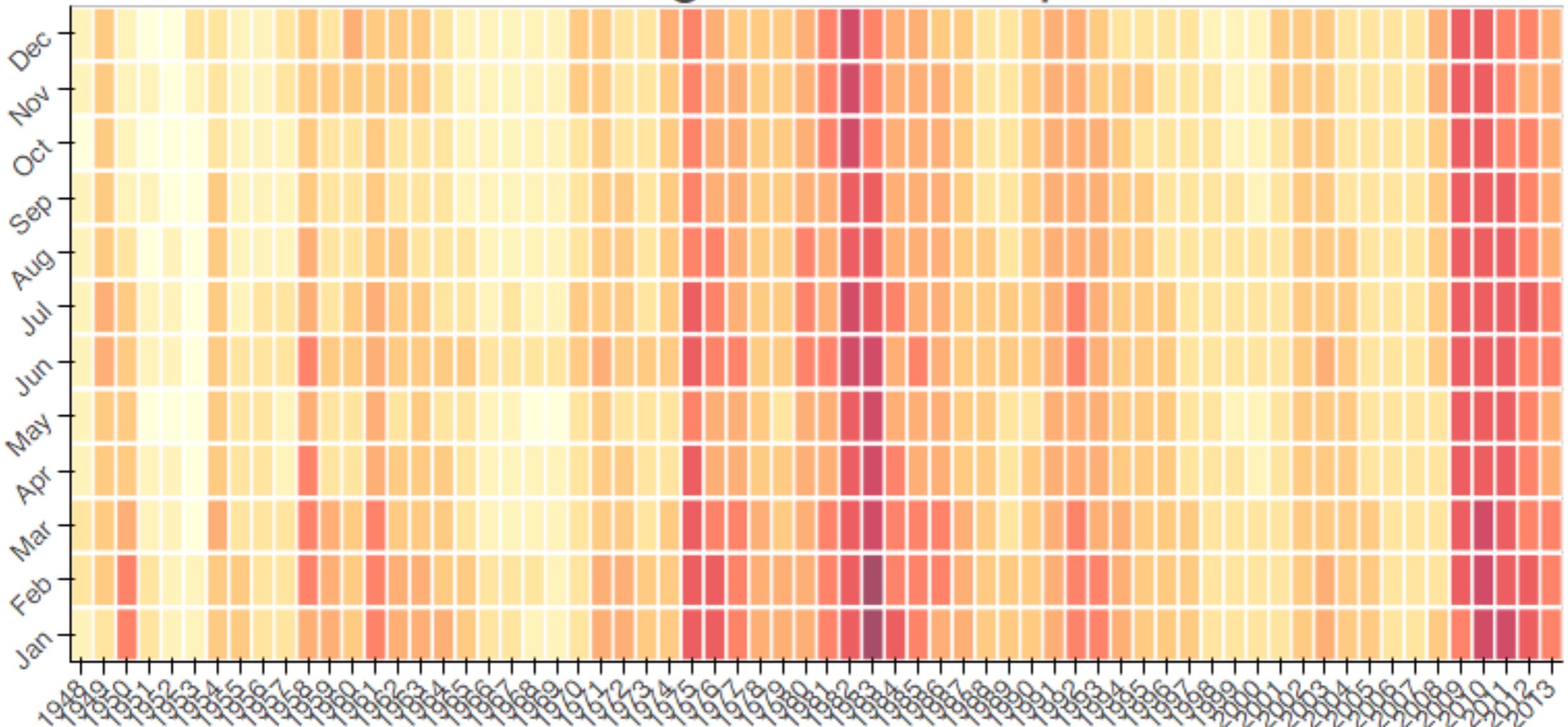


My Chart





categorical heatmap



http://bokeh.pydata.org/en/latest/docs/gallery/cat_heatmap_chart.html

```
from bokeh.charts import HeatMap, output_file, show
from bokeh.palettes import YlOrRd9 as palette
from bokeh.sampledata.unemployment1948 import data

# pandas magic
df = data[data.columns[:-1]]
df2 = df.set_index(df[df.columns[0]].astype(str))
df2.drop(df.columns[0], axis=1, inplace=True)
df3 = df2.transpose()

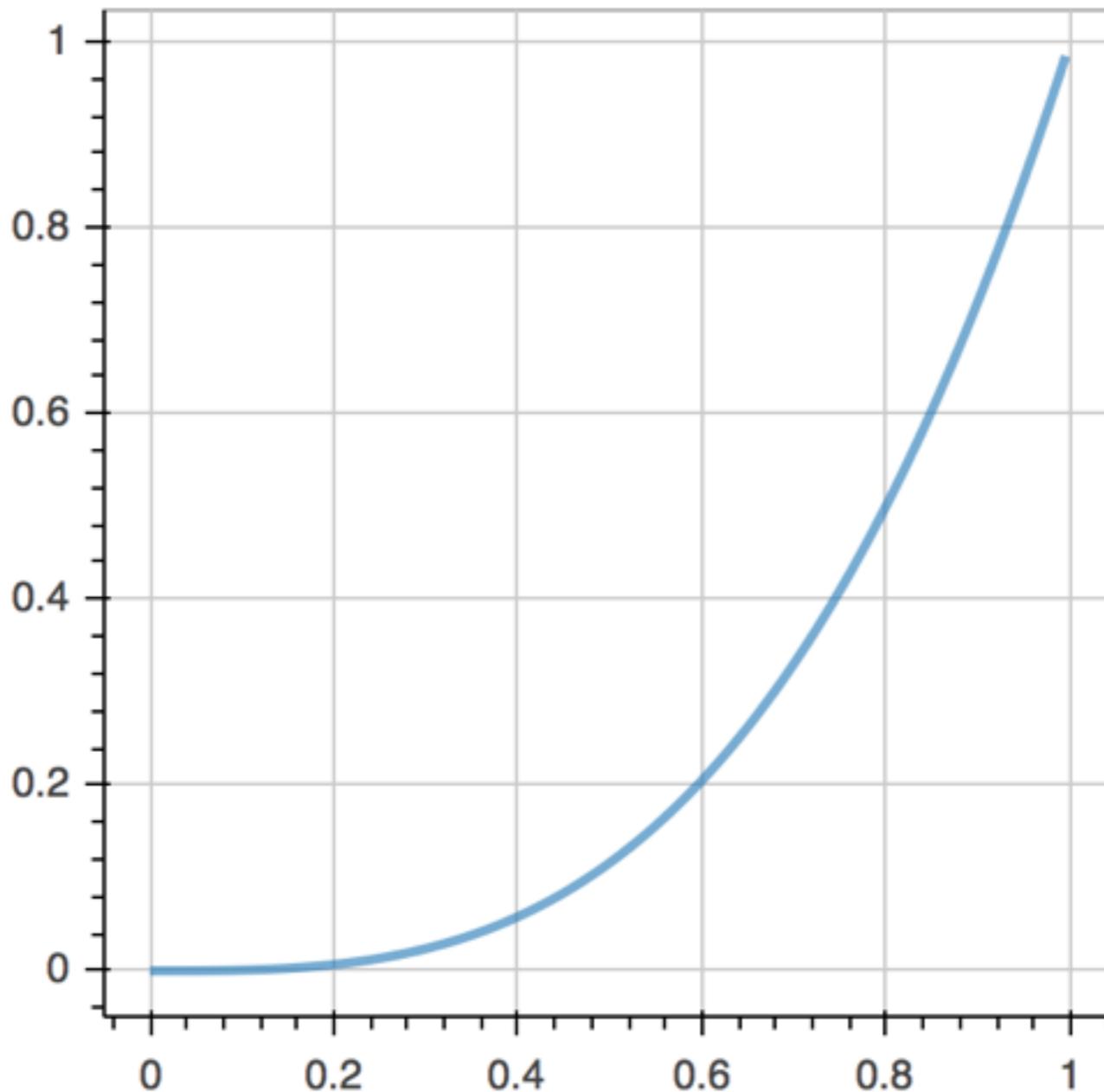
output_file("cat_heatmap.html")

palette = palette[::-1] # Reverse the color order so dark red
is highest unemployment
hm = HeatMap(df3, title="categorical heatmap", width=800,
palette=palette)

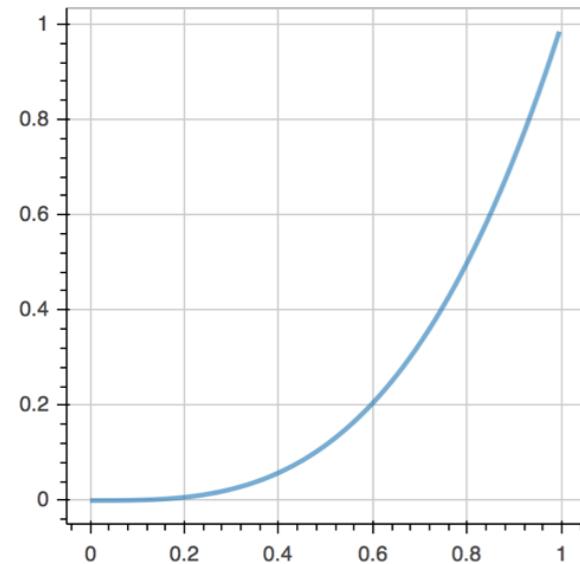
show(hm)
```

http://bokeh.pydata.org/en/latest/docs/gallery/cat_heatmap_chart.html

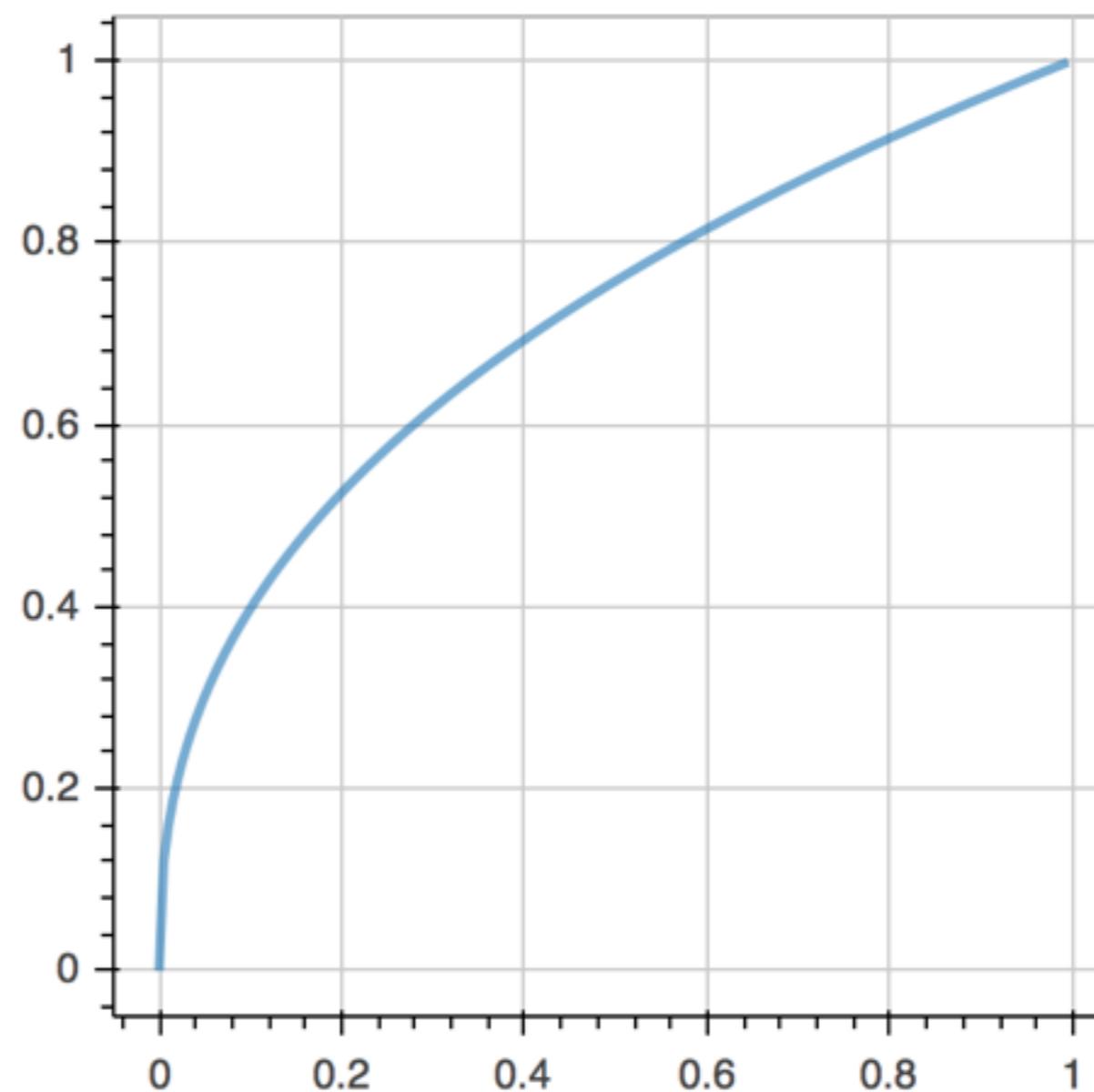
power: 3.1



power: 3.1



power: 0.4



```
from bokeh.io import vform
from bokeh.models import Callback, ColumnDataSource, Slider
from bokeh.plotting import figure, output_file, show

x = [x*0.005 for x in range(0, 200)]
y = x

source = ColumnDataSource(data=dict(x=x, y=y))

plot = figure(plot_width=400, plot_height=400)
plot.line('x', 'y', source=source, line_width=3, line_alpha=0.6)

callback = Callback(args=dict(source=source), code="""
    var data = source.get('data');
    var f = cb_obj.get('value')
    x = data['x']
    y = data['y']
    for (i = 0; i < x.length; i++) {
        y[i] = Math.pow(x[i], f)
    }
    source.trigger('change');
""")
slider = Slider(start=0.1, end=4, value=1, step=.1, title="power", callback=callback)

layout = vform(slider, plot)

show(layout)
```

Using Bokeh

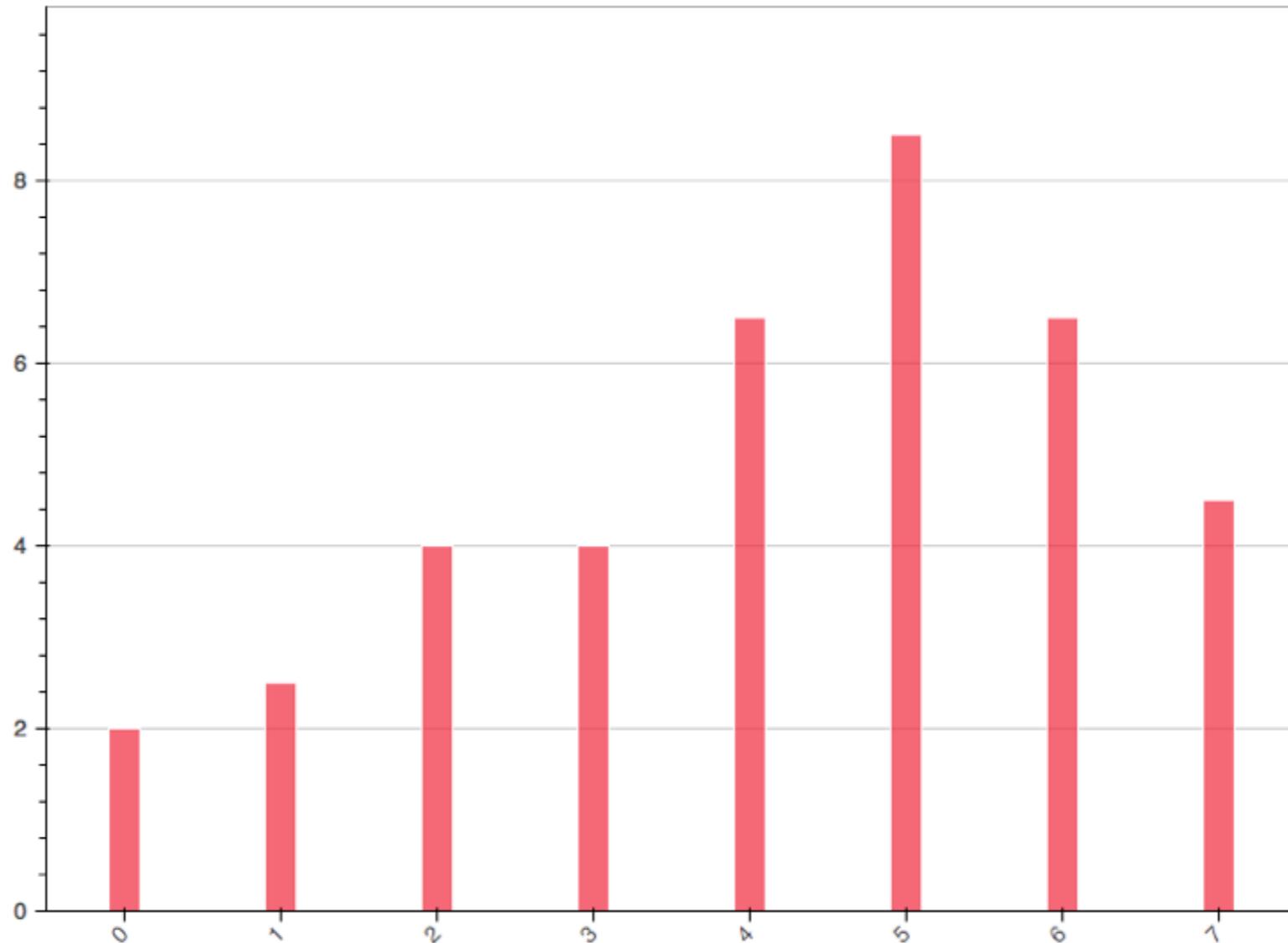
```
from bokeh.plotting import output_notebook  
output_notebook()
```

- Area (Overlapped & Stacked)
- Bar (Grouped & Stacked)
- BoxPlot
- Donut (Ick!)
- HeatMap
- Line
- Scatter
- Step
- Timeseries

```
from bokeh.charts import <chartname>, show
```

```
from bokeh.charts import Bar, show  
data = [2,3,4,4,7,9,7,5]  
  
barchart = Bar( data, notebook=True )  
show( )
```

```
from bokeh.charts import Bar, show  
data = [2,3,4,4,7,9,7,5]  
  
barchart = Bar( data, notebook=True )  
show( )
```





Method Chaining

```
barchart =  
    Bar( data, title="My Chart", xlabel="Categories", ylabel="Value",  
notebook=True )  
  
barchart.show()
```

```
barchart =  
    Bar( data, title="My Chart", xlabel="Categories", ylabel="Value",  
notebook=True )  
  
barchart.show()
```

```
barchart3 = Bar(data)  
.title( "My Chained Chart" )  
.notebook( True )  
.legend( True )  
.xlabel( "Categories" )  
.ylabel( "Value" )  
  
barchart3.show()
```

Bokeh Config Options

- **title** (str): the title of your plot.
- **xlabel** (str): the x-axis label of your plot.
- **ylabel** (str): the y-axis label of your plot.
- **legend** (str, bool): the legend of your plot.
- **xscale** (str): the x-axis type scale of your plot
- **yscale** (str): the y-axis type scale of your plot.
- **width** (int): the width of your plot in pixels.
- **height** (int): the height of you plot in pixels.
- **tools** (bool): to enable or disable the tools in your plot.
- **filename** (str or bool): the name of the file where your plot will be written.
- **server** (str or bool): the name of your plot in the server.
- **notebook** (bool): if you want to output (or not) your plot into the IPython notebook.

In Class Exercise

Please complete Data Visualization Worksheet

```
#Get the data into two seperate data frames
start = datetime.datetime(2015, 1, 1)
end = datetime.datetime(2015, 7, 27)
tsla = web.DataReader("TSLA", 'google', start, end)
lnkd = web.DataReader("LNKD", 'google', start, end)

#Remove unneeded columns from the TSLA dataframe
tsla = tsla.drop( ['Open', 'High', 'Low', 'Volume'], axis=1)

#Rename the Closing price column to TSLA
tsla['TSLA'] = tsla.Close

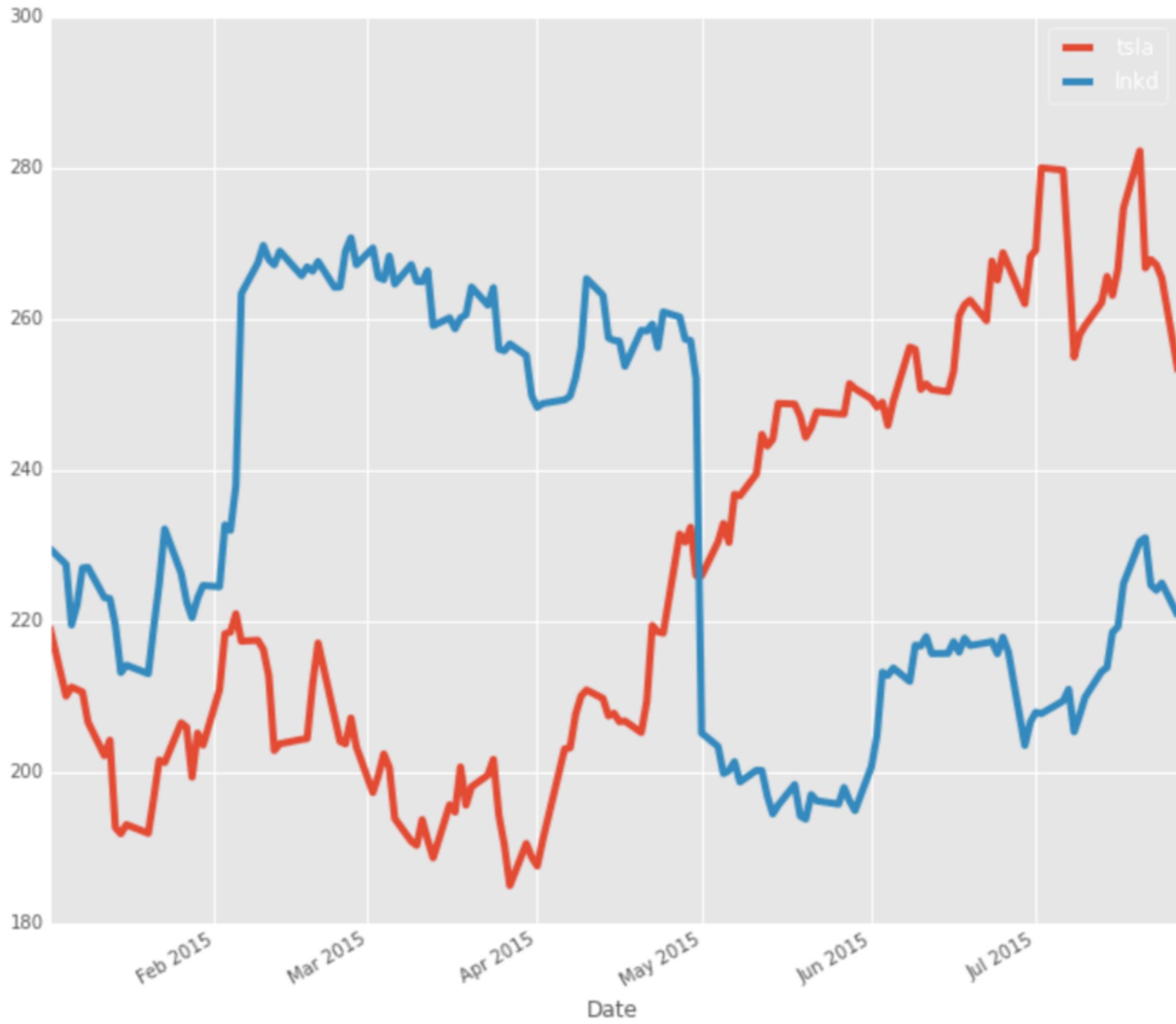
#Drop the Close column
tsla = tsla.drop( 'Close', axis=1)

#Remove the unneeded columns in the LinkedIN DataFrame
lnkd = lnkd.drop( ['Open', 'High', 'Low', 'Volume'], axis=1)

#Add a LNKD column to the TSLA dataframe
tsla['lnkd'] = lnkd.Close

#Draw the chart
tsla.plot(figsize=(20,10))
```

Date	tsla	Inkd
2015-01-02	219.31	229.65
2015-01-05	210.09	227.51
2015-01-06	211.28	219.51
2015-01-07	210.95	222.16
2015-01-08	210.62	227.05



BASED ON YOUR
INTERNET HISTORY,
YOU MIGHT BE DUMB
ENOUGH TO ENJOY
EXTREME SPORTS.



CLICK HERE TO BUY A
TICKET TO BASE JUMP
FROM THE INTERNA-
TIONAL SPACE STATION.



Dilbert.com DilbertCartoonist@gmail.com

I THINK
THE INTER-
NET IS
TRYING TO
KILL ME.



WE
CALL IT
"MACHINE
LEARNING."



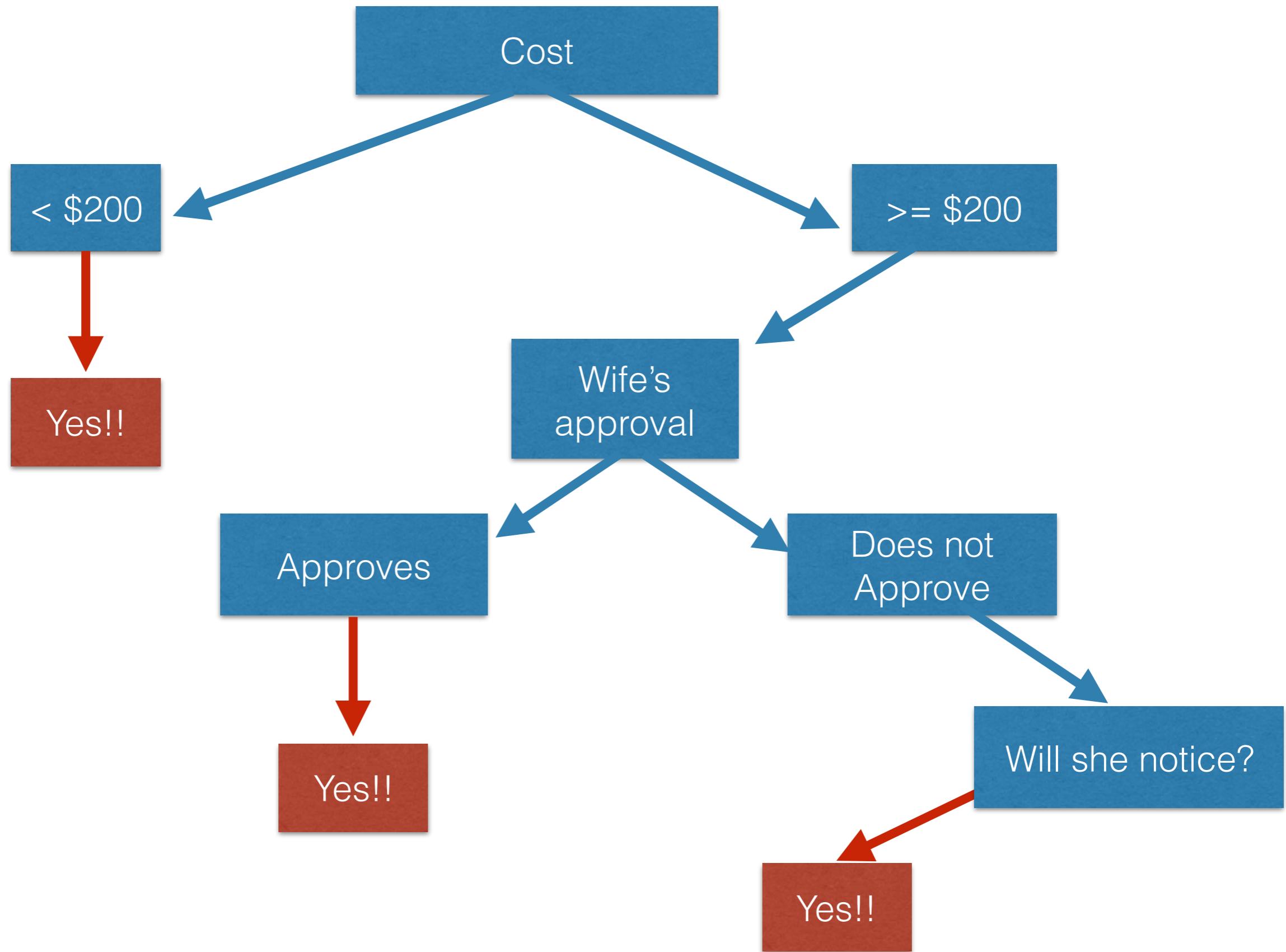
2-2-13 © 2013 Scott Adams, Inc./Dist. by Universal Uclick

A photograph of a lush tropical forest. In the foreground, the ground is covered with fallen brown leaves. Several large, mature trees with thick trunks and sprawling root systems are prominent. Sunlight filters through the dense canopy of green leaves, creating bright highlights and deep shadows. The overall atmosphere is serene and natural.

Classification with Decision Trees

Decision Trees are a tool which use a tree-like graph to model a series of decisions and their consequences.

Should I buy a new tech gadget?



Entropy

Entropy

- A measure of how “pure” the a dataset is
- 0 indicates completely homogenous, and 1 means that the sample is equally divided

$$H = - \sum p(x) \log p(x)$$

Information Gain

- A metric to determine how predictive a particular feature is. 1 being the most predictive, and 0 being not predictive.

$$GAIN(T, X) = Entropy(T) - Entropy(T, X)$$

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	TRUE	No

Example from: http://www.saedsayad.com/decision_tree.htm

Play Golf

Yes	No
9	5

$$\begin{aligned}\text{Entropy(PlayGolf)} &= \text{Entropy}(5, 9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= (0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$

Example from: http://www.saedsayad.com/decision_tree.htm

$$GAIN(T, X) = Entropy(T) - Entropy(T, X)$$

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

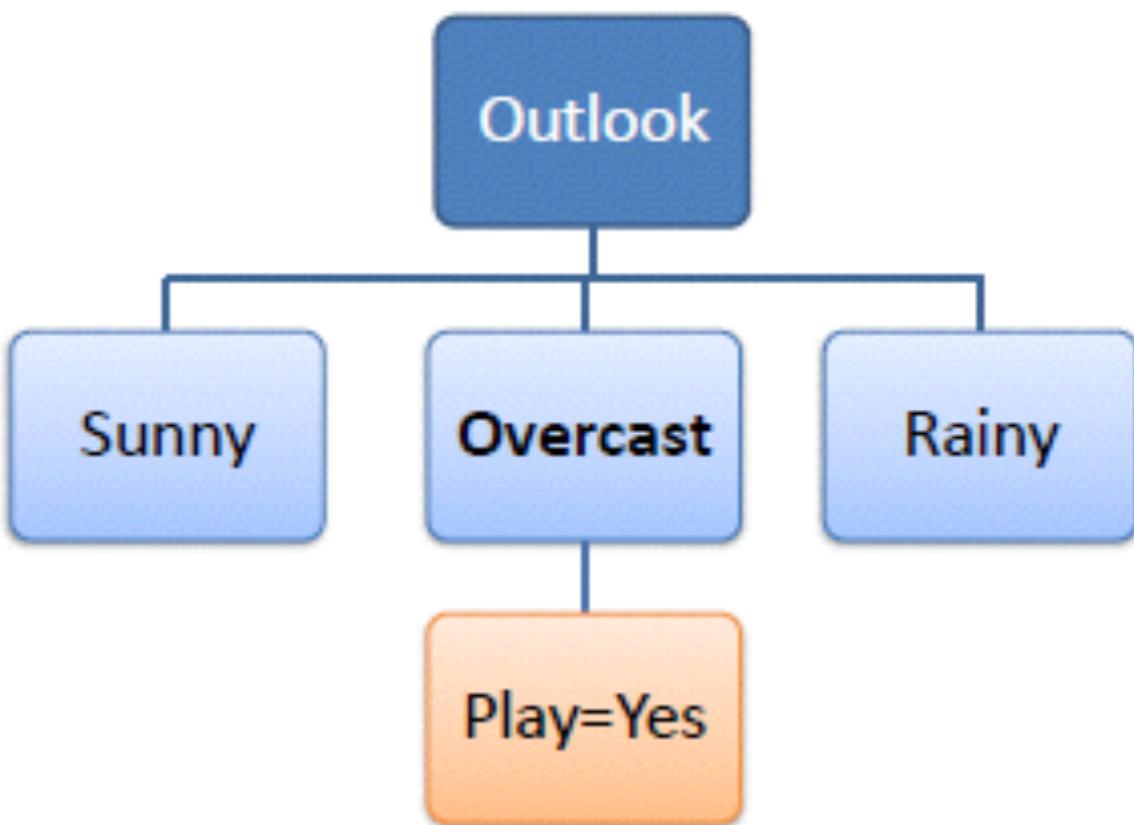
		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

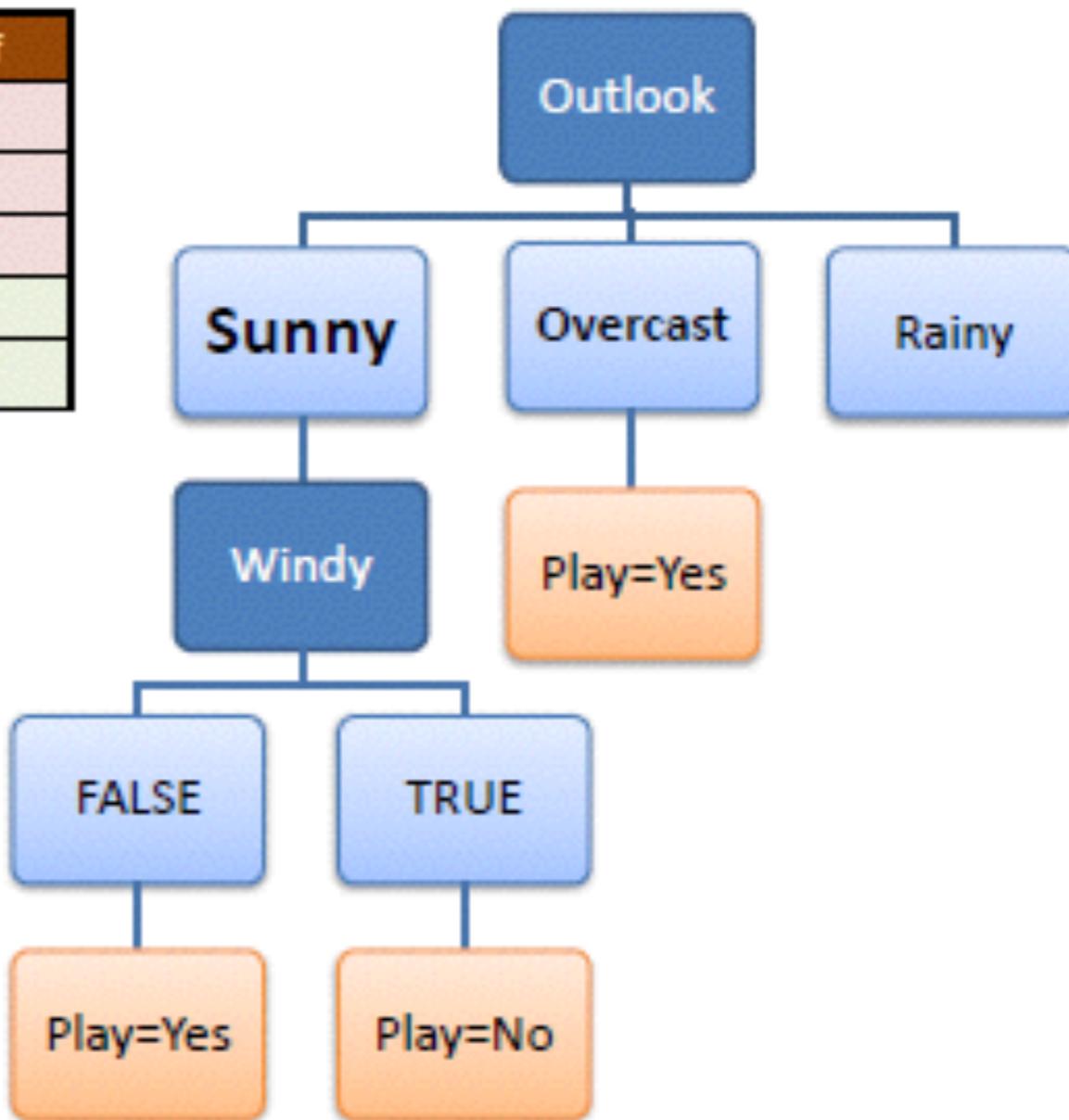
Example from: http://www.saedsayad.com/decision_tree.htm

Temp.	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes
Hot	High	FALSE	Yes



Example from: http://www.saedsayad.com/decision_tree.htm

Temp.	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Example from: http://www.saedsayad.com/decision_tree.htm

One Hot Encoding

One Hot Encoding

Color
Red
Red
Blue
Green
Yellow
Red

One Hot Encoding

Color	Color=Red	Color=Blue	Color=Yellow	Color=Green
Red	1	0	0	0
Red	1	0	0	0
Blue	0	1	0	0
Green	0	0	0	1
Yellow	0	0	1	0
Red	1	0	0	0

```
%matplotlib inline  
import numpy as np  
import pandas as pd  
  
from sklearn import feature_extraction  
from sklearn import tree
```

```
data = pd.read_csv( 'Data/playgolf.csv' )
```

	Outlook	Temp	Humidity	Windy	Play_Golf
0	Rainy	Hot	High	FALSE	No
1	Rainy	Hot	High	TRUE	No
2	Overcast	Hot	High	FALSE	Yes
3	Sunny	Mild	High	FALSE	Yes
4	Sunny	Cool	Normal	FALSE	Yes
5	Sunny	Cool	Normal	TRUE	No
6	Overcast	Cool	Normal	TRUE	Yes
7	Rainy	Mild	High	FALSE	No
8	Rainy	Cool	Normal	FALSE	Yes
9	Sunny	Mild	Normal	FALSE	Yes
10	Rainy	Mild	Normal	TRUE	Yes
11	Overcast	Mild	High	TRUE	Yes
12	Overcast	Hot	High	FALSE	Yes
13	Sunny	Mild	High	TRUE	No

```
cat_columns = ['Outlook', 'Temp', 'Humidity', 'Windy']
cat_dict = data[cat_columns].to_dict(orient='records')

[
    {
        'Windy': False,
        'Temp': 'Hot',
        'Humidity': 'High',
        'Outlook': 'Rainy'
    },
    {
        'Windy': True,
        'Temp': 'Hot',
        'Humidity': 'High',
        'Outlook': 'Rainy'
    }...
]
```

```
vec = feature_extraction.DictVectorizer()
cat_vector = vec.fit_transform(cat_dict).toarray()
```

```
[ [ 1.  0.  0.  1.  0.  0.  1.  0.  0.]
  [ 1.  0.  0.  1.  0.  0.  1.  0.  1.]
  [ 1.  0.  1.  0.  0.  0.  1.  0.  0.]
  [ 1.  0.  0.  0.  1.  0.  0.  1.  0.]
  [ 0.  1.  0.  0.  1.  1.  0.  0.  0.]
  [ 0.  1.  0.  0.  1.  1.  0.  0.  1.]
  [ 0.  1.  1.  0.  0.  1.  0.  0.  1.]
  [ 1.  0.  0.  1.  0.  0.  0.  1.  0.]
  [ 0.  1.  0.  1.  0.  1.  0.  0.  0.]
  [ 0.  1.  0.  0.  1.  0.  0.  1.  0.]
  [ 0.  1.  0.  1.  0.  0.  0.  1.  1.]
  [ 1.  0.  1.  0.  0.  0.  0.  1.  1.]
  [ 1.  0.  1.  0.  0.  0.  1.  0.  0.]
  [ 1.  0.  0.  0.  1.  0.  0.  1.  1.] ]
```

```
df_vector = pd.DataFrame(cat_vector)
vector_columns = vec.get_feature_names()
df_vector.columns = vector_columns
df_vector.index = data.index
```

	Humidity=High	Humidity=Normal	Outlook=Overcast	Outlook=Rainy	\
0	1	0	0	1	
1	1	0	0	1	
2	1	0	1	0	
3	1	0	0	0	
4	0	1	0	0	
5	0	1	0	0	
6	0	1	1	0	
7	1	0	0	0	1
8	0	1	0	0	1
9	0	1	0	0	0
10	0	1	0	0	1
11	1	0	1	0	0
12	1	0	0	1	0
13	1	0	0	0	0

```
data = data.drop(cat_columns, axis=1)
data = data.join(df_vector)
```

	Play_Golf	Humidity=High	Humidity=Normal	Outlook=Overcast	Outlook=Rainy
0	No	1	0	0	1
1	No	1	0	0	1
2	Yes	1	0	1	0
3	Yes	1	0	0	0
4	Yes	0	1	0	0

```
data.loc[data.Play_Golf == 'No', 'decision'] = 0  
data.loc[data.Play_Golf == 'Yes', 'decision'] = 1
```

```
# Split the data set into features and labels
features = data.drop(['Play_Golf'], axis=1)
labels = data.Play_Golf
```

```
test_features = features[-1:]
test_label = labels[-1:]

# Train the decision tree based on the entropy
criterion
clf = tree.DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(features[:-1], labels[:-1])
```

```
# Make a prediction with test data
pred = clf.predict(test_features)
print(features[-1:].T)
print('Predicted class:', pred)
print('Accurate prediction?', pred[0] == test_label.values[0])
```

```
# Make a prediction with test data
pred = clf.predict(test_features)
print(features[-1:].T)
print('Predicted class:', pred)
print('Accurate prediction?', pred[0] == test_label.values[0])
```

	13
Humidity=High	1
Humidity=Normal	0
Outlook=Overcast	0
Outlook=Rainy	0
Outlook=Sunny	1
Temp=Cool	0
Temp=Hot	0
Temp=Mild	1
Windy	1
decision	0
Predicted class:	['No']
Accurate prediction?	True

URL Classification Challenge

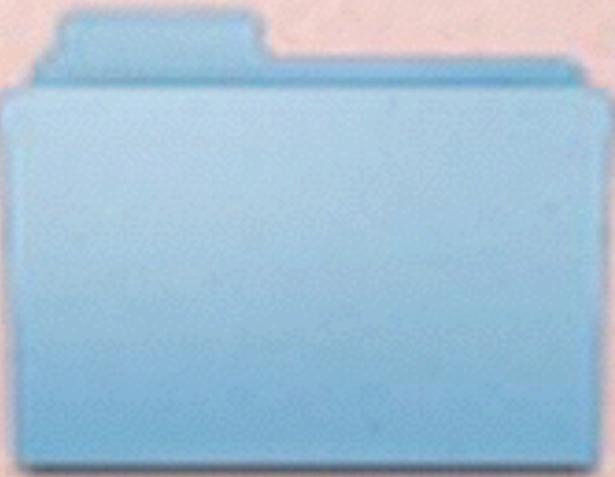
Whitelist

Blacklist

facebook.com

facebook.com

facebook.com.ru



not porn

Seems legit

facebook.com.ru

- Has 2 digits
- Has 2 periods
- Has .ru suffix

facebook.com

- Has 0 digits
- Has 1 period
- Has .com suffix

<http://cseweb.ucsd.edu/~savage/papers/ICML09.pdf>

In Class Exercise

Complete URL Decision Tree Worksheet, Steps 1-5

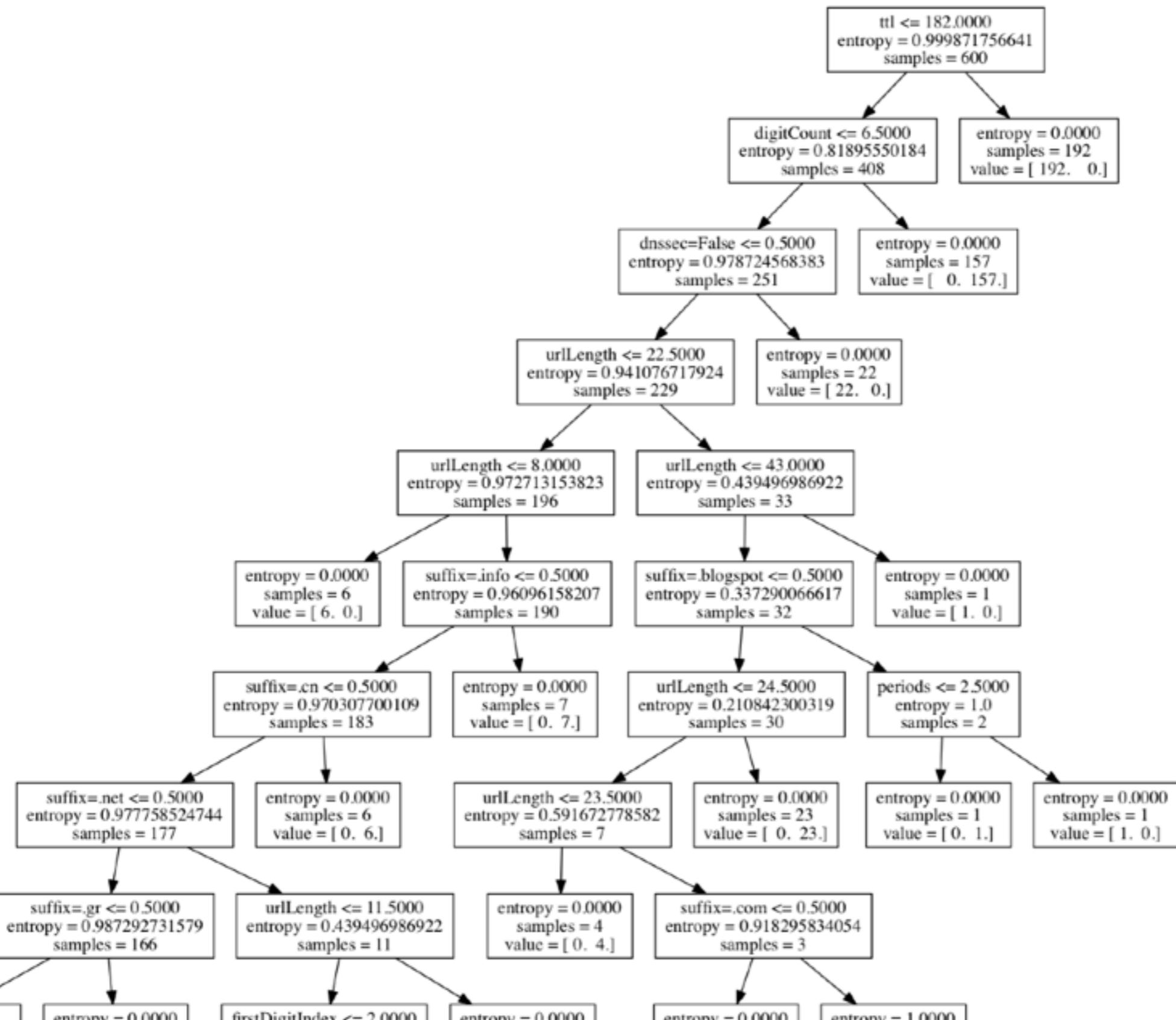
Visualizing your Tree

```
from sklearn.externals.six import StringIO
from IPython.core.display import Image
import pydotplus as pydot

dot_data = StringIO()
tree.export_graphviz(
    clf,
    out_file=dot_data,
    feature_names=features.columns)

graph = pydot.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())
```



Validating Your Model

Accuracy

Accuracy

A measurement of the proportion of instances correctly classified by the classifier.

Accuracy

A measurement of the proportion of instances correctly classified by the classifier.

$$(True\ Positives + True\ Negatives) / Sample\ Size$$

```
def hasCancer( x ):  
    return false  
  
hasCancer( humanRace )
```

Precision

Precision

The proportion of instance predicted as positives than were correctly evaluated.

Precision

The proportion of instance predicted as positives than were correctly evaluated.

$$\textit{Precision} = \textit{True Positive} / (\textit{True Positives} + \textit{False Positives})$$

Recall

Recall

The proportion of positive instances that were correctly evaluated.

Recall

The proportion of positive instances that were correctly evaluated.

$$\textit{Recall} = \textit{True Positives} / (\textit{True Positives} + \textit{False Negatives})$$

F1-Score

F1-Score

Combines precision and recall into a single number

F1-Score

Combines precision and recall into a single number

$$F1\text{-Score} = 2 * Precision * Recall / (Precision + Recall)$$

Validating your Model

Now that you have a model, the next step is to assess how accurate the model is.

- 1 Call the `.predict()` method with your training data and store the results in a new dataframe called `training_predictions`.
- 2 Next, call the `metrics.accuracy_score()` method using your training target data and the predictions you just made.
- 3 Print out the result.

```
from sklearn import metrics
```

```
metrics.accuracy_score( training_labels,  
predictions_from_training_data )  
  
metrics.classification_report(  
    training_labels,  
    predictions_from_training_data,  
    target_names=[ 'Not Malicious', 'Malicious' ]  
)
```

```
clf.score(testing_features, testing_labels)
```

```
from sklearn.cross_validation import cross_val_score, KFold
from sklearn.pipeline import Pipeline
from scipy.stats import sem

cross_validator = KFold( features.shape[0], 5, shuffle=True,
random_state=33)
```

In Class Exercise

Complete URL Decision Tree Worksheet, Step 6

LET'S SOLVE THIS PROBLEM BY
USING THE BIG DATA NONE
OF US HAVE THE SLIGHTEST
IDEA WHAT TO DO WITH



TOM
FISH
BURNE

Data

Data

I LOVE IT WHEN



YOU CALL ME BIG
DATA

What is big data?

Big data generally refers to data sets which are too large to process using conventional applications.

Volume: The actual amount of data

Volume: The actual amount of data

Velocity: The speed at which the data is generated

Volume: The actual amount of data

Velocity: The speed at which the data is generated

Variety: Data arrives in many different forms

<http://onesecond.designly.com>

















Cloud Computing



MapReduce

<http://research.google.com/archive/mapreduce.html>

MapReduce in Python

```
import sys

for line in sys.stdin:
    line = line.strip()
    keys = line.split()
    for key in keys:
        value = 1
        print( "%s\t%d" % (key, value) )
```

MapReduce in Python

away--you 1
may 1
do 1
practically 1
ANYTHING 1
with 1
public 1
domain 1
eBooks. 1
Redistribution 1
is 1
subject 1

MapReduce in Python

```
import sys

last_key = None
running_total = 0

for input_line in sys.stdin:
    input_line = input_line.strip()
    this_key, value = input_line.split("\t", 1)
    value = int(value)

    if last_key == this_key:
        running_total += value
    else:
        if last_key:
            print( "%s\t%d" % (last_key, running_total) )
        running_total = value
        last_key = this_key

if last_key == this_key:
    print( "%s\t%d" % (last_key, running_total) )
```

MapReduce in Python

```
cat sherlock.txt | ./map.py | sort | ./reduce.py
```

```
available 1
avenue 1
avenue. 1
average 3
averse 5
aversion 2
avert 3
averted 1
avoid 3
avoided 2
...
...
```

Map Reduce is Hard...Really Hard





YAHOO!

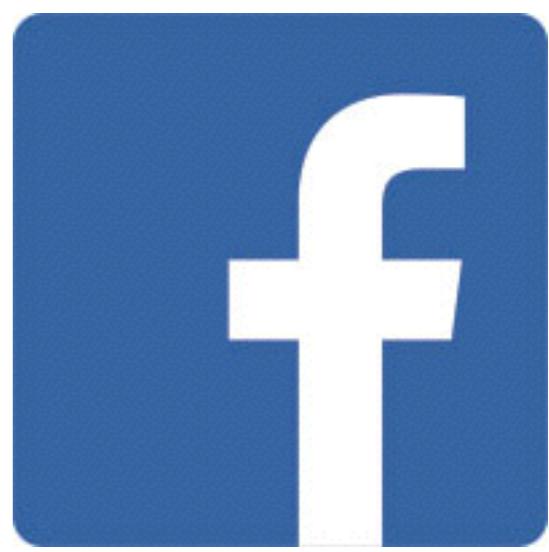
Pigs Eat Anything

Pigs Live Anywhere

Pigs Are Domestic
Animals

Pigs Fly

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS  
(line:chararray);  
  
-- Extract words from each line and put them into a pig bag  
-- datatype, then flatten the bag to get one word on each row  
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line))  
AS word;  
  
-- filter out any words that are just white spaces  
filtered_words = FILTER words BY word MATCHES '\\\\w+';  
  
-- create a group for each word  
word_groups = GROUP filtered_words BY word;  
  
-- count the entries in each group  
word_count = FOREACH word_groups GENERATE  
COUNT(filtered_words) AS count, group AS word;  
  
-- order the records by count  
ordered_word_count = ORDER word_count BY count DESC;  
STORE ordered_word_count INTO '/tmp/number-of-words-on-  
internet';
```



```
SELECT word, count(1) as count
FROM
(
    SELECT explode(split(sentence, ' ')) AS word
    FROM texttable
)tempTable
GROUP BY word
```



APACHE
DRILL

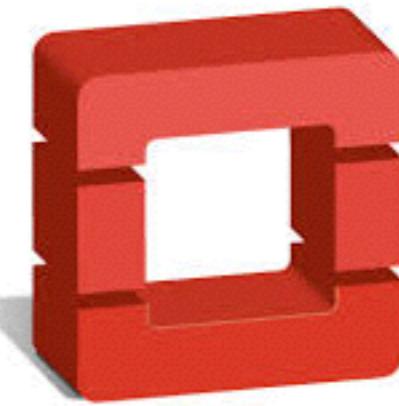
Apache Drill can
significantly increase
your access to data

Allows you to “query”
data with no schema

Drill supports
standard SQL

BFD?

Drill allows you to quickly
explore data sets of any
size





Install Drill

<https://drill.apache.org/>

Using Drill

```
[  
 {  
   "changed": "2015-04-03 00:00:00",  
   "created": "2014-04-10 00:00:00",  
   "dnssec": "False",  
   "expires": "2016-04-10 00:00:00",  
   "isMalicious": 0,  
   "url": "nuteczki.com"  
 }...  
 ]
```

localhost

Desktop/ Worksheet Tree Code KMeans wo... How-to: Us... ► J. S. Bac Inbox (Apache Drill . +

Apache Drill Query Profiles Storage Metrics Threads Options Documentation

Sample SQL query: `SELECT * FROM cp.`employee.json` LIMIT 20` ×

Query Type

SQL
 PHYSICAL
 LOGICAL

Query

```
SELECT * FROM bhfiles.`/Users/cgivre/OneDrive/Blackhat 2015 Training/Data/urlData2.json` LIMIT 10
```

Submit

localhost

Desktop/ Worksheet Tree Code KMeans wo... How-to: Us... ► J. S. Bac Inbox (Apache Drill . +

Apache Drill Query Profiles Storage Metrics Threads Options Documentation

Show 10 entries Search: Show / hide columns

changed	created	dnssec	expires	isMalicious	url
2015-04-03 00:00:00	2014-04-10 00:00:00	False	2016-04-10 00:00:00	0	nuteczki.com
2015-05-14 00:00:00	2015-01-31 00:00:00	False	2016-01-31 00:00:00	0	daftarcaramembuat.com
null	2009.06.11	null	2016-06-11 00:00:00	0	price59.ru
2014-11-20 03:51:55	null	False	null	0	bulinews.de
null	null	null	null	0	game.es
False	False	null	False	0	mailbox.pt
2014-02-05 00:00:00	: 2004-11-04T01:34:38Z	False	2015-11-04 00:00:00	0	buildyourownclone.com
2014-12-31 08:08:01	2002-12-05 00:00:00	False	null	0	nomura-tailor.co.jp
2014-10-31 09:14:09	1996-10-22 00:00:00	False	null	0	bunkei.co.jp
2014-03-12 04:19:36	null	False	null	0	fc-heidenheim.de

Drill Explorer

Schemas:

- .DS_Store
- apache-log-sample1.txt
- apache-log-sample2.txt
- apitruck.txt
- domainData.xlsx
- playgolf.csv
- sherlockHolmes.txt
- studentData.csv
- terrorismData.xlsx
- twitter1.csv
- url-blacklist
- url-blacklist-forML.csv
- url-blacklist-small.csv
- url-whitelist-reallysmall.csv
- url-whitelist-small.csv
- url-whitelist.csv
- urlData2.csv
- urlData2.json
- urlsForAustin.csv
- whitelist-urlData.json
- whitelist-whois.csv

► iPython Notebooks

► Scripts

- .DS_Store
- anomaly_log.py
- Blackhat 2015.pptx

Browse SQL

▼ Metadata:

Row

No content in table

▼ Preview:

Number of Rows: 100

Row	changed	created	dnssec	expires	isMa
1	2015-04-03 00:00:00	2014-04-10 00:00:00	False	2016-04-10 00:00:00	0
2	2015-05-14 00:00:00	2015-01-31 00:00:00	False	2016-01-31 00:00:00	0
3		2009.06.11	2009.06.11	2016-06-11 00:00:00	0
4	2014-11-20 03:51:55	2014-11-20 03:51:55	False	False	0
5					0
6	False	False	False	False	0
7	2014-02-05 00:00:00	: 2004-11-04T01:34:38Z	False	2015-11-04 00:00:00	0
8	2014-12-31 08:08:01	2002-12-05 00:00:00	False	False	0

Sample 100

Disconnect Connect

v1.1.0.1000

```
import requests
import pandas as pd
import json

url = "http://localhost:8047/query.json"
```

```
import requests
import pandas as pd
import json

url = "http://localhost:8047/query.json"

employee_query = """SELECT * FROM bhfiles.`urlData2.json`"""
```

```
import requests
import pandas as pd
import json

url = "http://localhost:8047/query.json"
employee_query = """SELECT * FROM bhfiles.`urlData2.json`"""

data = {"queryType" : "SQL", "query": employee_query }

data_json = json.dumps(data)
headers = { 'Content-type': 'application/json' }

response = requests.post(url, data=data_json,
headers=headers)
```

```
import requests
import pandas as pd
import json

url = "http://localhost:8047/query.json"
employee_query = """SELECT * FROM bhfiles.`urlData2.json`"""
data = {"queryType" : "SQL", "query": employee_query }

data_json = json.dumps(data)
headers = {'Content-type': 'application/json'}

response = requests.post(url, data=data_json, headers=headers)

df = pd.DataFrame( response.json( )[ 'rows' ] )
```

```
import pyodbc
import pandas as pd

MY_DSN = "DRIVER=/opt/mapr/drillodbc/lib/universal/
libmaprdrillodbc.dylib;Host=localhost;Port=31010;Connect
ionType=Direct;Catalog=Drill;Schema=mfs.views;Authentica
tionType=No Authentication"

conn = pyodbc.connect(MY_DSN, autocommit=True)

employee_query = "SELECT * FROM bhfiles.`urlData2.json`"

data = pd.read_sql( employee_query, conn )
```

spark



cassandra



MESOS

APACHE
HBASE

Spark is Modular

Spark natively supports
Python, Java and Scala.

Spark handles batch,
interactive, and real time
data.

Spark has a DataFrame/
SQL layer built in.

Spark also has a robust
Machine Learning library

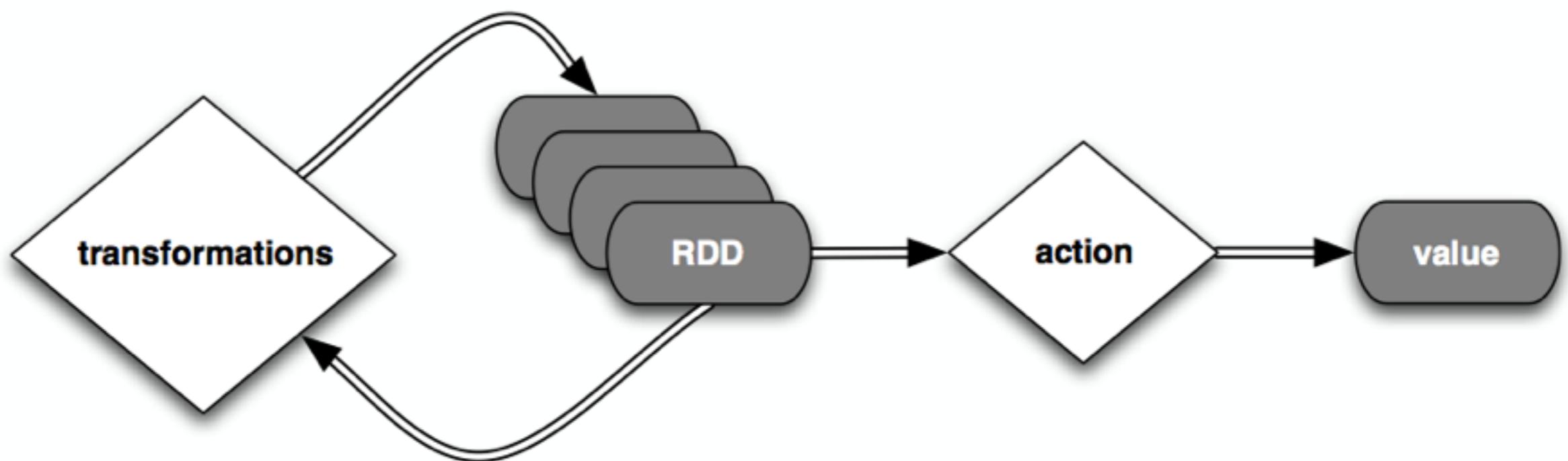
Spark is fast

Spark programming is at a
higher abstraction level
than Hadoop/MapReduce.

**Introducing the
Resilient
Distributed
Dataset: (RDD)**

Spark has two classes of operations: transformations, and actions.

Spark has two classes of operations: transformations, and actions.



```
from operator import add

f = sc.textFile("README.md")
wc = f.flatMap( lambda x: x.split(' '))
    .map( lambda x: (x, 1 ))
    .reduceByKey( add)

wc.saveAsTextFile( "wc_out.txt" )
```

```
from pyspark.sql import SQLContext
from pyspark import SparkContext
sc = SparkContext()

sqlCtx = SQLContext(sc)

#Load a text file and convert each line to a dictionary
lines = sc.textFile("examples/src/main/resources/people.txt")

parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: {"name": p[0], "age": int(p[1])})

peopleTable = sqlCtx.inferSchema(people)
peopleTable.registerAsTable("people")

# SQL can be run over SchemaRDDs that have been registered as a table
teenagers = sqlCtx.sql("SELECT name FROM people WHERE age >= 13 AND age
<= 19")

teenNames = teenagers.map(lambda p: "Name: " + p.name)
teenNames.collect()
```

```
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
from pyspark.mllib.util import MLUtils

# Load and parse the data file into an RDD of LabeledPoint.
data = MLUtils.loadLibSVMFile(sc, 'data/mllib/sample_libsvm_data.txt')

# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = data.randomSplit([0.7, 0.3])

# Train a DecisionTree model.
model = DecisionTree.trainClassifier(trainingData,
    numClasses=2,
    categoricalFeaturesInfo={},
    impurity='entropy',
    maxDepth=5, maxBins=32)

# Evaluate model on test instances and compute test error
predictions = model.predict(testData.map(lambda x: x.features))
labelsAndPredictions = testData.map(lambda lp: lp.label).zip(predictions)
testErr = labelsAndPredictions.filter(lambda (v, p): v != p).count() /
float(testData.count())

print('Test Error = ' + str(testErr))
print('Learned classification tree model:')
print(model.toDebugString())
```