Rapid Data Exploration With Pandas

Course Agenda

- Intro: What is Pandas?
  - Overview of Data Exploration and the Pandas libraries place in the analysts toolbox
- Understanding Pandas Data Structures
  - It was the best of data, it was the worst of data. A tale of two structures.
- Loading and Manipulating Data
  - How to load different data sources into Pandas objects.
- Fundamental Statistical Analysis
  - Low math high impact.
- Life After Pandas

Expectations

- Please participate and **ask questions.**
- Please follow along and **TRY OUT** the examples yourself during the class
- All the answers are in the slide decks or GitHub repository, but please try to complete the exercises **without looking at the answers**.
- Join the conversation in slack!
- Have fun!

Introduction

Our Lawyers Make Us Say This

Brian Genz

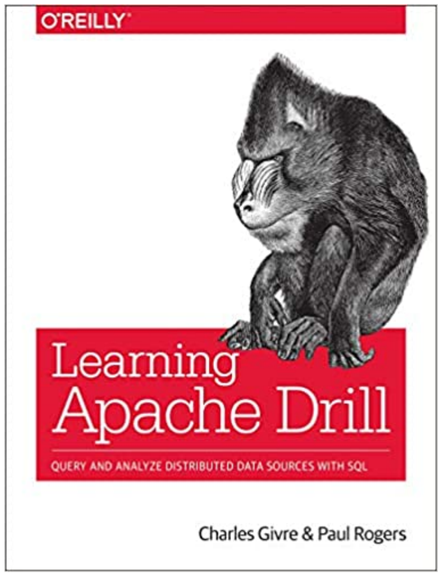
Brian leads the Threat & Vulnerability Management team at Splunk and serves as a Senior Instructor with GTK Cyber.

Prior to leading the red team at Splunk, he led threat hunting and security orchestration, automation and response (SOAR) efforts with Splunk Phantom at a Fortune 100 financial company. He has experience in the defense intelligence, manufacturing, and financial sectors and has worked in both offensive and defensive security. Degrees and certifications include MBA, M.S. in Information Technology Management, CISSP, GREM, GNFA, GCFA, and GCIH.

Brian has taught with GTK Cyber at Black Hat USA and in Europe. He has presented at multiple industry conferences, including the DEFCON AI Village, the Conference on Applied Machine Learning for Information Security (CAMLIS), Derby Con, Circle City Con, Cyphercon, the ISSA International Conference, ISACA Milwaukee, Infragard, and others. He has served as adjunct faculty at Wayne State University and Milwaukee Area Technical College. He is also featured in the book, "Tribe of Hackers: Red Team Edition."

Charles Givre, CISSP

- Lead Data Scientist at JP Morgan Chase
- PMC Chair for Apache Drill
- Senior Lead Data Scientist @ Booz Allen
- 5 Years @ CIA
- Undergraduate in Comp.Sci & Music



JPMorgan Chase & Co.

Deutsche Bank

Booz | Allen | Hamilton

100 YEARS

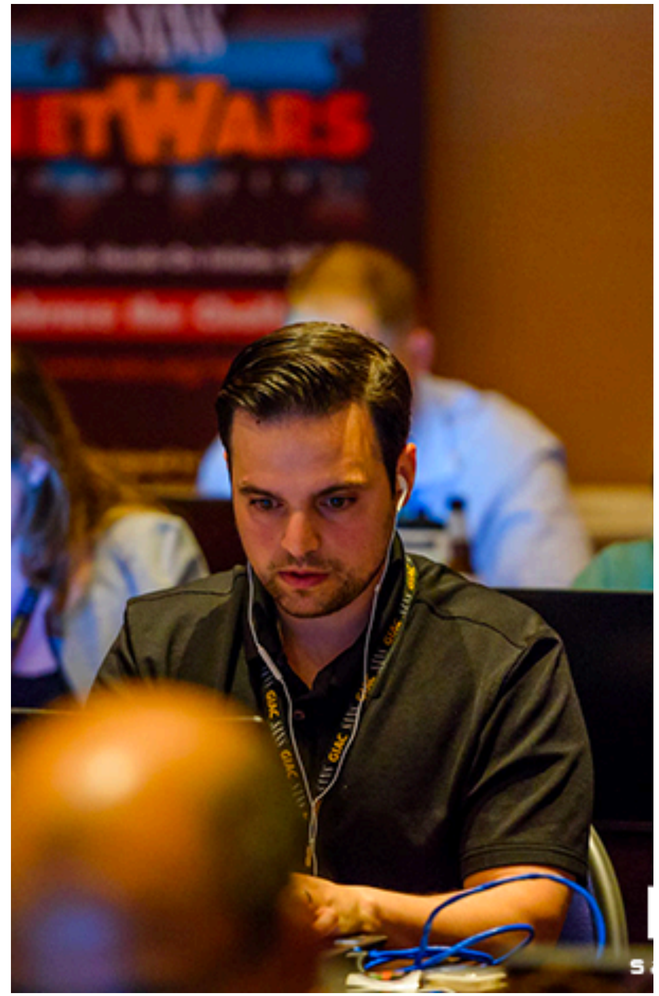
ARIZONA

Brandeis

TRUTH · EVEN UNTO ITS INNERMOST PARTS · אמת

Austin Taylor, CISSP

- Director, Cybersecurity R&D @ IronNet Cybersecurity
- Cyber Warfare Operator @ USAF (MDANG)
- Projects: Flare, Bluewall, VulnWhisperer
- Publications: Build A World Class Monitoring
- System for Enterprise, Small Office, or Home
- SANS Instructor - Continuous Monitoring
- GTK Cyber - Instructor



U.S. AIR FORCE

Flare
An analytical framework for network data

VulnWhisperer

SANS

@HuntOperator

www.austintaylor.io

Curtis Lambert, CISSP

- CTO at DataDistillr
- U.S. Army Intelligence Analyst and Linguist
- Security Researcher
- Former BAH Data Scientist supporting DoD
- GTK Cyber - Instructor
- SANS Certification Collector
- Husband/Father
- Terrible guitar player but a decent mechanic

Who are you?

- Your name (or what you want us to call you)
- Your job role
- What you hope to get out of this class
- Your level of experience with coding

What is Data Exploration?

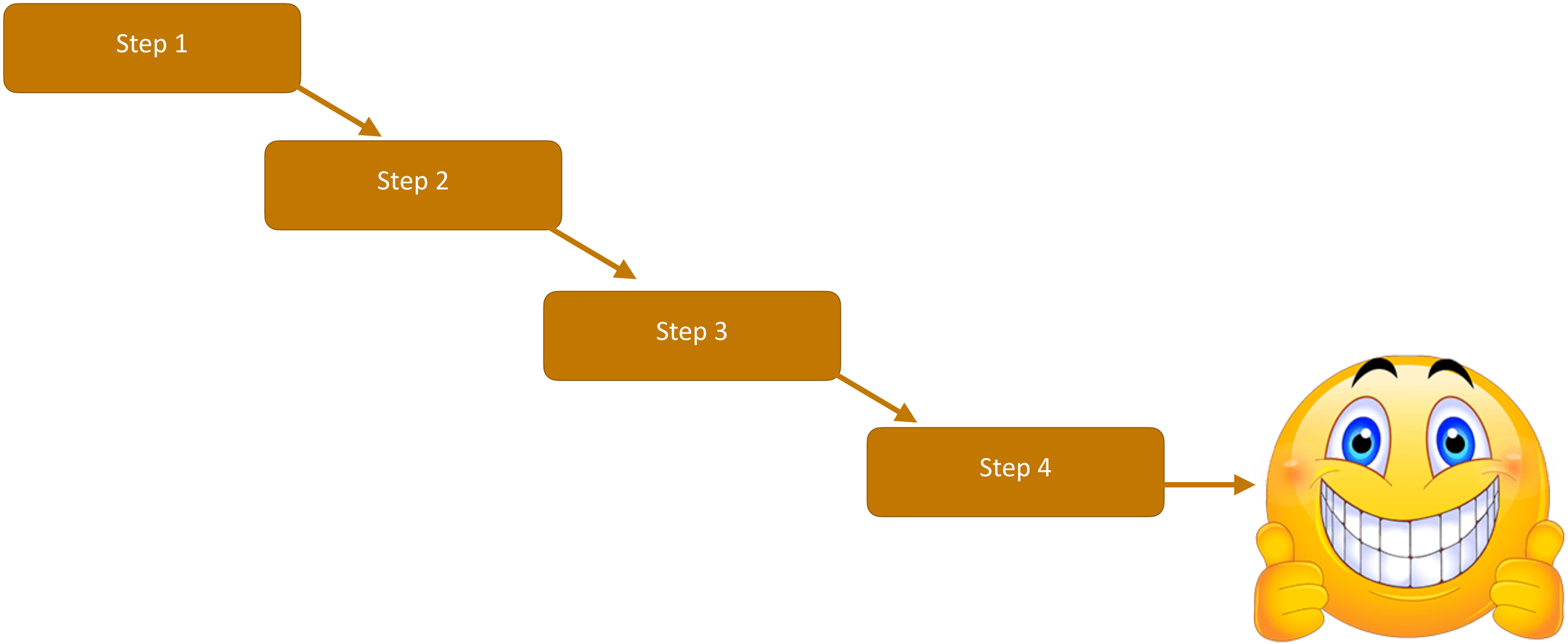
Data Exploration: Is the art of understanding data characteristics and relationships. This is accomplished through analyzing the size, shape, characteristics and correctness of data.

Analyst  Developer

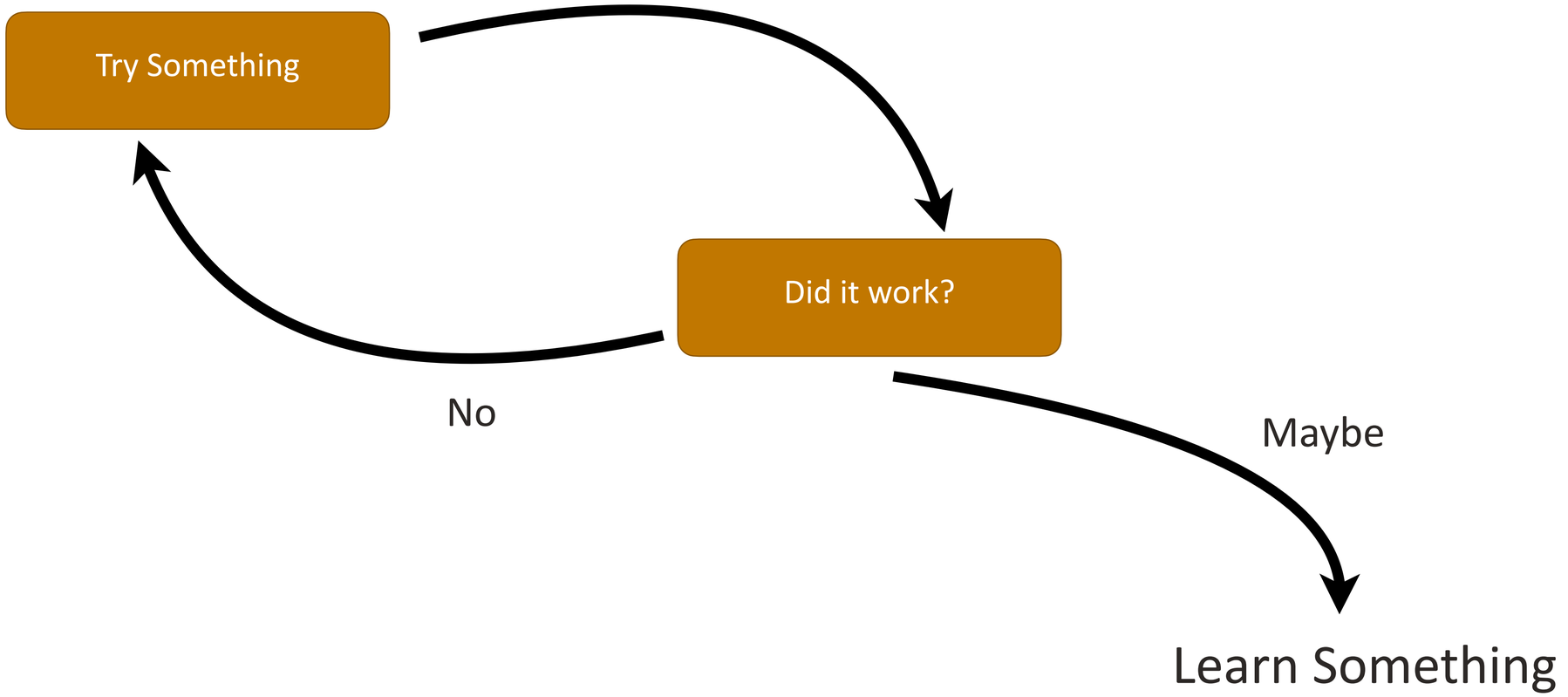
Vs.

Analyst + Developer

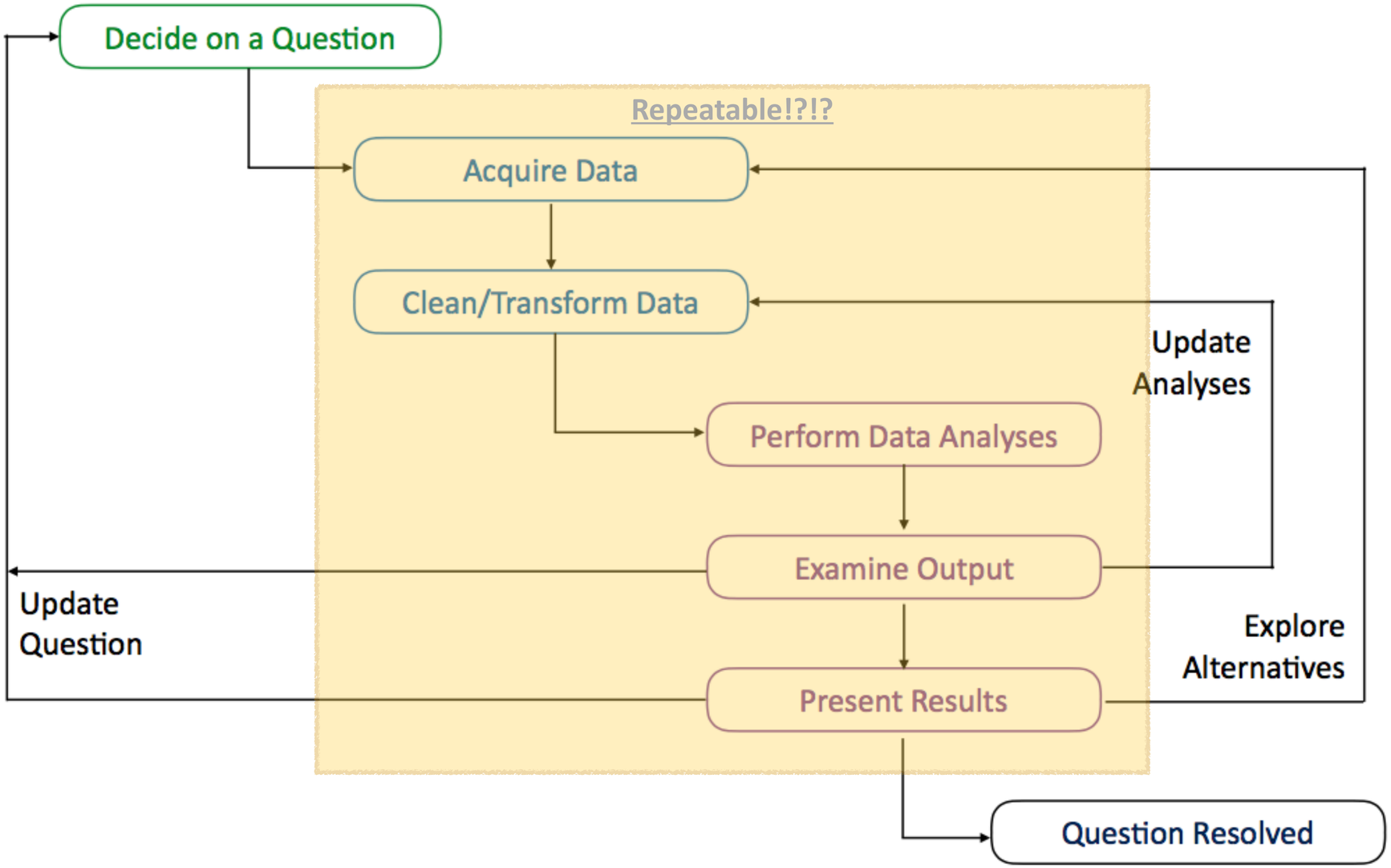
What Data Exploration is Not



What Data Exploration Is



Research Process



Why are you doing this analysis?

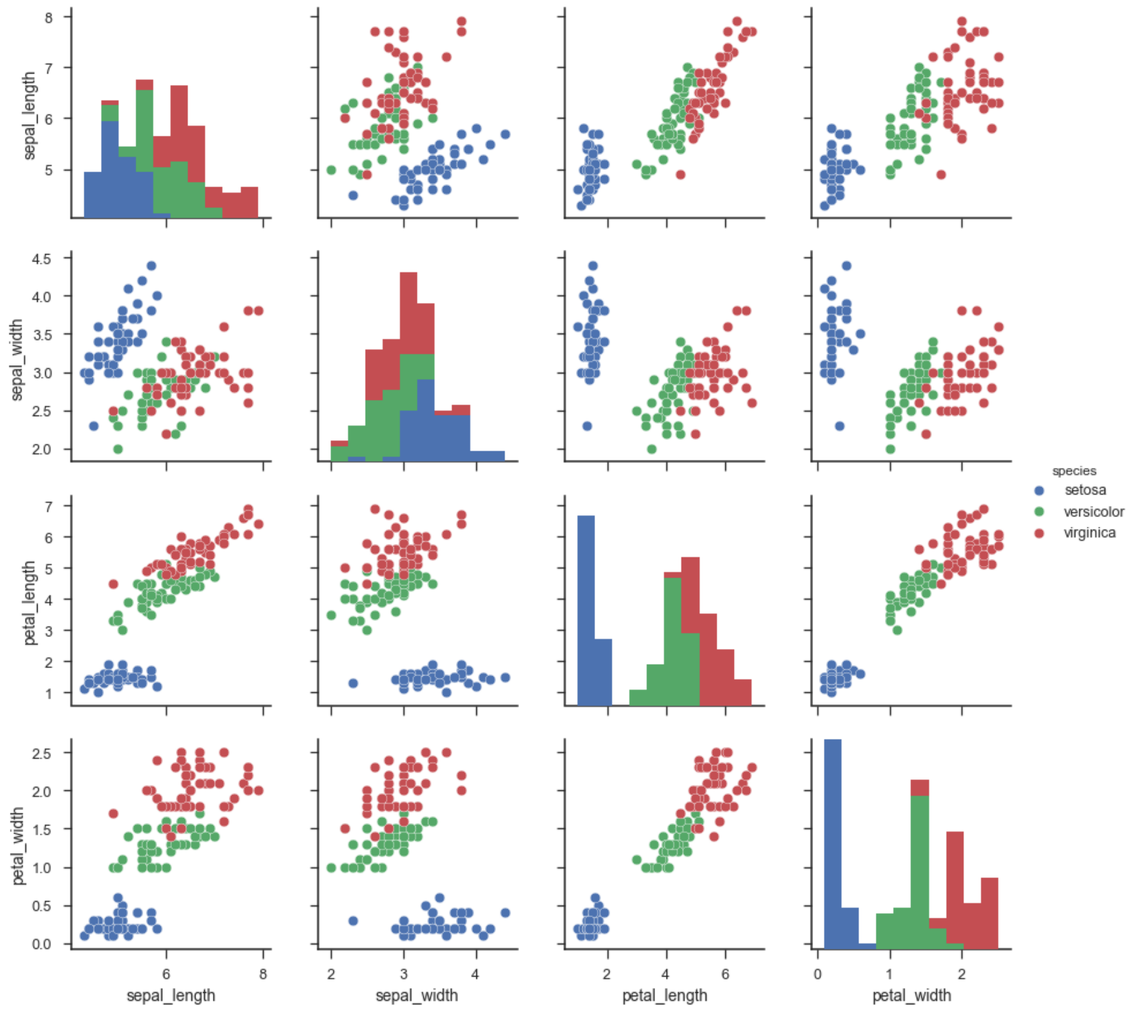
How do you define success?

What are you measuring?

Choose data sources

- What do you need?
- What do you have?
- What can you get?
- What gaps does that leave?
- Is the data reliable/clean/consistent?

Gather and Explore Your Data



- Policies/Legal constraints
- Biases in Data
- Latency/Availability/Data size
- Is the data good enough?
- What are the rules governing its use?
- Do I have enough?
- Do problems or biases exist in the data that could cause problems?

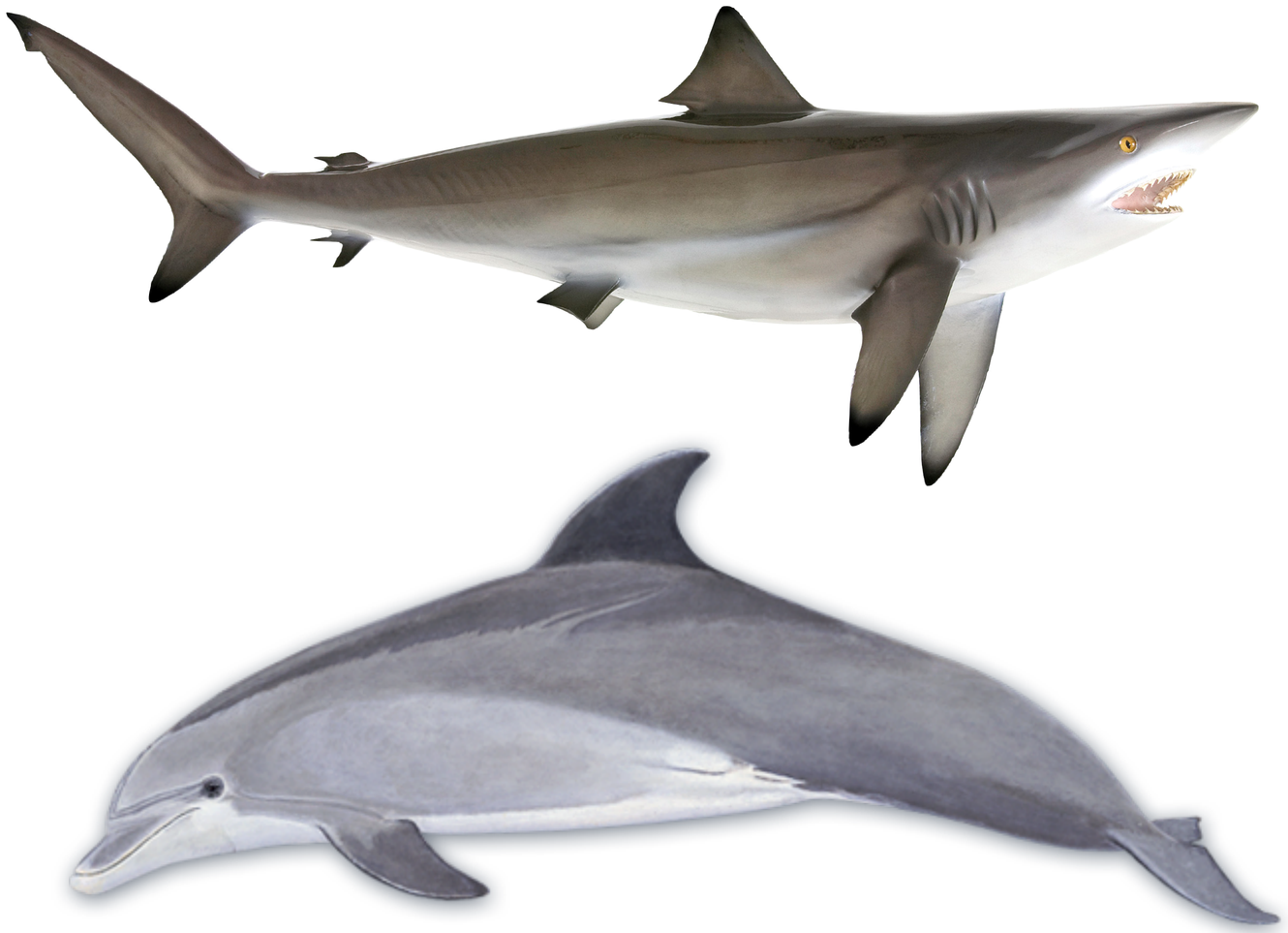
Feature Engineering

- Define what you are trying to measure. These will become the **observations** or rows of your final dataset
- Define how you will represent your data. This will become the **features** or columns of your final dataset.

Feature Engineering

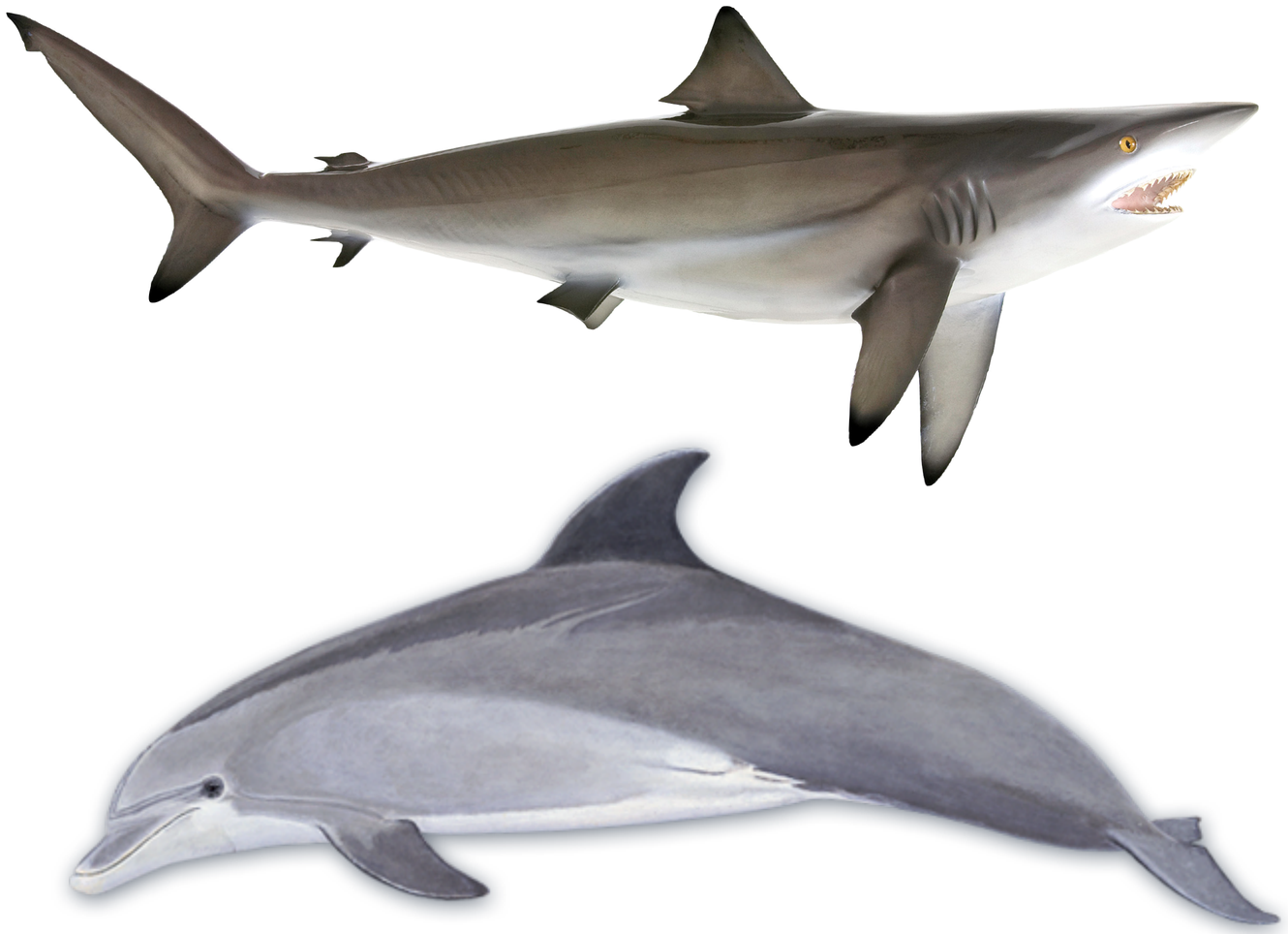
Feature	Value
Color	Gray
Fins	7
Predator	TRUE

Feature Engineering



Feature	Value
Color	Gray
Fins	7
Predator	TRUE

Feature Engineering



Feature	Value
Color	Gray
Fins	7
Predator	TRUE
Mammal	**TRUE**

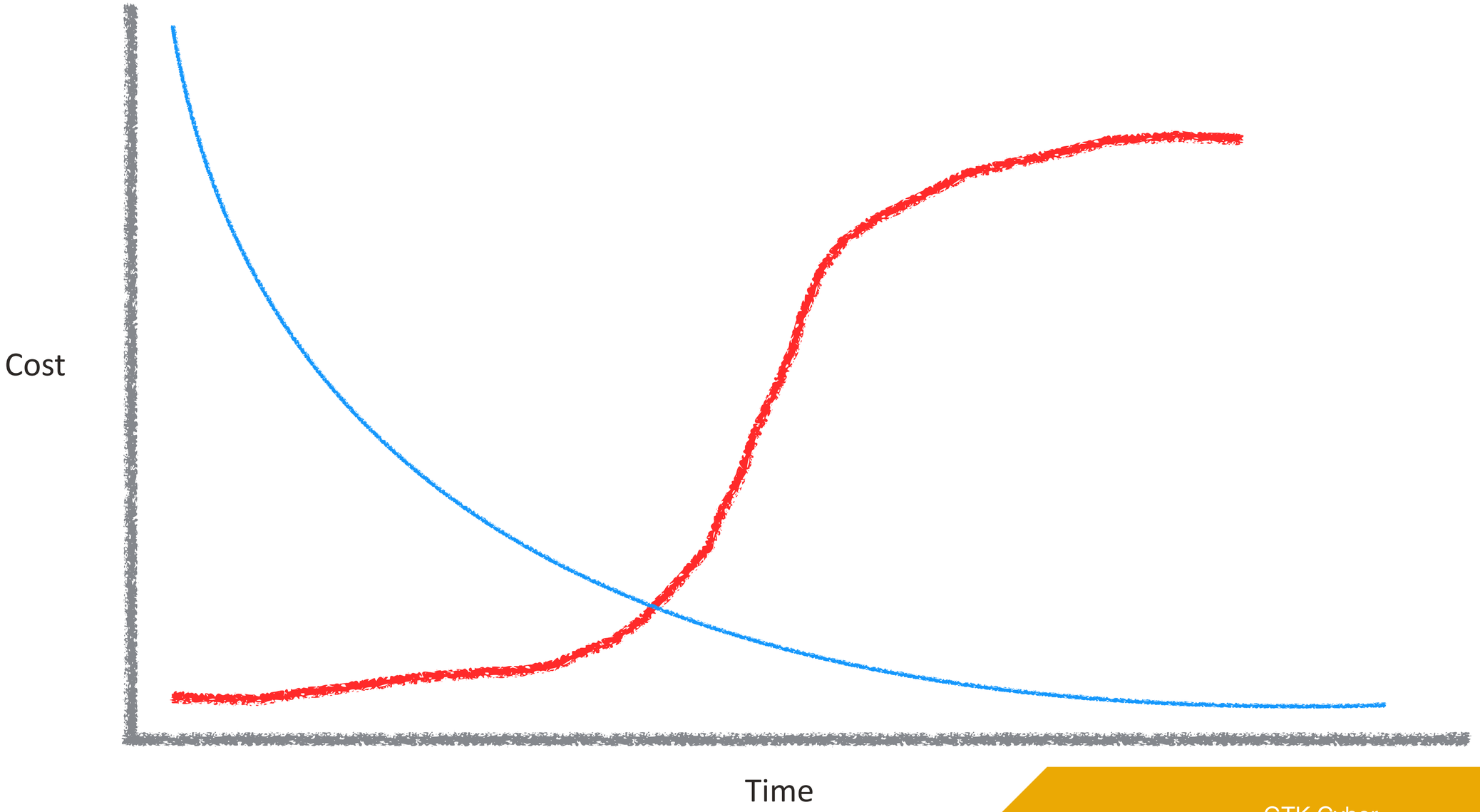
“The term "data scientist" will subside and may well sound dated five years from now. **The skills will become more commonplace and commoditized. When that happens, the real boom will begin**, because the technology will become widely adopted and thus more useful. … **Instead, we need self-service tools 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/

Time to Insight

Time == $$

Value Vs. Cost of Insights Over Time

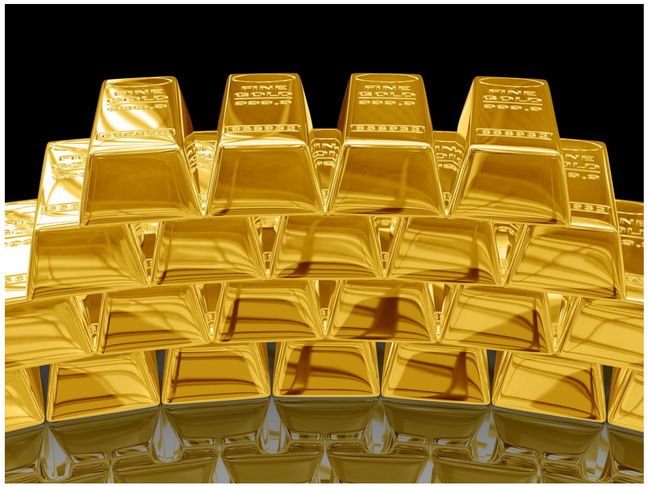


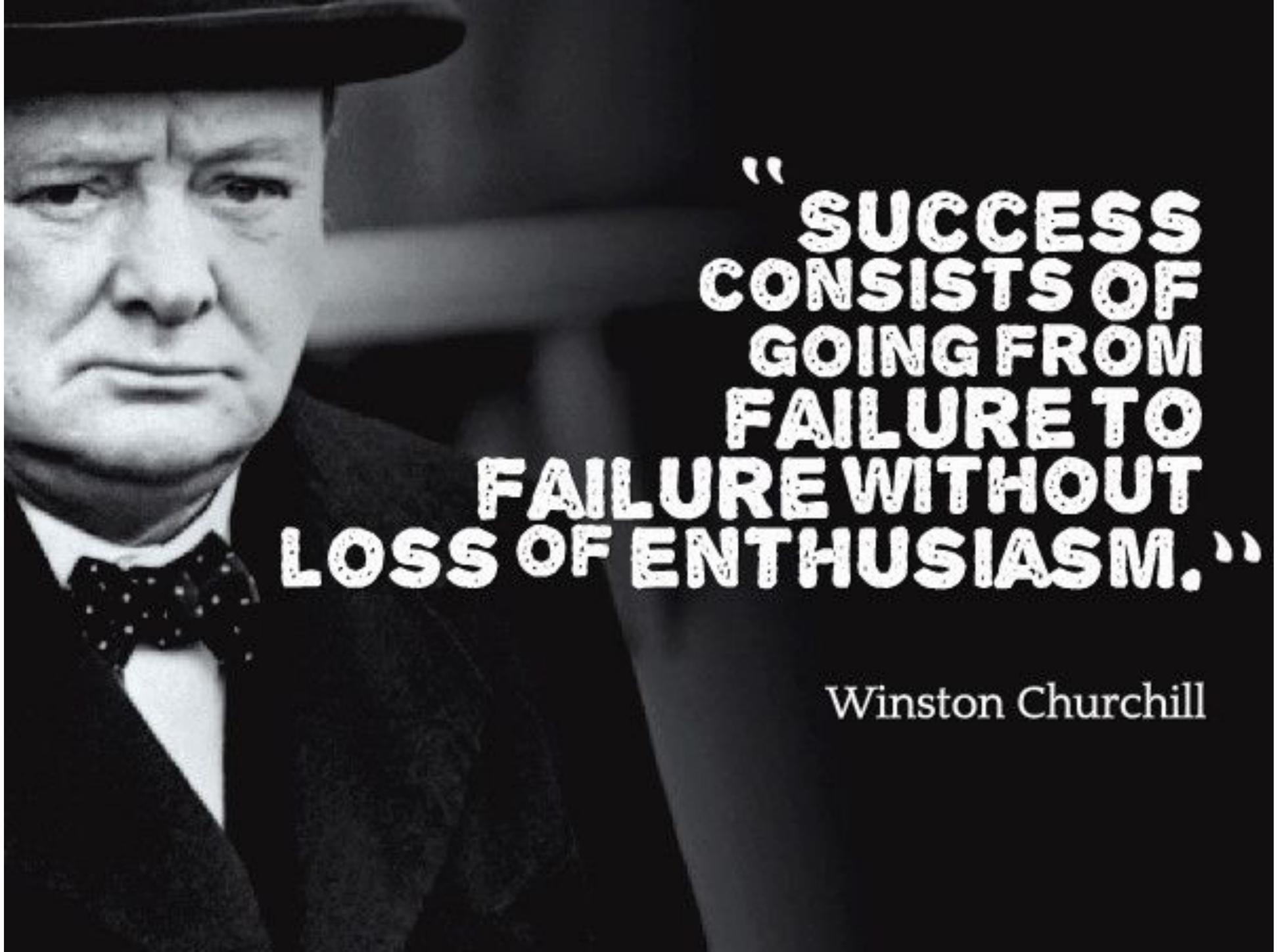
Data exploration and analysis accounts for 50-90% of time to production

Data Janitors

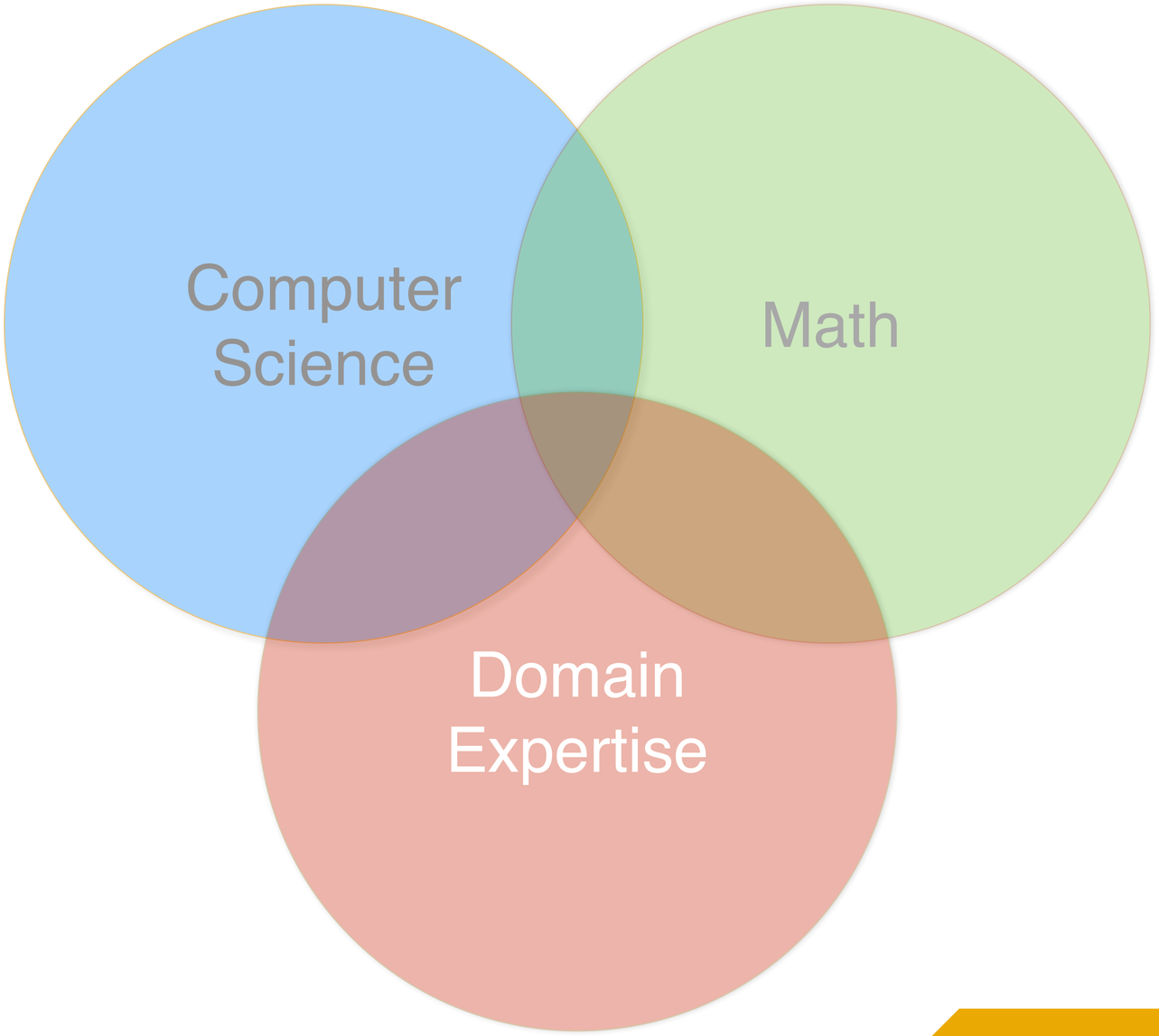
Your:
Time
Money
Job?

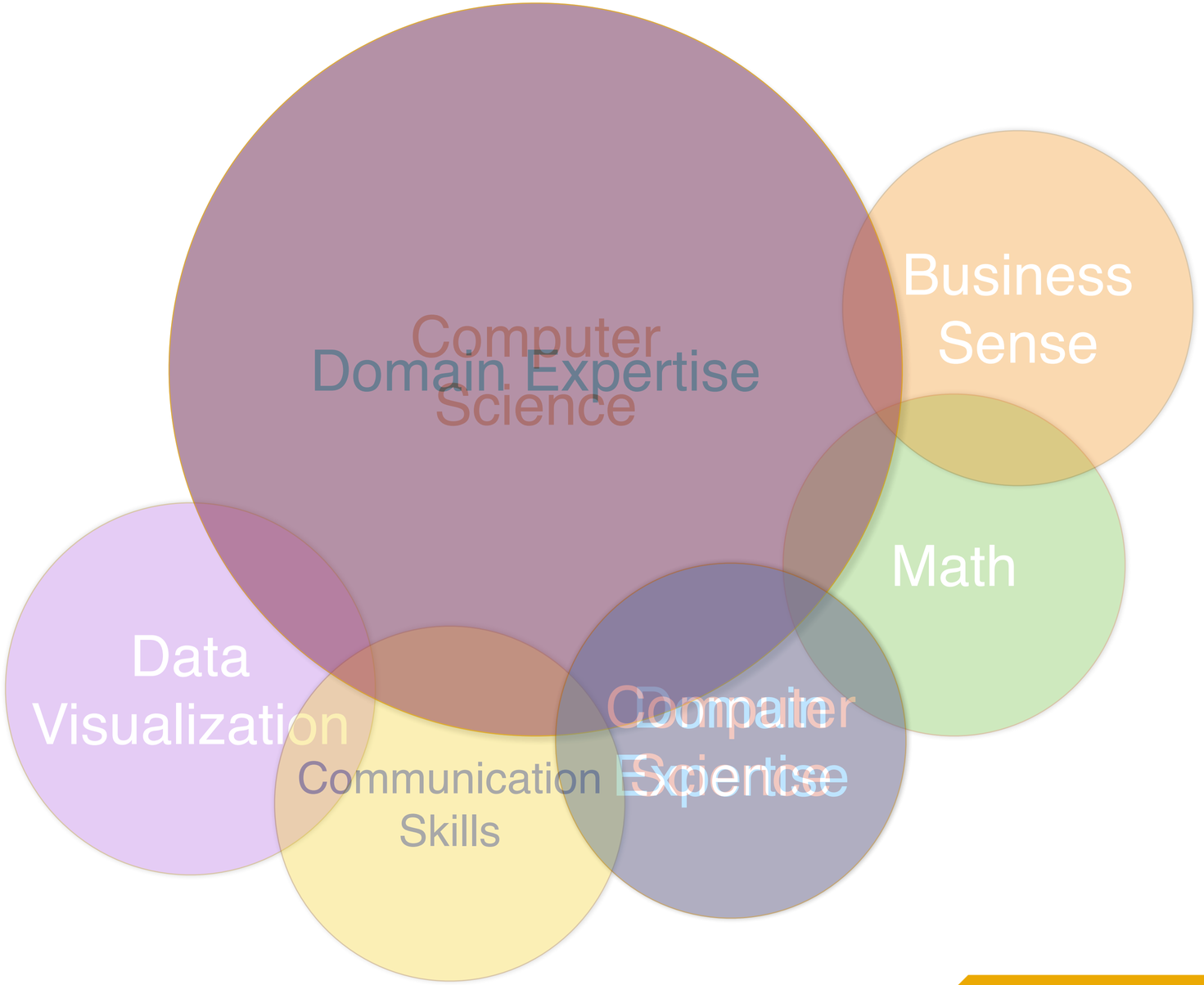
Align Goals to Corporate Strategy?

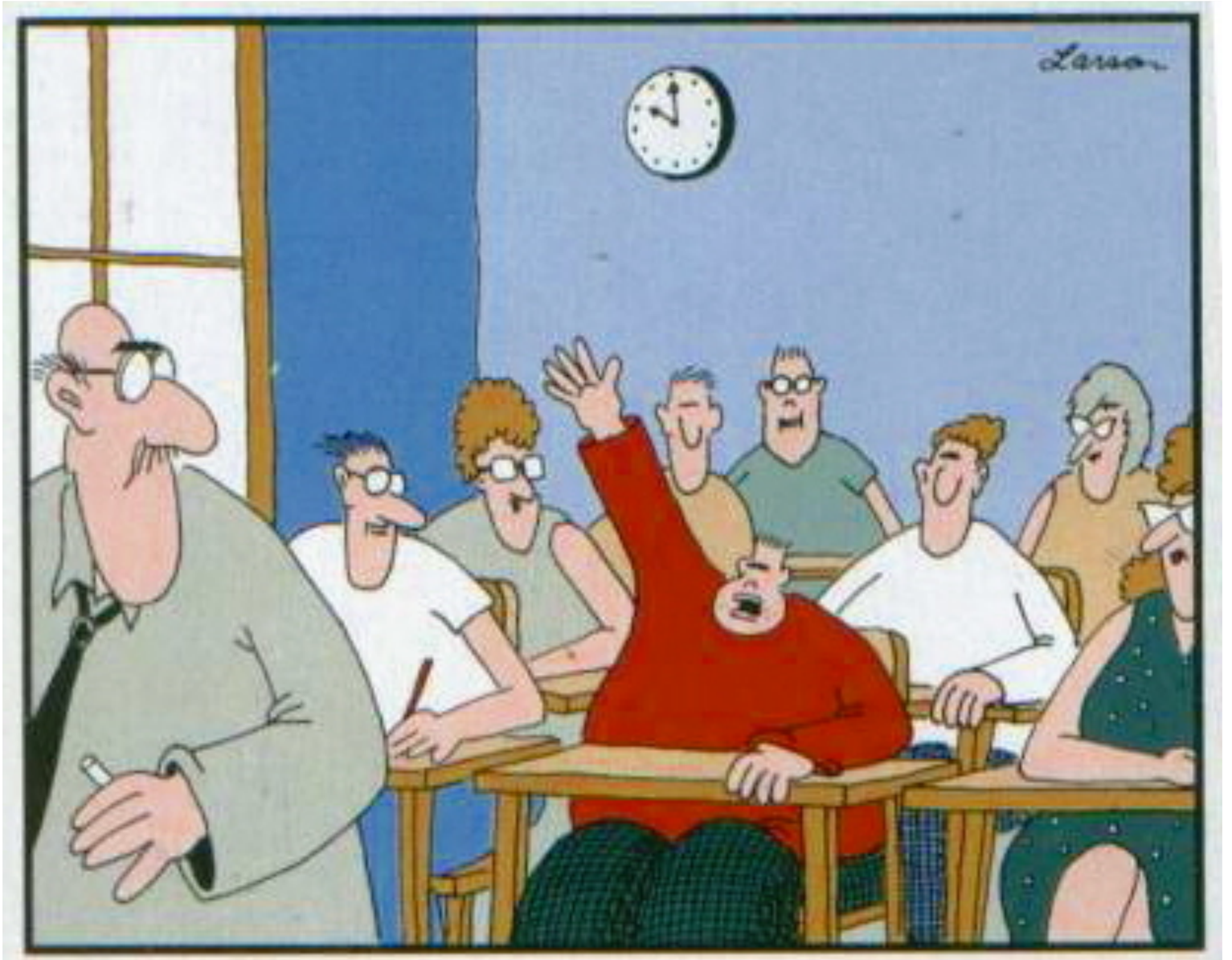



"SUCCESS CONSISTS OF GOING FROM FAILURE TO FAILURE WITHOUT LOSS OF ENTHUSIASM."
Winston Churchill

What Skills Do You Need?


Computer
Science
Math
Domain
Expertise


Computer
Domain Expertise
Science
Business
Sense
Math
Data
Visualization
Communication
Skills
Computer
Expertise



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

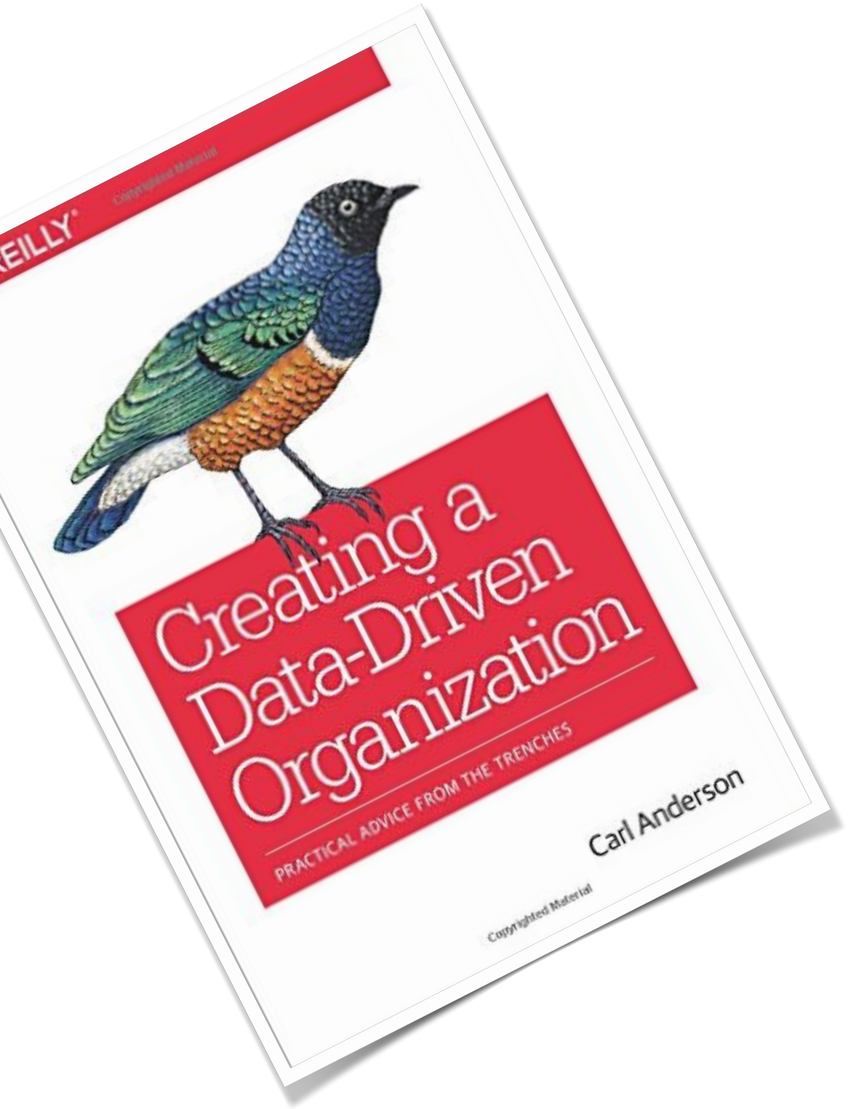
Build the right team for Data Initiatives

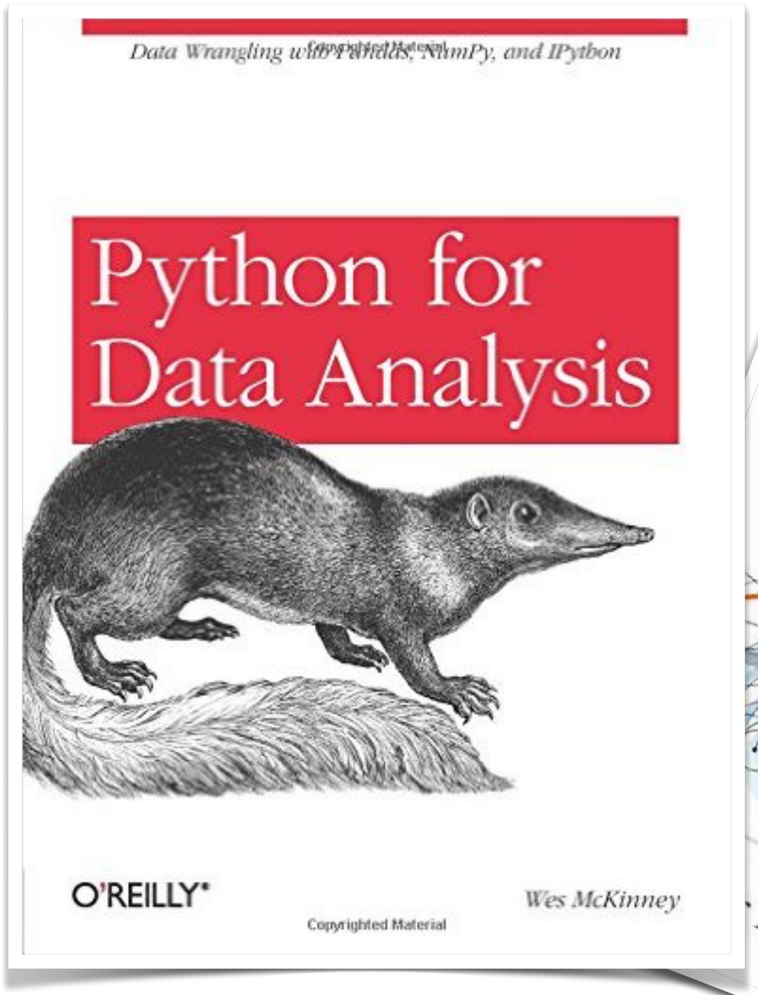

I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
E-mail: SCOTTADAMS@AOL.COM
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1996 United Feature Syndicate, Inc.(NYC)
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.

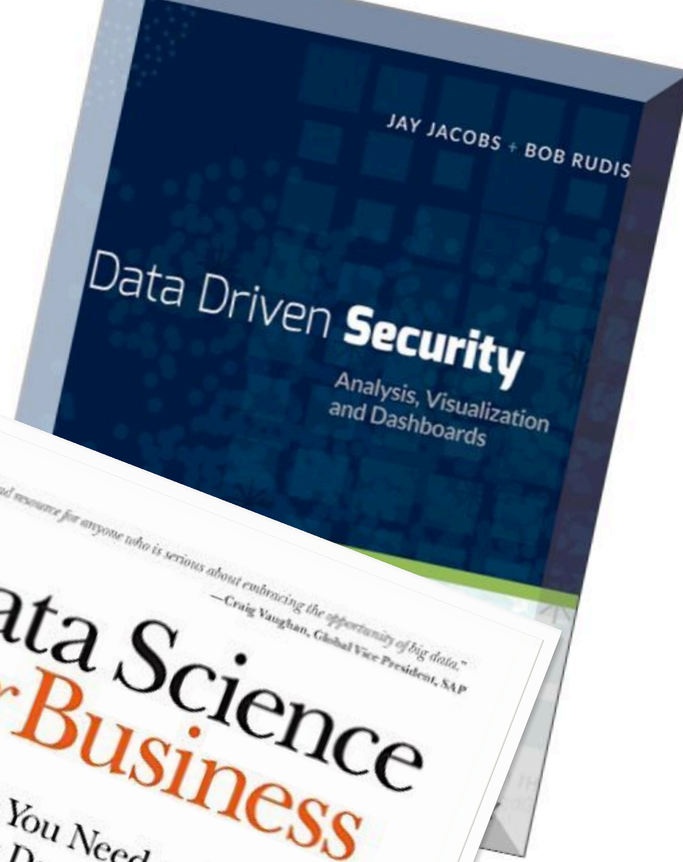
By The End of the Class, You Will Be Able To:

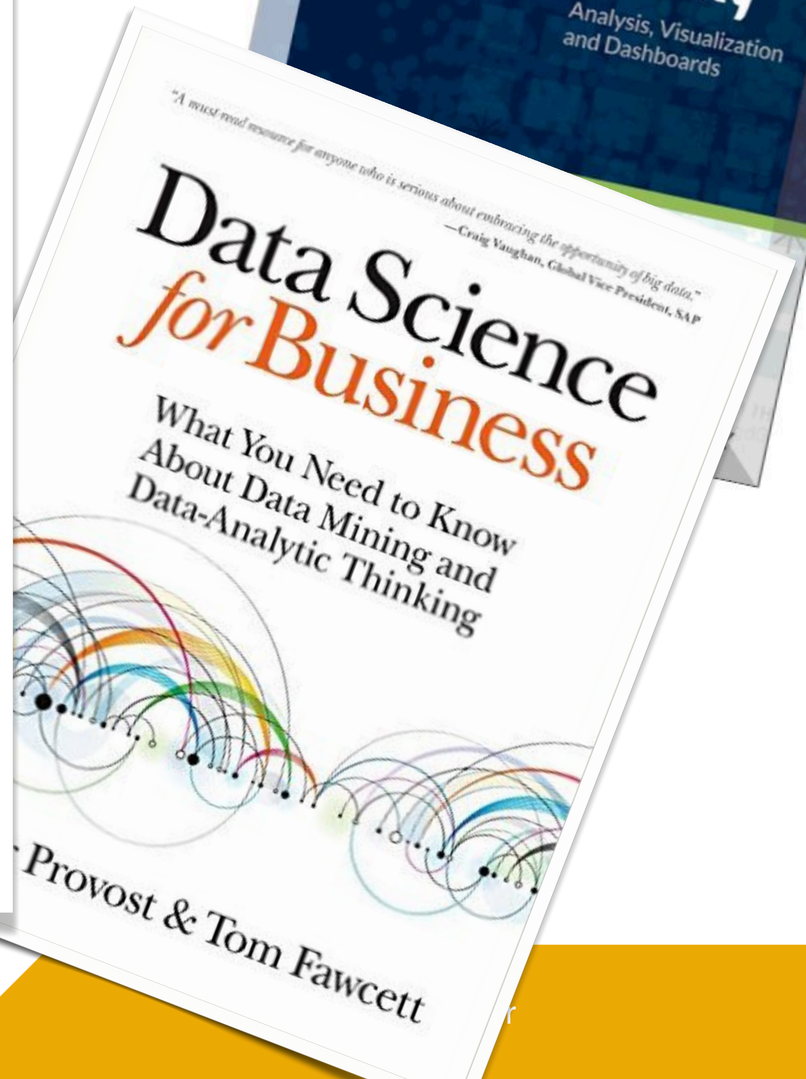
Quickly and effectively prepare data and perform exploratory analysis to derive insights

Recommended Reading









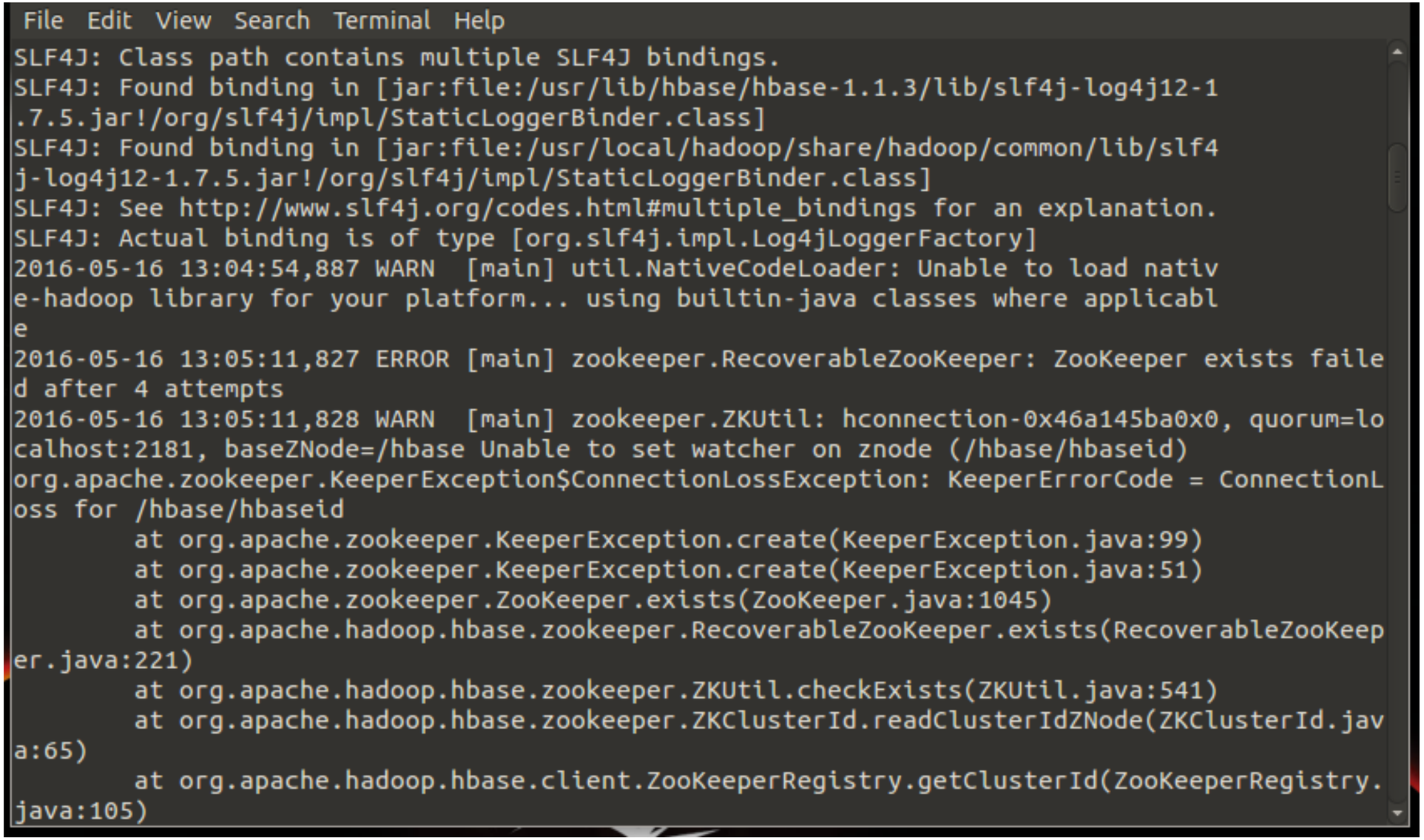
The Virtual Machine: Griffon

Built on Ubuntu MATE

Ease to use

Nice interface

Familiar navigation styles

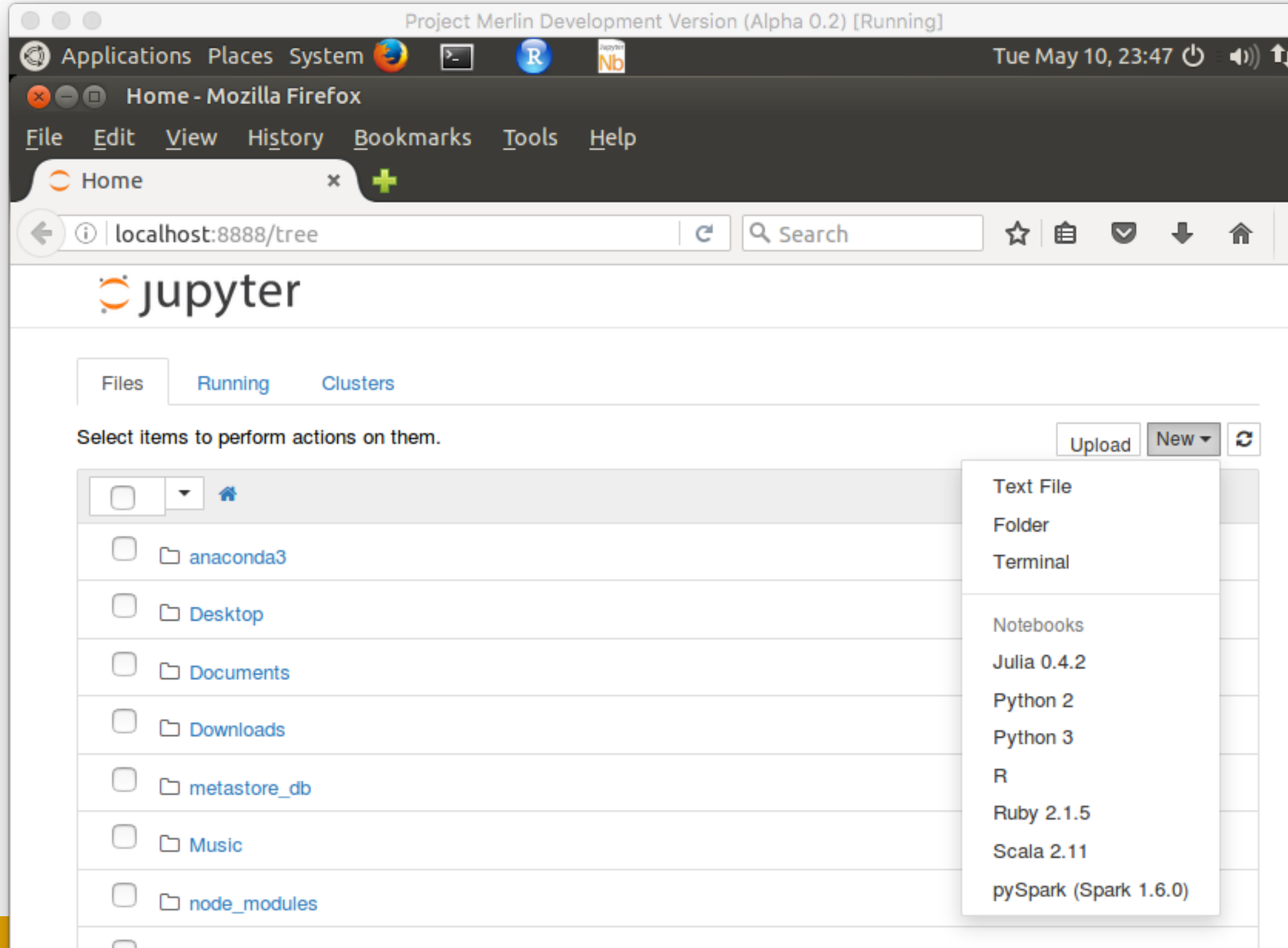

File Edit View Search Terminal Help
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hbase/hbase-1.1.3/lib/slf4j-log4j12-1
.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2016-05-16 13:04:54,887 WARN [main] util.NativeCodeLoader: Unable to load nativ
e-hadoop library for your platform... using builtin-java classes where applicabl
e
2016-05-16 13:05:11,827 ERROR [main] zookeeper.RecoverableZooKeeper: ZooKeeper exists faile
d after 4 attempts
2016-05-16 13:05:11,828 WARN [main] zookeeper.ZKUtil: hconnection-0x46a145ba0x0, quorum=lo
calhost:2181, baseZNode=/hbase Unable to set watcher on znode (/hbase/hbaseid)
org.apache.zookeeper.KeeperException$ConnectionLossException: KeeperErrorCode = ConnectionL
oss for /hbase/hbaseid
at org.apache.zookeeper.KeeperException.create(KeeperException.java:99)
at org.apache.zookeeper.KeeperException.create(KeeperException.java:51)
at org.apache.zookeeper.ZooKeeper.exists(ZooKeeper.java:1045)
at org.apache.hadoop.hbase.zookeeper.RecoverableZooKeeper.exists(RecoverableZooKeep
er.java:221)
at org.apache.hadoop.hbase.zookeeper.ZKUtil.checkExists(ZKUtil.java:541)
at org.apache.hadoop.hbase.zookeeper.ZKClusterId.readClusterIdZNode(ZKClusterId.jav
a:65)
at org.apache.hadoop.hbase.client.ZooKeeperRegistry.getClusterId(ZooKeeperRegistry.
java:105)

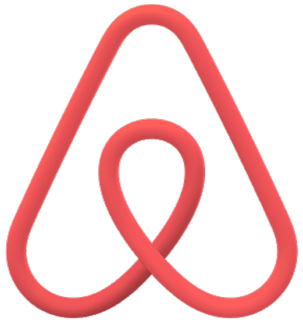
Programming Languages

- Languages
- Libraries
- Editors and Notebooks
- Databases + Administrative tools
- Big Data Tools
- Machine Learning Libraries
- Data Visualization


Griffon RC3 [Running]
Applications
Places
System
Accessories
Big Data
Databases
Data Visualization
Education
Graphics
Internet
Office
Other
Programming
Sound & Video
System Tools
Universal Access
passwords.txt
Shells
Atom
DBeaver CE
H2O.ai
IntelliJ IDEA Community Edition
Jupyter Notebook
PyCharm Community Edition
Rodeo
RStudio
Spyder
Julia
NodeJS
PHP Shell
Pig Shell
Python (v2.7)
Python (v3.5)
R
Ruby
Scala

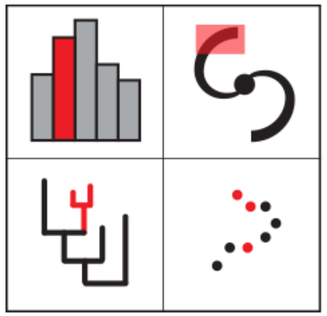

Jupyter
Nb


Project Merlin Development Version (Alpha 0.2) [Running]
Applications Places System
Tue May 10, 23:47
Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Home
localhost:8888/tree
Search
jupyter
Files Running Clusters
Select items to perform actions on them.
Upload
New
Text File
Folder
Terminal
Notebooks
Julia 0.4.2
Python 2
Python 3
R
Ruby 2.1.5
Scala 2.11
pySpark (Spark 1.6.0)
anaconda3
Desktop
Documents
Downloads
metastore_db
Music
node_modules




plotly

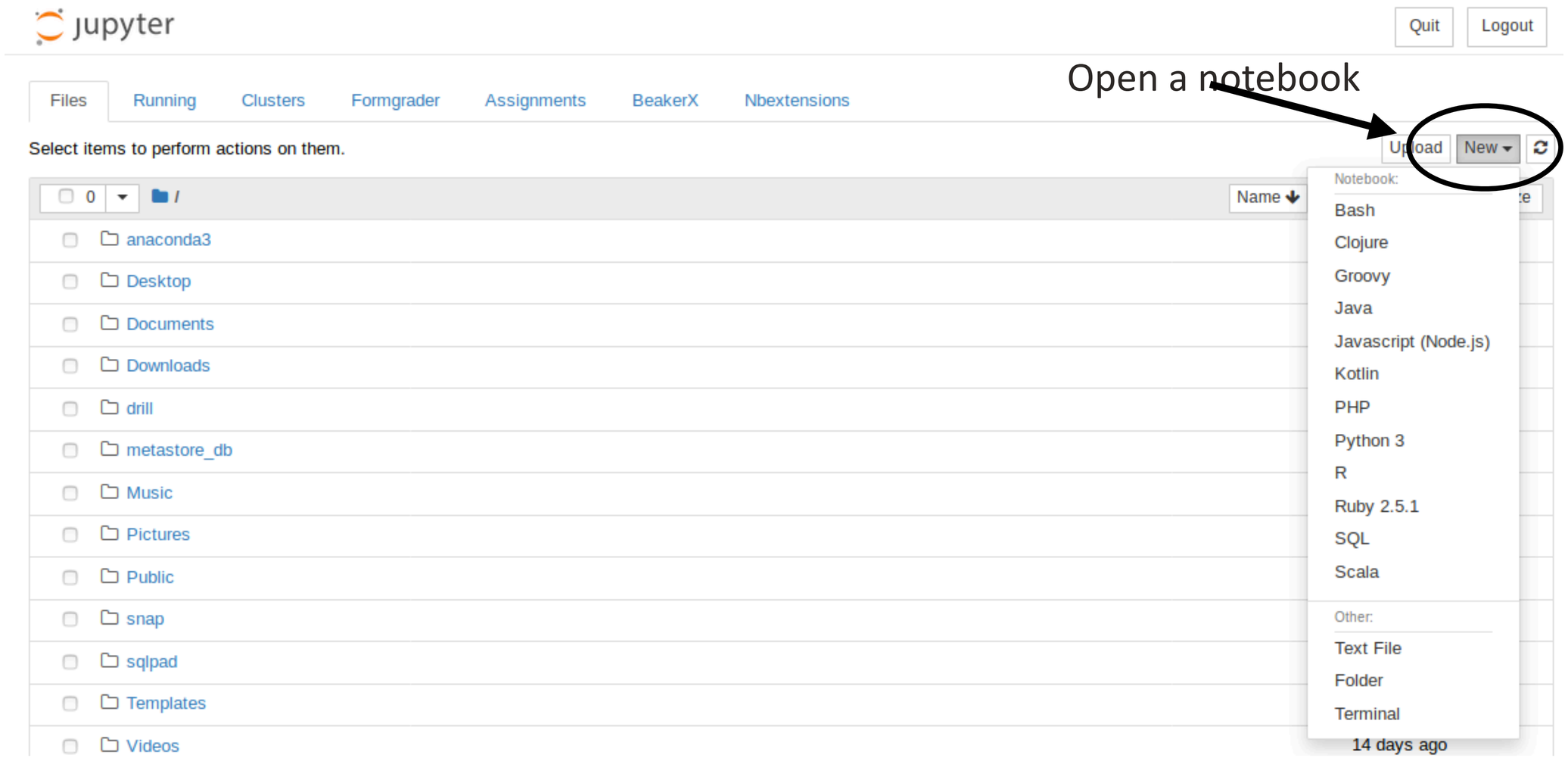

orange
DATA MINING
FRUITFUL&FUN



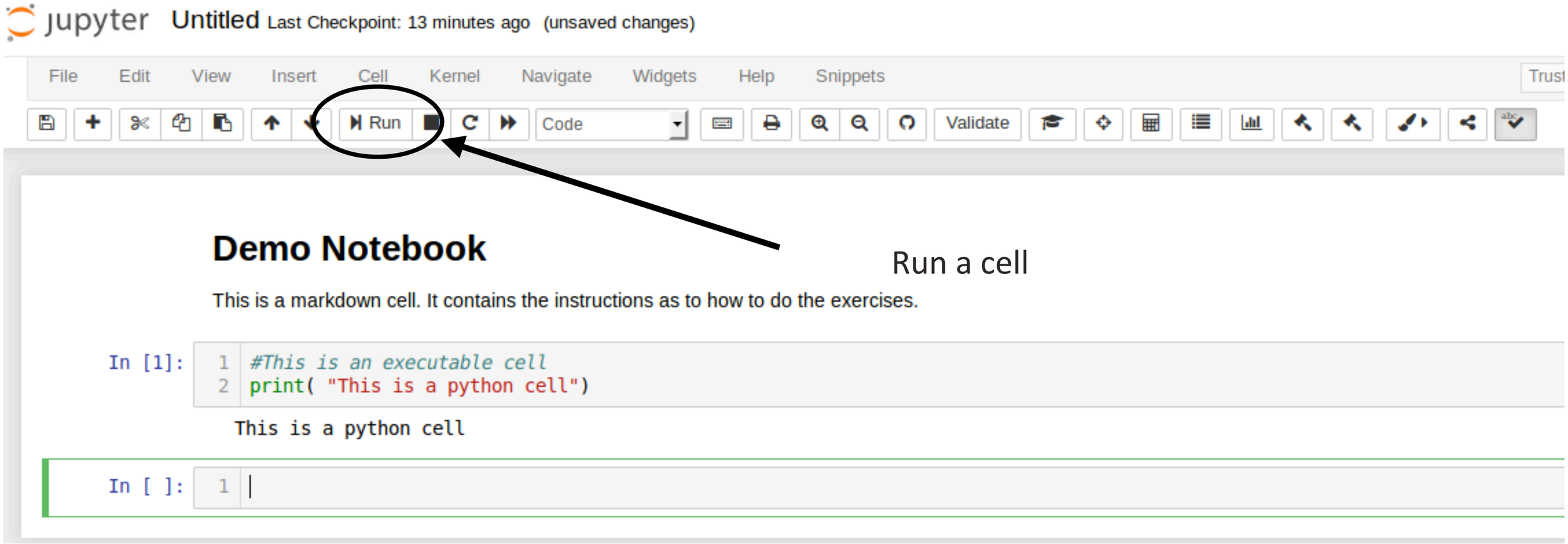
Questions?

Using Jupyter Notebook

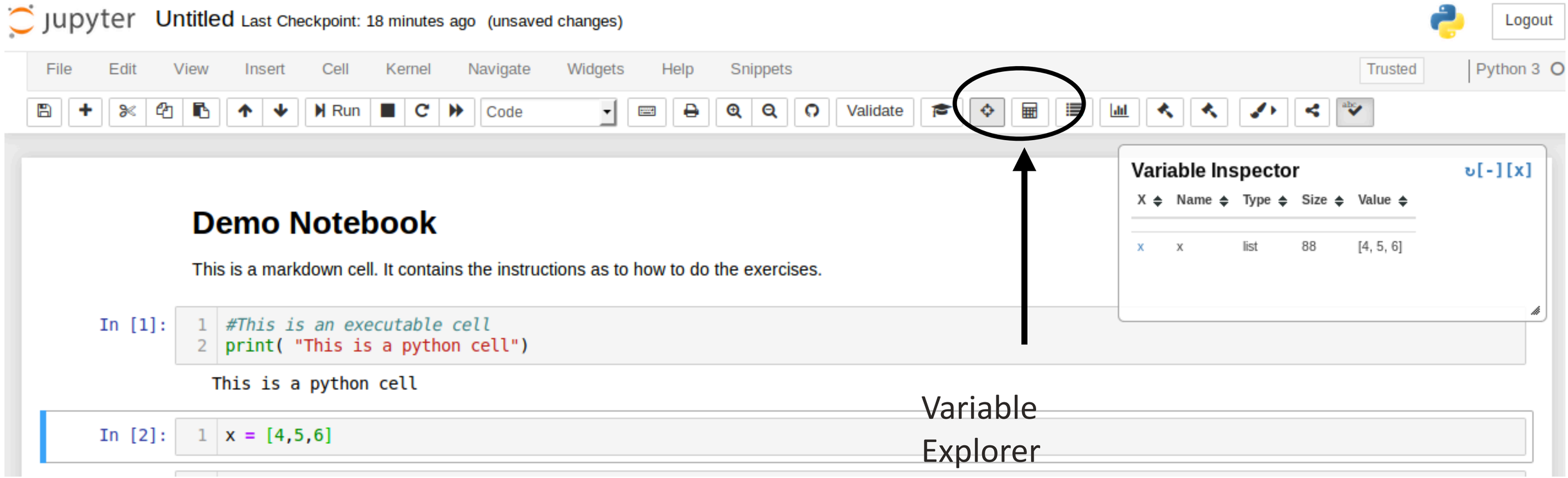
Using Jupyter Notebook



Using Jupyter Notebook



Using Jupyter Notebook



Questions?

What is Pandas?

Pandas is a software library written for the Python programming language for data manipulation and analysis.

https://en.wikipedia.org/wiki/Pandas_(software)

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~~

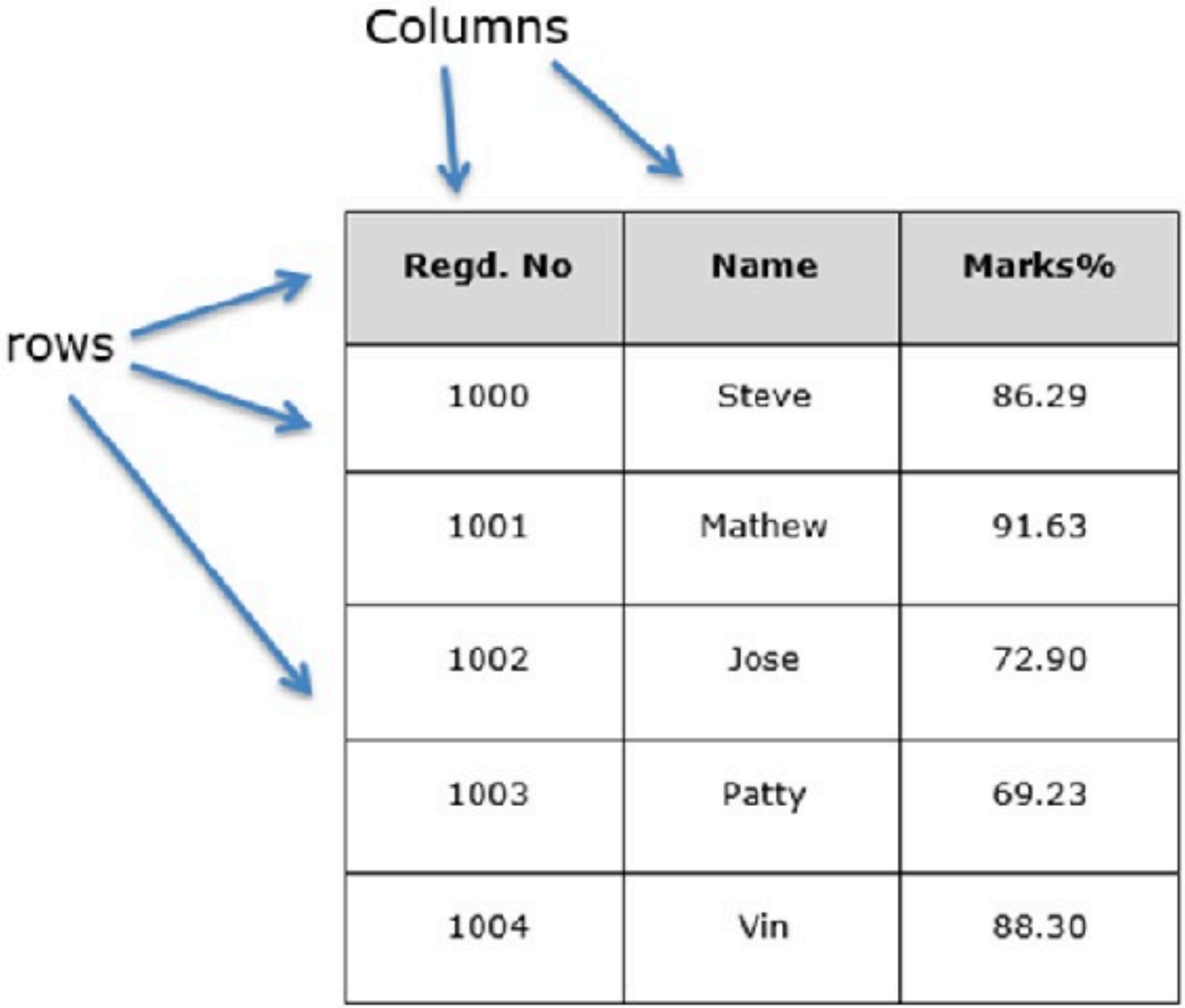


Image Source:

Pandas Place in The Data Science Universe!!!

Machine Learning Ecosystem

- **Data Gathering:**
  - *Pandas*, Drill, BeautifulSoup, PyDBAPI, PyDAL, Boto3
- **Exploration & Feature Extraction:**
  - *Pandas*, NumPy, Featuretools
- **Machine Learning:** (fed by *Pandas* objects)
  - "Regular" ML: Scikit-learn (sklearn), h2o, mllib (PySpark)
  - Deep Learning: Tensorflow, Keras, Theano, Caffe, PyTorch
- **Visualization**: (fed by *Pandas* objects)
  - Matplotlib, Seaborn, Yellowbrick, LIME, ggplot, plot.ly,

Pandas Data Structures

Vectorized Data Structures

What are they?

Vectorized Data Structures let you perform operations on your data all at once

Advantages of Vectorized Data Structures

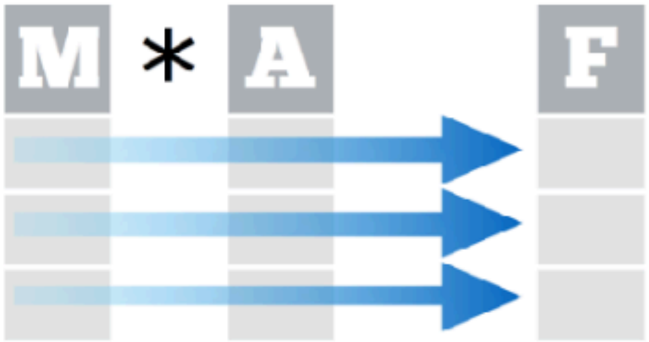
- More readable and simpler code
- Allows the module to optimize for CPU and Memory usage
- Pass serialized objects from one system to another

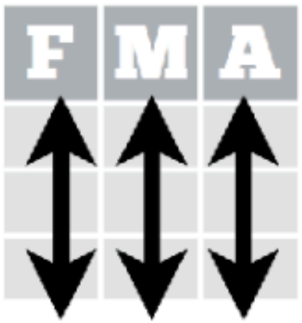
```
for x in range(0, len( data )):
    data[x] = data[x] + 1
```

```
for x in range(0, len( data )):
    data[x] = data[x] + 1
```

```
odds = evens + 1
```

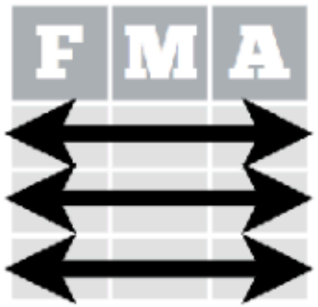
F = M * A





Each **variable** is saved in its own **column**

&



Each **observation** is saved in its own **row**

No loops

Buckaroo Bitguy and Data Exploration in The First Dimension!

The Series Object

The Series Object

Regd. No
1000
1001
1002
1003
1004

A Series is like a *list* or a *dictionary* but 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

You can access items in a Pandas Series by index, similar to a python list. Just

as in a regular list, indexing starts at zero.

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

        print( series1[3] )
```

```
d
```

Accessing a Series: Slicing

```
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.random_integers(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.random_integers(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
```

Filtering Data in a Series

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

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

Date/Time Operations

```
pd.to_datetime(<times>, format='<format string>')
```

https://www.python.org/dev/peps/pep-0321/

Selected Date/Time Operations

Function	Explanation
`series.dt.year`	Returns the year of the date time
`series.dt.month`	Returns the month of the date time
`series.dt.day`	Returns the days of the date time
`series.dt.weekday`	Returns the day of the week
`series.dt.is_month_start`	Returns true if it is the first day of the month
`series.dt.strftime()`	Returns an array of formatted strings specified by a `date_format`

https://pandas.pydata.org/pandas-docs/stable/api.html#datetimelike-properties

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

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

```
def addTwo( n ):
    return n + 2

odds.apply( addTwo )
```

```
odds.apply( lambda x: x + 1 )
```

Questions?

In Class Exercise

Please take 20 minutes and complete
Worksheet 1.1: Working with One Dimensional Data

Exercise 1

```
email_series = pd.Series(emails)
filtered_emails =
email_series[email_series.str.contains('.edu')]
```

```
2           mdixon2@cmu.edu
11      jjonesb@arizona.edu
15     eberryf@brandeis.edu
```

```
accounts = filtered_emails.str.split('@').str[0]
```

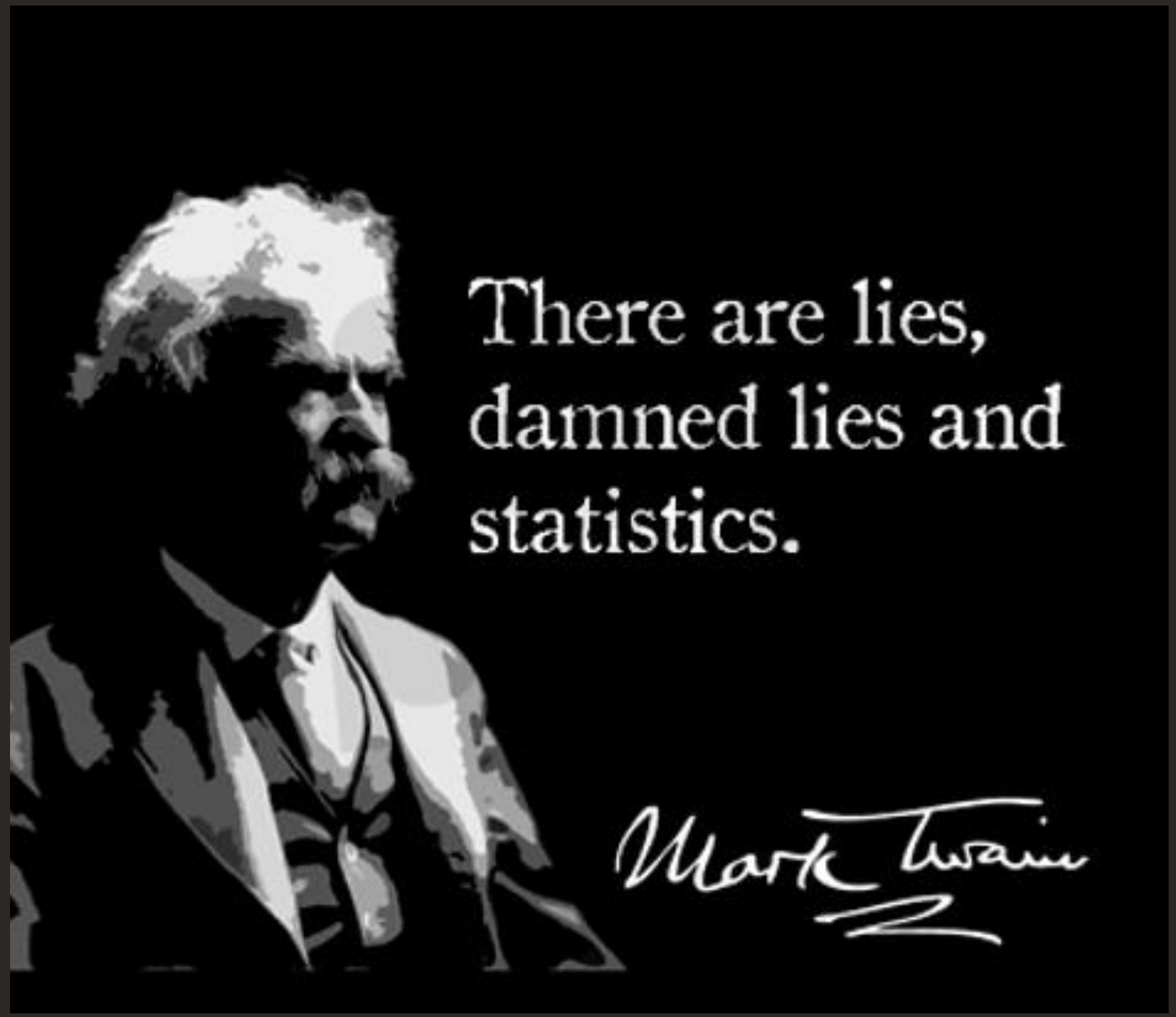
```
2      mdixon2
11     jjonesb
15     eberryf
dtype: object
```

Exercise 2

```
pounds = pd.Series( weights )
kilos = pounds.apply( poundsToKilograms )
```

Exercise 3

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


There are lies,
damned lies and
statistics.
Mark Twain

- **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.

Intro Stats: IQR

The **interquartile range** is a measure of where the “middle fifty” is in a data set. Where a range is a measure of where the beginning and end are in a set, **an interquartile range is a measure of where the bulk of the values lie.**

```
IQR = data.quantile(0.75) - data.quantile(0.25)
```

http://www.statisticshowto.com/probability-and-statistics/interquartile-range/

Intro Stats: Variance

The **variance** of a set of values, σ^2, is the average of the square of the difference of the values in the dataset from the dataset's mean.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}$$

```
data.var(axis=0)
```

Intro Stats:
Std. Deviation

The standard deviation, σ, is the square root of the variance:

We use the **standard deviation** much more regularly than the **variance** because it is on the same scale as the original data:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

```
data.std(axis=0)
```

Intro Stats: Covariance

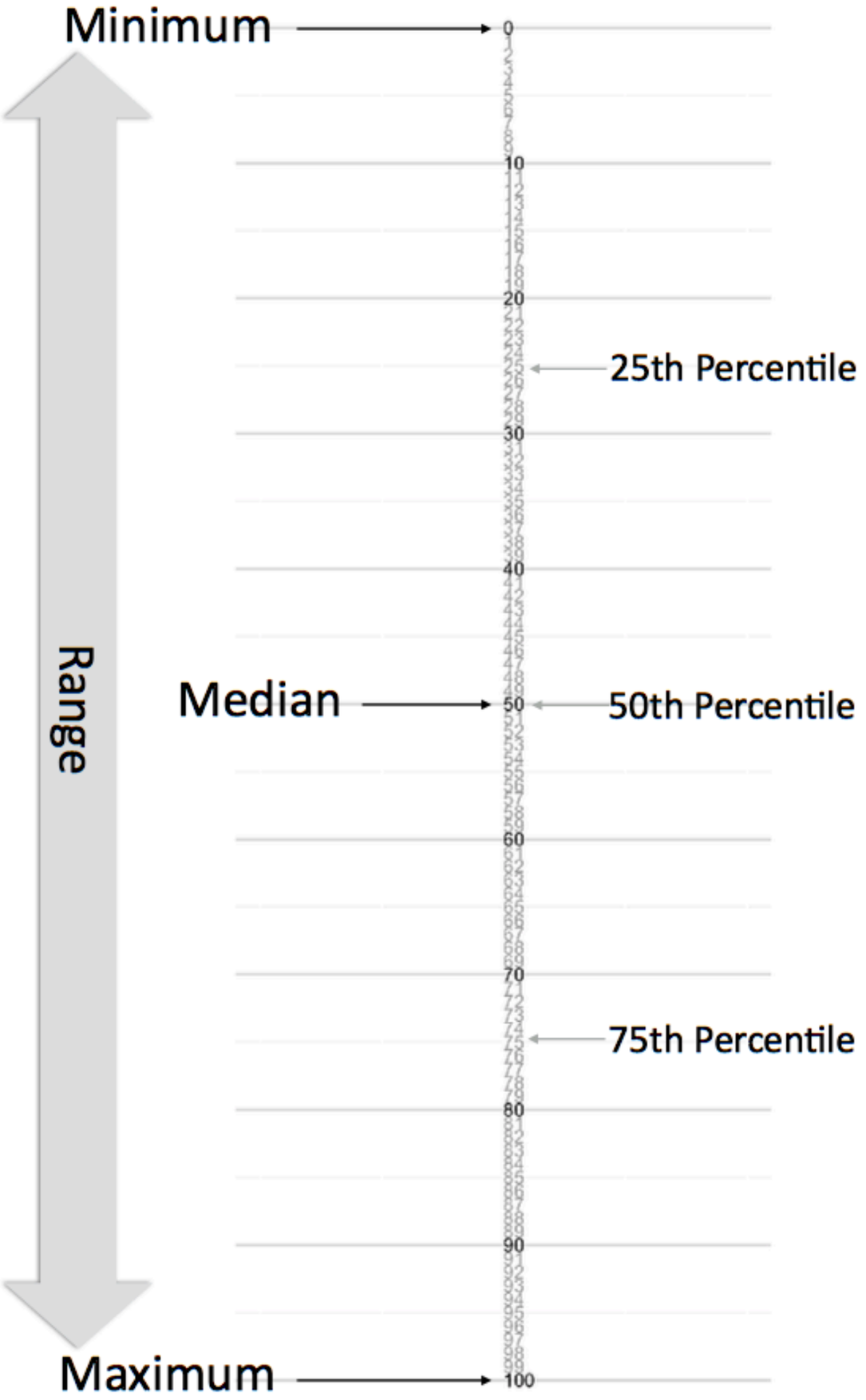
Covariance, like the variance, is a measure of spread, however it also measures how closely two datasets track each other.

- **Covariance** is a squared quantity, so it is not on the same scale as the mean
- **Covariance** of different pairs of variables can have completely different scales

$$Cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{n}$$

Tukey 5 Number Summary

- **Minimum:** The smallest value in the dataset
- **Lower Quartile:** Smallest 25% of the dataset
- **The Median:** The middle value of the dataset
- **Upper Quartile:** The largest 25% of the dataset
- **Maximum:** The largest value in the dataset


Minimum
0
10
20
25th Percentile
30
40
Median
50
50th Percentile
60
70
75th Percentile
80
90
Maximum
100
Range

`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 standard deviation of a Series

```
Series.describe()
```

Series.describe()

```
>>> random_numbers.describe()
count     50.000000
mean      50.620000
std       30.102471
min        1.000000
25%       25.500000
50%       54.000000
75%       73.000000
max       99.000000
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
```

Finding Unique Values

```
unique_nums = []
for key, value in random_numbers.iteritems():
    if value not in unique_nums:
        unique_nums.append(value)
```

NO!!!

```
series.drop_duplicates()
series.unique()
```

Counting Unique Occurrences

```
series.value_counts()
```

```
series.value_counts(bins=5)
```

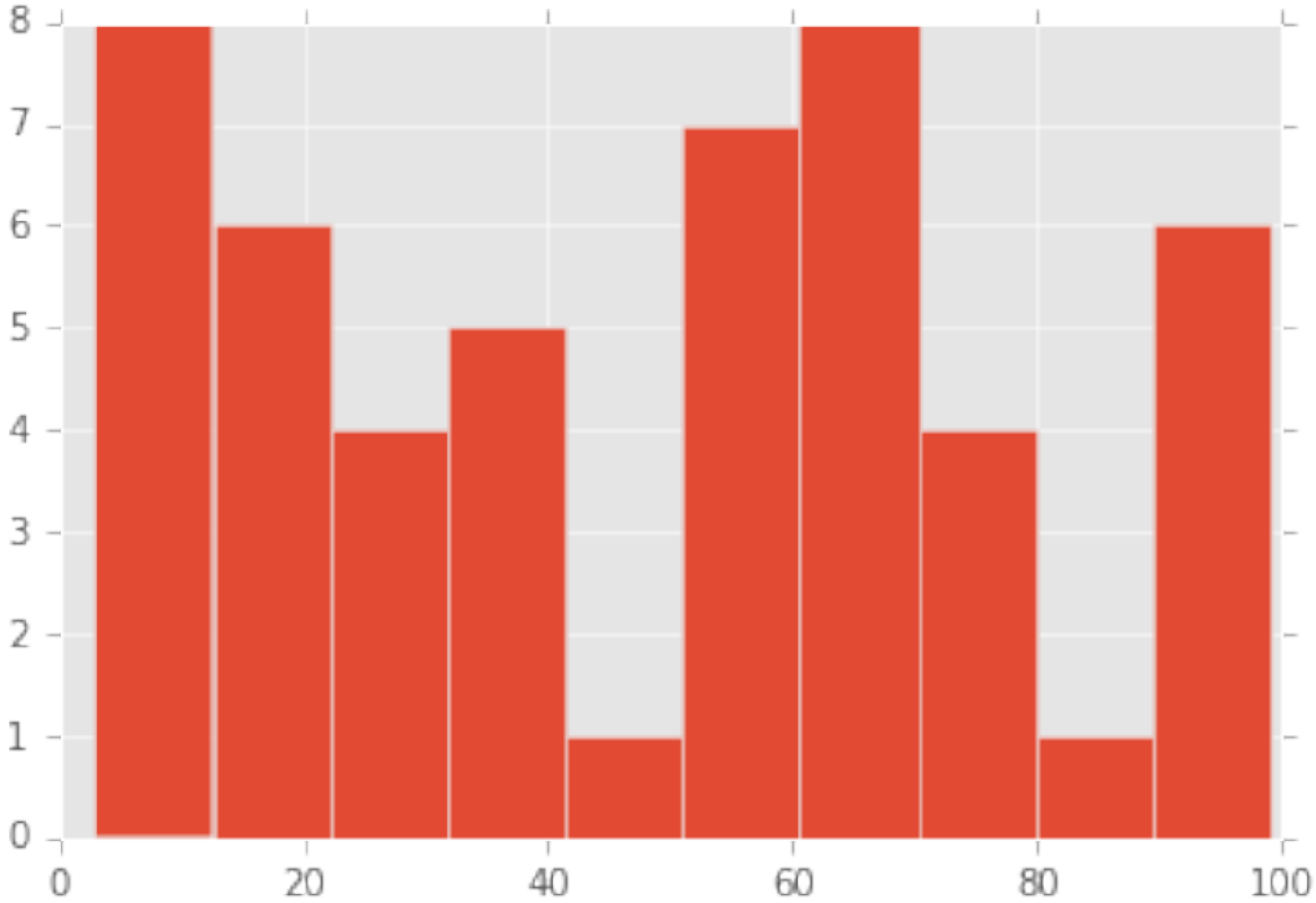
```
series.value_counts(bins=5,
normalize=True)
```

```
series.value_counts(bins=5,
                    normalize=True,
                    dropna=False)
```

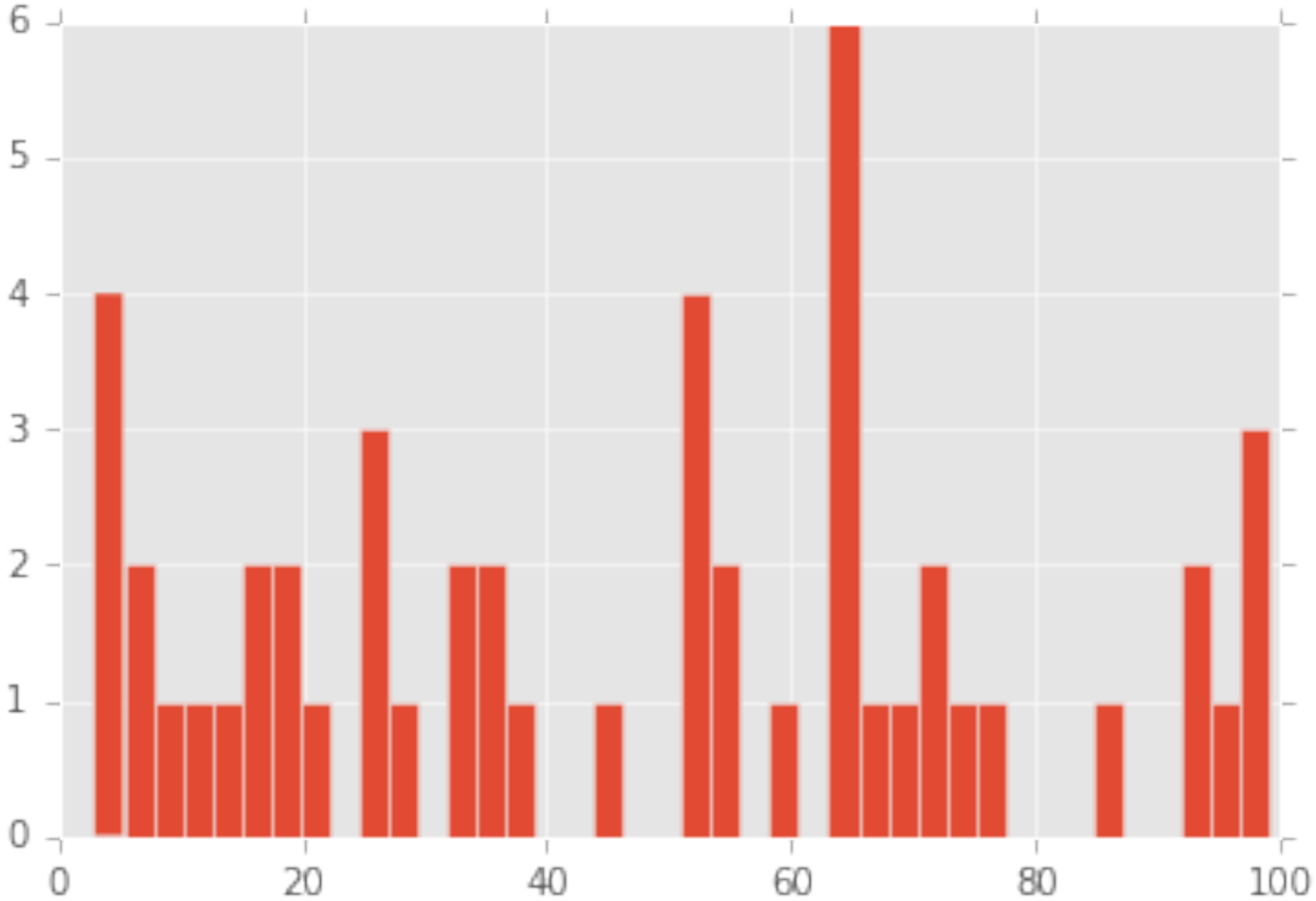
```
In [7]: area_codes2.value_counts()

Out[7]: 833    7
        811    5
        822    5
        899    3
        844    3
        855    2
        dtype: int64
```

```
series.hist()
```

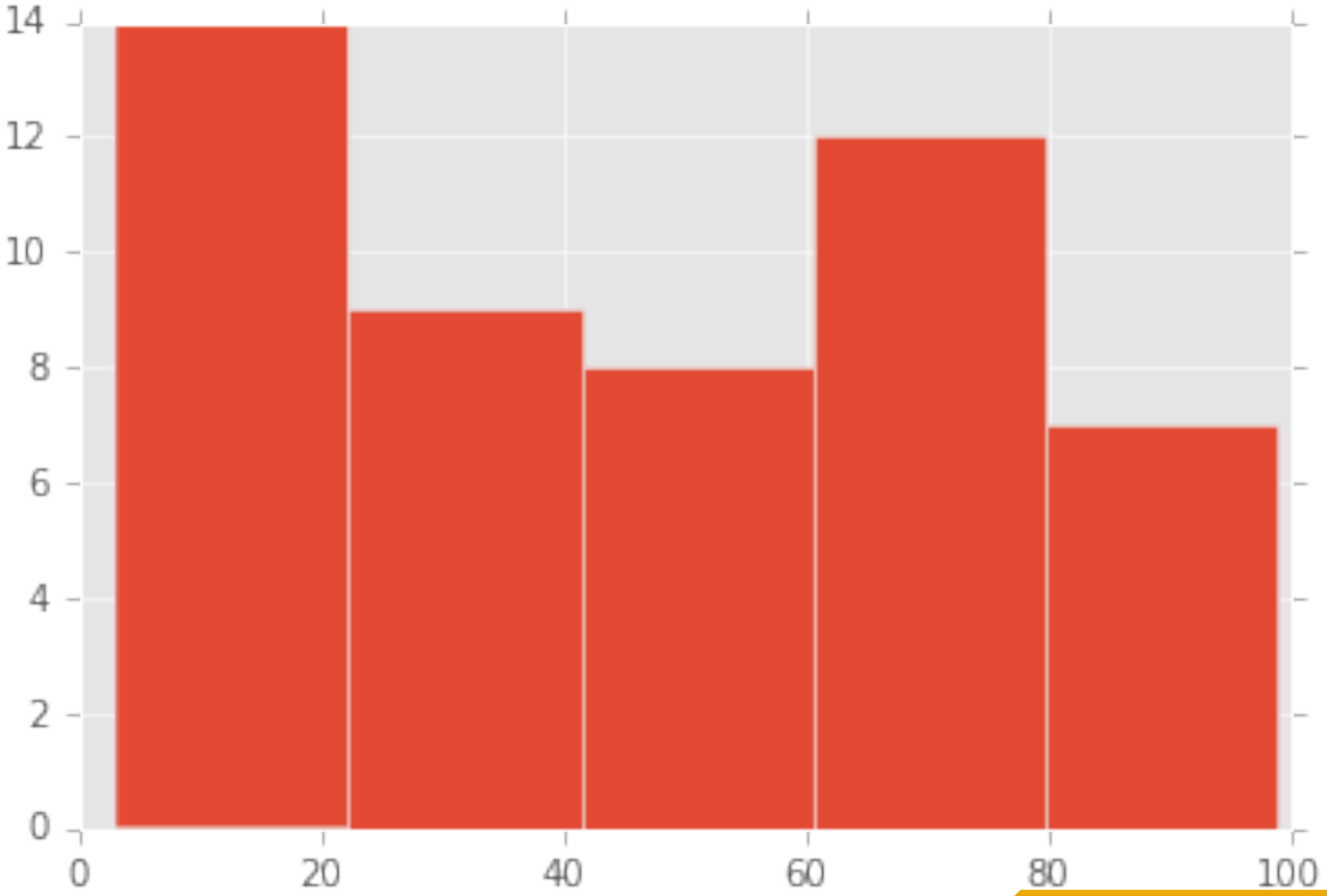
series.hist()



series.hist(bins=40)

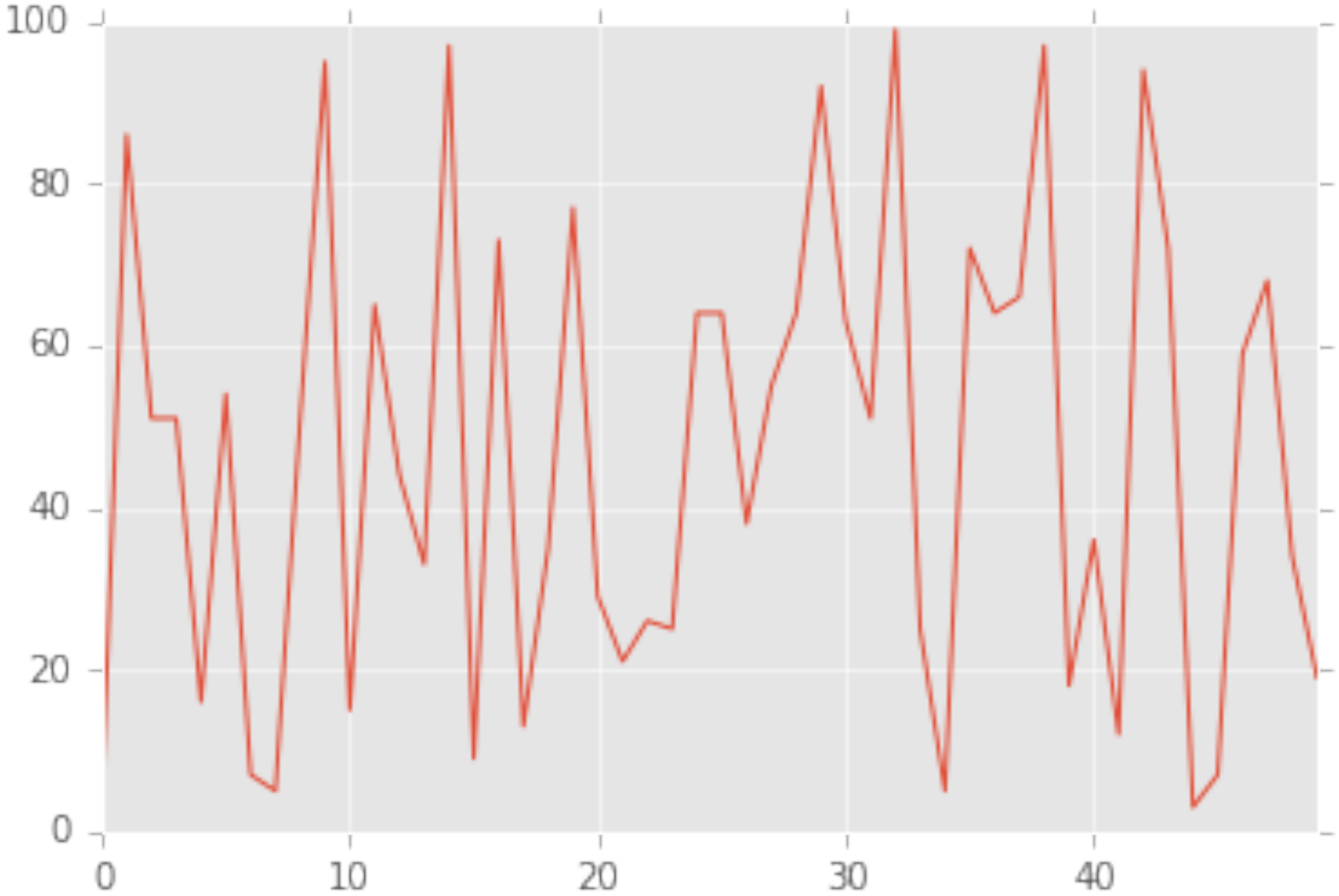


series.hist(bins=5)

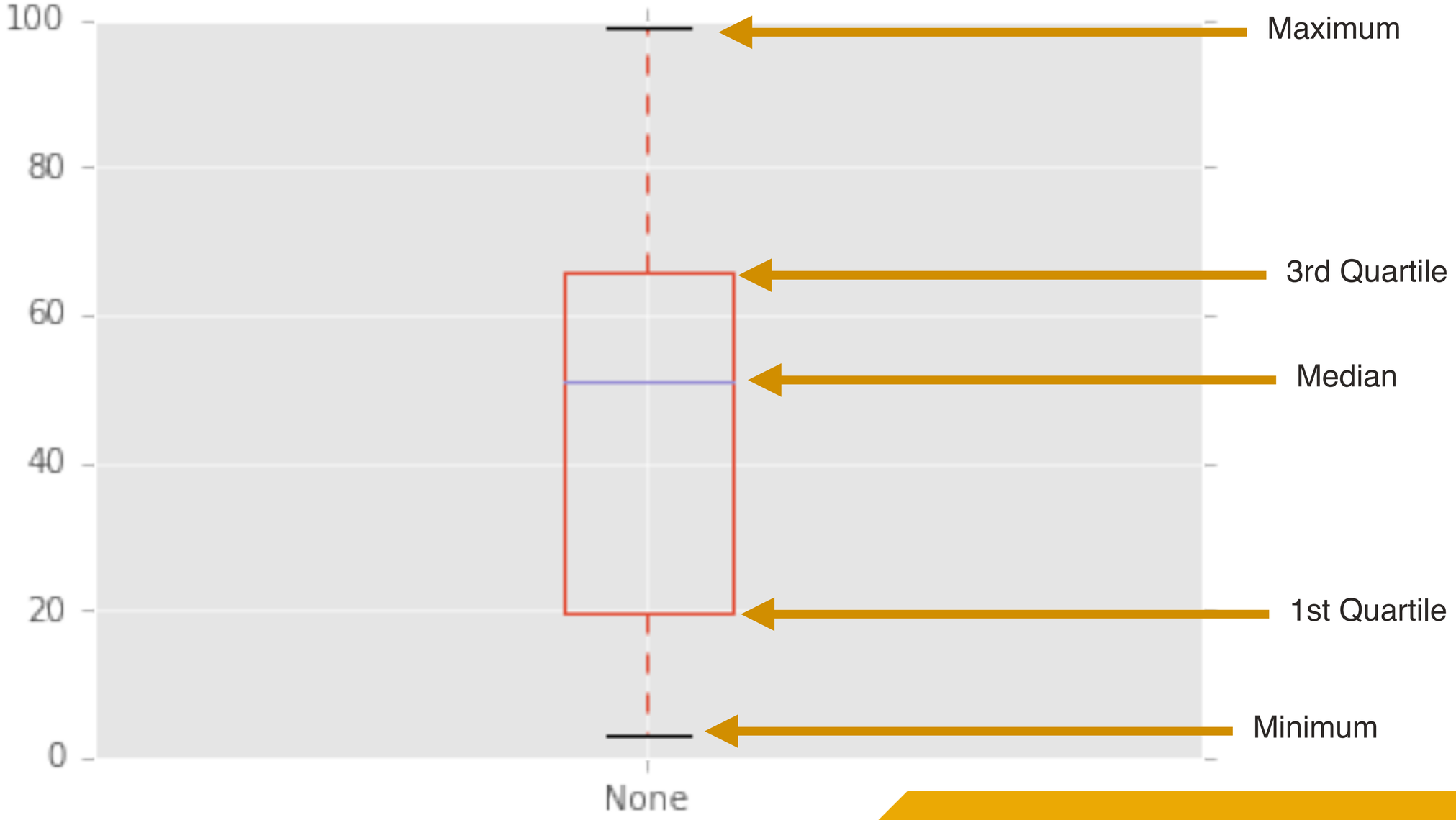


```
series.plot()
```

series.plot()

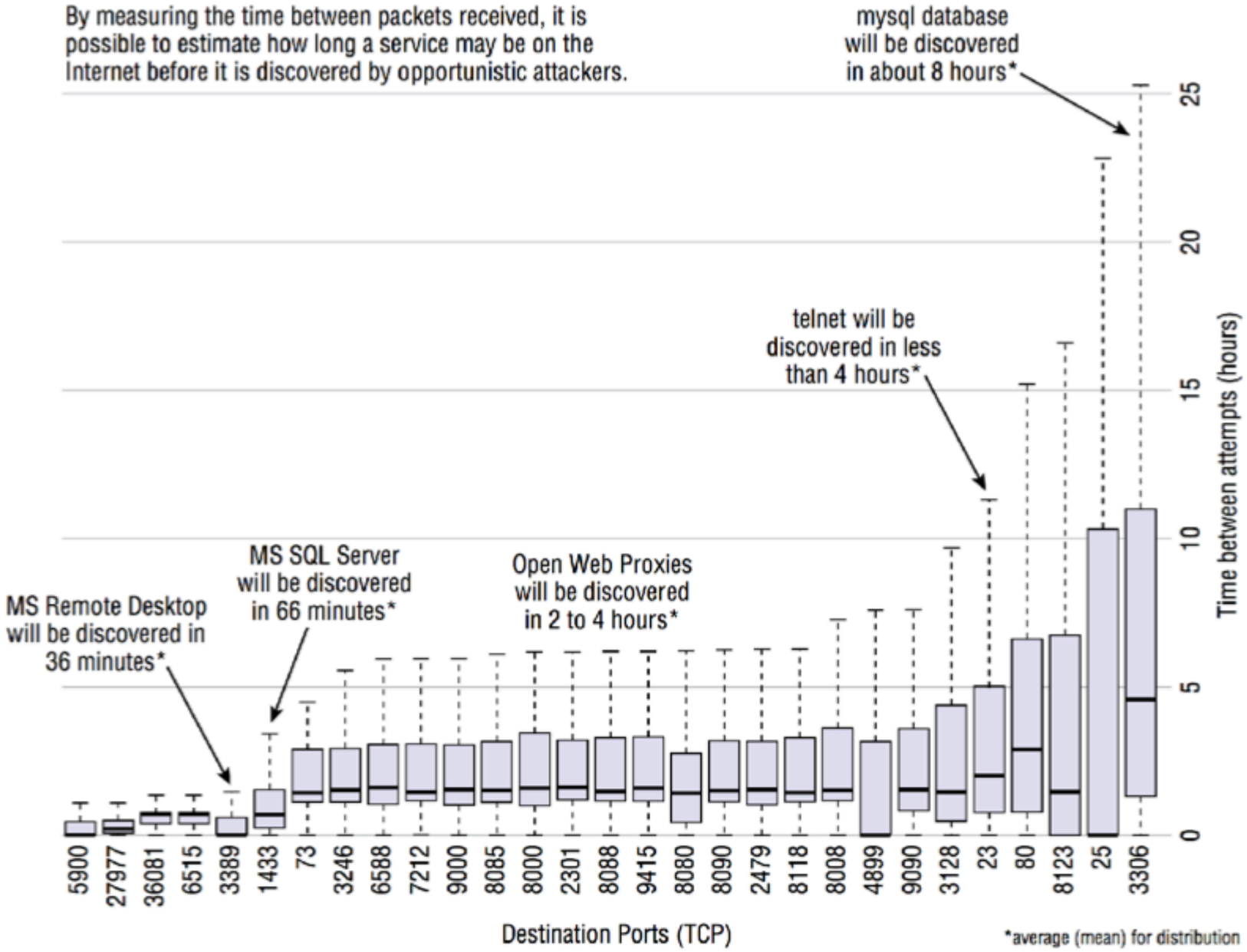


series.plot(kind='box')



How long will a service go undiscovered by opportunistic attackers?

By measuring the time between packets received, it is possible to estimate how long a service may be on the Internet before it is discovered by opportunistic attackers.

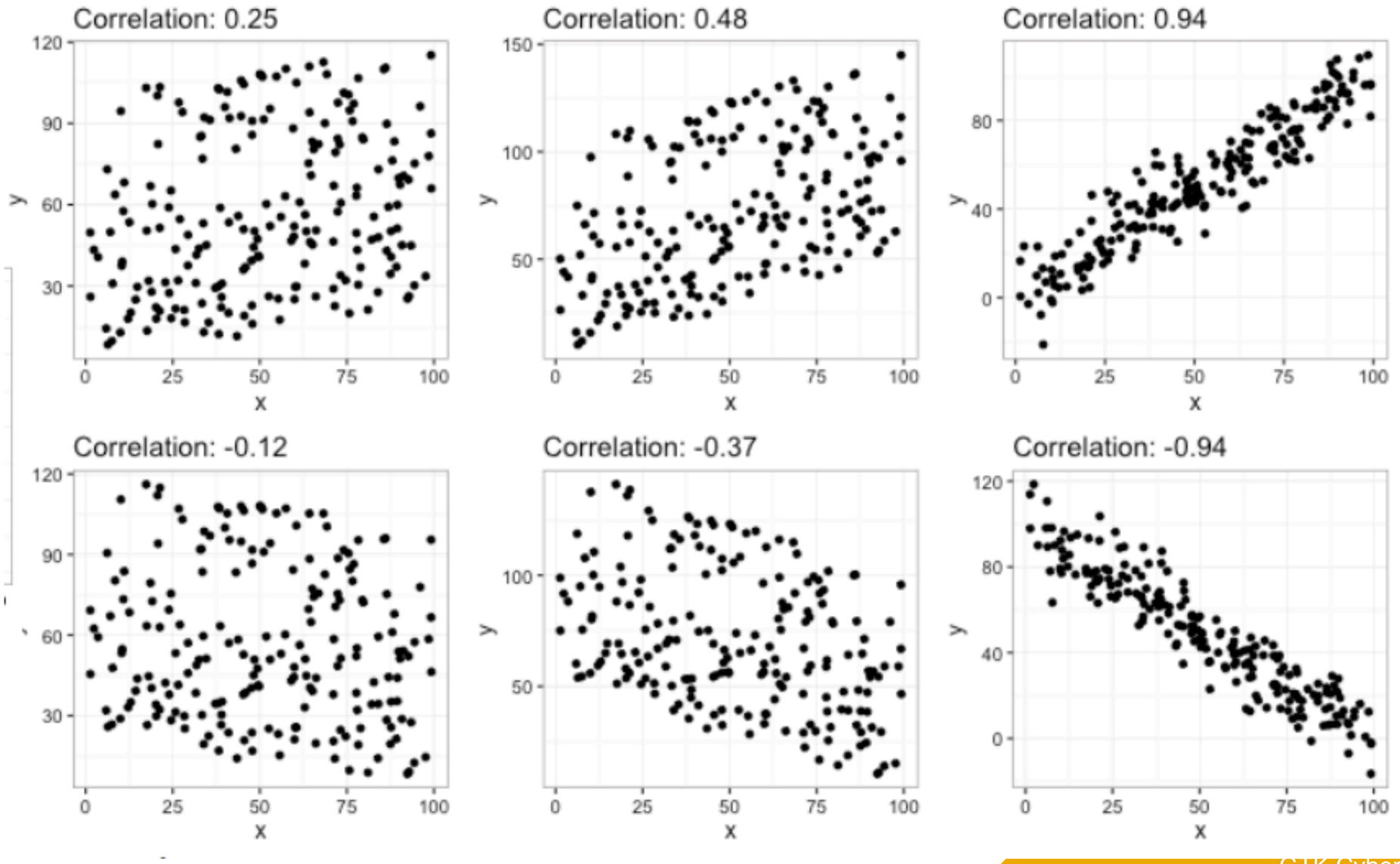


Correlations between Datasets

Correlations

- Measurement of the relationship between **two continuous variables**
- Output is between -1 and 1, with 1 being perfect correlation, -1 is perfect negative correlation, 0 is no correlation.
- Correlation measurement is referred to as r.

Correlations

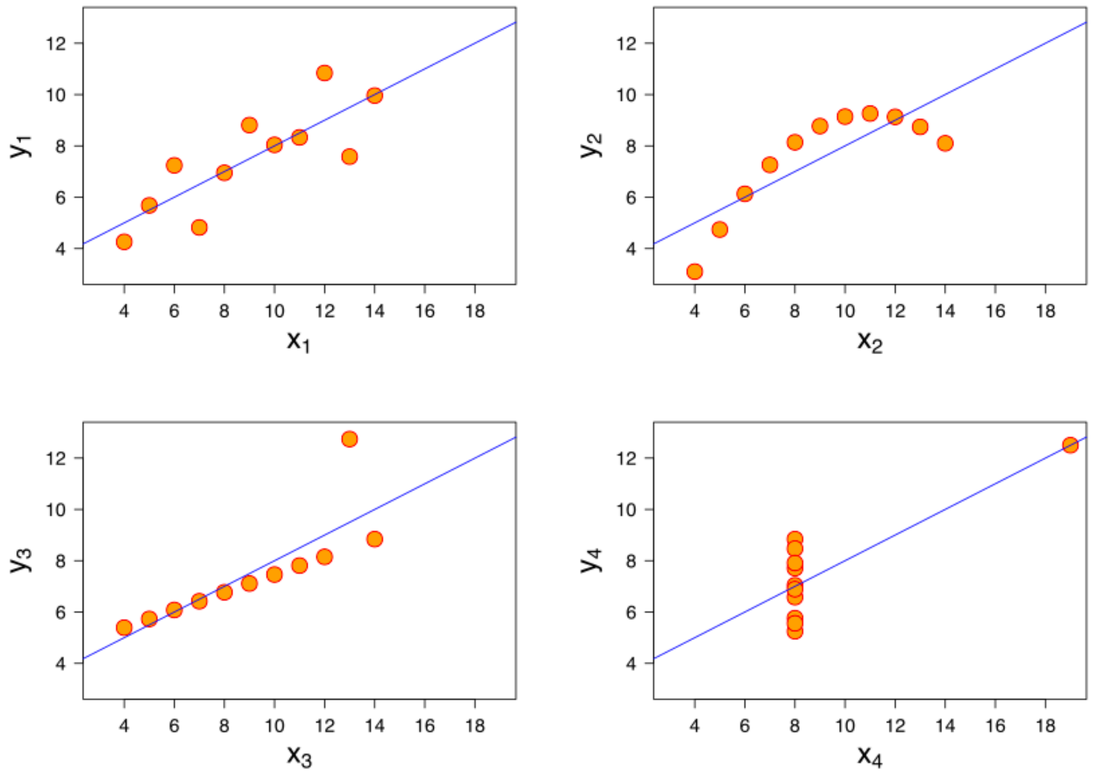


Correlations

```
series1.corr( series2 )
```

Anscombe’s Quartet

Nearly identical descriptive statistics but drastically different distributions when graphed



Questions?

In Class Exercise

Please take 20 minutes and complete

Worksheet 1.2: Exploring One Dimensional Data

We Control the Vertical and the Horizontal!

The Data Frame

The Data Frame

Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30

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

CSV

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

df = pd.read_csv( <url> )
```

Excel

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

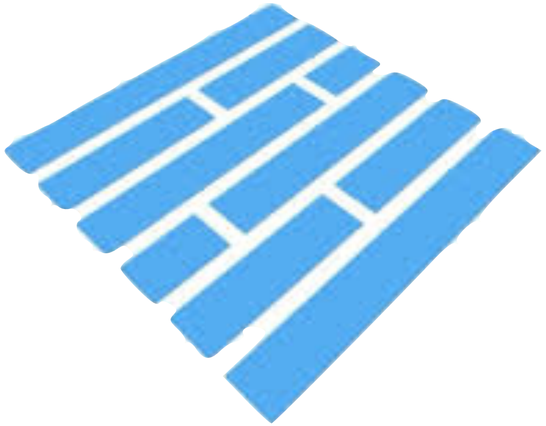
```
[
   {
    “changed":"2015-04-0300:00:00",
    "created":"2014-04-10 00:00:00”,
    “dnssec":"False",
    "expires":"2016-04-10 00:00:00”,
    “isMalicious":0,
    “url":"nuteczki.com"
   },
…
]
```

JSON

```
df = pd.read_json( <file> )
or
df = pd.read_json( <url> )
```

From a Database

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



Parquet

```
df = pd.read_parquet( <file> )
```

HTML

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

XML

```
import requests

user_agent_url = 'http://www.user-agents.org/allagents.xml'
xml_data = requests.get(user_agent_url).content
```

http://www.austintaylor.io/lxml/python/pandas/xml/dataframe/2016/07/08/convert-xml-to-pandas-dataframe/

XML

```
import xml.etree.ElementTree as ET

class XML2DataFrame:

    def __init__(self, xml_data):
        self.root = ET.XML(xml_data)

    def parse_root(self, root):
        return [self.parse_element(child) for child in iter(root)]

    def parse_element(self, element, parsed=None):
        if parsed is None:
            parsed = dict()
        for key in element.keys():
            parsed[key] = element.attrib.get(key)
        if element.text:
            parsed[element.tag] = element.text
        for child in list(element):
            self.parse_element(child, parsed)
        return parsed

    def process_data(self):
        structure_data = self.parse_root(self.root)
        return pd.DataFrame(structure_data)

xml2df = XML2DataFrame(xml_data)
xml_dataframe = xml2df.process_data()
```

http://www.austintaylor.io/lxml/python/pandas/xml/dataframe/2016/07/08/convert-xml-to-pandas-dataframe/

Log Files…

```
070823 21:00:32       1 Connect     root@localhost on test1
070823 21:00:48       1 Query       show tables
070823 21:00:56       1 Query       select * from category
070917 16:29:01      21 Query       select * from location
070917 16:29:12      21 Query       select * from location where id = 1 LIMIT 1
```

```
070823 21:00:32       1 Connect     root@localhost on test1
070823 21:00:48       1 Query       show tables
070823 21:00:56       1 Query       select * from category
070917 16:29:01      21 Query       select * from location
070917 16:29:12      21 Query       select * from location where id = 1 LIMIT 1
```

```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
```

	raw
0	070823 21:00:32 1 Connect root@local...
1	070823 21:00:48 1 Query show tables
2	070823 21:00:56 1 Query select * f...
3	070917 16:29:01 21 Query select * f...
4	070917 16:29:12 21 Query select * f...

```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
logdf['raw'].str.extract('(\d{6}\s\d{2}:\d{2}:\d{2})\s+(\d+)\s(\S+)
\s(.+)', expand=False)
```

	0	1	2	3
0	070823 21:00:32	1	Connect	root@localhost on test1
1	070823 21:00:48	1	Query	show tables
2	070823 21:00:56	1	Query	select * from category
3	070917 16:29:01	21	Query	select * from location
4	070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1

```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
logdf['raw'].str.extract('(?P<date>\d{6}\s\d{2}:\d{2}:\d{2})\s+(?
P<PID>\d+)\s(?P<Action>\S+)\s(?P<Query>.+)', expand=False)
```

	Date	PID	Action	Query
0	070823 21:00:32	1	Connect	root@localhost on test1
1	070823 21:00:48	1	Query	show tables
2	070823 21:00:56	1	Query	select * from category
3	070917 16:29:01	21	Query	select * from location
4	070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1

Web Server Logs

Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)
Gecko/20100101 Firefox/35.0"
```

Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)
Gecko/20100101 Firefox/35.0"
```

```
import apache_log_parser
line_parser = apache_log_parser.make_parser("%h %l %u %t
\"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"")
```

Web Server Logs

```
import apache_log_parser
line_parser = apache_log_parser.make_parser("%h %l %u %t
\"%r\" %>s %b \"%{Referer}i\" \”%{User-agent}i\”")

server_log = open(“../data/hackers-access.httpd”, "r")
parsed_server_data = []
for line in server_log:
    data = {}
    data = line_parser(line)
    parsed_server_data.append( data )

server_df = pd.DataFrame( parsed_server_data  )
```

Web Server Logs

er	request_first_line	request_header_referer	request_header_user_agent	request_http_ver	request_method	request_url
	GET /linux/doing-pxe-without-dhcp-control HTTP...	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	GET	/linux/doing-pxe-v
	GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.0	GET	/join_form
	POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	POST	/join_form
	GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.0	GET	/join_form
	POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	POST	/join_form
	GET /acl_users/credentials_cookie_auth/require...	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	GET	/acl_users/creden

Windows Event Logs

```
pip install python-evtx
```

```
import Evtx.Evtx as evtx

xml = "<Events>"
with evtx.Evtx("window_event_log.evtx") as log:
    for record in log.records():
        xml += record.xml()
xml += "</Events>"
```

Nested Data?

Nested Data?


```
{"time": 1084443427.311224,
 "timestamp": "2004-05-13T10:17:07.311224",
"IP": {
   "version": 4,
   "ttl": 128,
   "proto": 6,
   "options": [],
   "len": 48,
   "dst": "65.208.228.223",
   "frag": 0,
   "flags": 2, "src": “145.254.160.237",
   "chksum": 37355
   },
   "Ethernet": {"src": "00:00:01:00:00:00", "type": 2048, "dst": “fe:ff:20:00:01:00"},
   …
```

Nested Data

```
pd.read_json( 'nested_data.json' )
```

```
pd.read_json( 'nested_data.json' )
```

	DNS	Ethernet	IP	TCP	UDP	time	timestamp
0	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 48, 'version'...	{'window': 8760, 'chksum': 49932, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:07.311224
1	NaN	{'type': 2048, 'dst': '00:00:01:00:00:00', 'sr...	{'dst': '145.254.160.237', 'len': 48, 'version...	{'window': 5840, 'chksum': 23516, 'sport': 80,...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
2	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 40, 'version'...	{'window': 9660, 'chksum': 31076, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
3	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 519, 'version...	{'window': 9660, 'chksum': 43352, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
4	NaN	{'type': 2048, 'dst': '00:00:01:00:00:00', 'sr...	{'dst': '145.254.160.237', 'len': 40, 'version...	{'window': 6432, 'chksum': 33825, 'sport': 80,...	NaN	1.084443e+09	2004-05-13 10:17:08.783340

Nested Data

```
from pandas.io.json import json_normalize
import json
import pandas as pd

with open('nested.json') as data_file:
    pcap_data = json.load(data_file)

df = pd.DataFrame( json_normalize(pcap_data) )
```

```
df = pd.DataFrame( json_normalize(pcap_data) )
```

...	TCP.seq	TCP.sport	TCP.urgptr	TCP.window	L
...	951057939.0	3372.0	0.0	8760.0	
...	290218379.0	80.0	0.0	5840.0	
...	951057940.0	3372.0	0.0	9660.0	
...	951057940.0	3372.0	0.0	9660.0	
...	290218380.0	80.0	0.0	6432.0	

ElasticSearch & Splunk

splunk>

ElasticSearch & Splunk

```
pip install huntlib
```

```
e = ElasticDF(
  url="https://localhost:9200",
  ssl=True,
  username="myuser",
  password="mypass"
)
df = e.search_df(
  lucene="proto:tcp AND port:80",
  index="myindex-*",
  days=1,
  normalize=False
)
```

```
s = SplunkDF(
  host="<splunk_ip>",
  username="myuser",
  password="mypass"
)
df = s.search_df(
  spl='search index=win_events src="10.9.*.*"',
  start_time=datetime.now()-timedelta(days=2),
  end_time=datetime.now()
)
```

Manipulating a DataFrame

```
series = df['column1']
```

Returns a **series**

```
df['ip'].value_counts().head()
```

Creating a new Column

```
df['new_col'] = df['column1'] + 3
```

Two Ways of Accessing Columns

```
series = df[‘column1']
```

```
series .
```

```
column1
```

Don't use the dots!

```
df = df[['column1','column2','column3']]
```

Returns a DataFrame

Filtering a DataFrame

```
df[<boolean condition>]
```

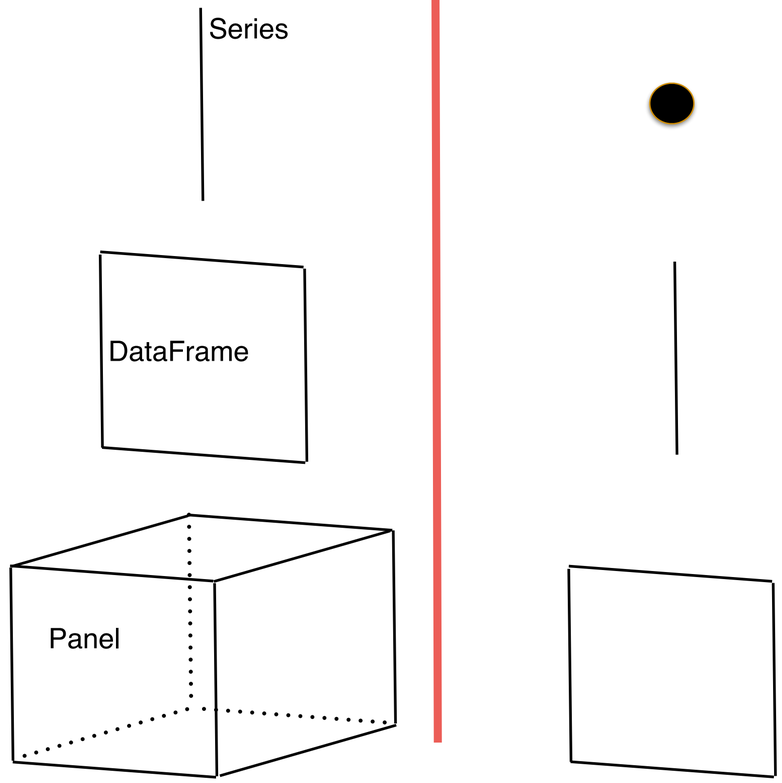
```
df[['col1','col2']][df['col3'] > 5]
```

```
data.loc[<index>]
data.loc[<list of indexes>]
data.sample(<n>)
```

- `data.apply( <function> )`

IF
Call Apply On...

THEN
Function
Receives



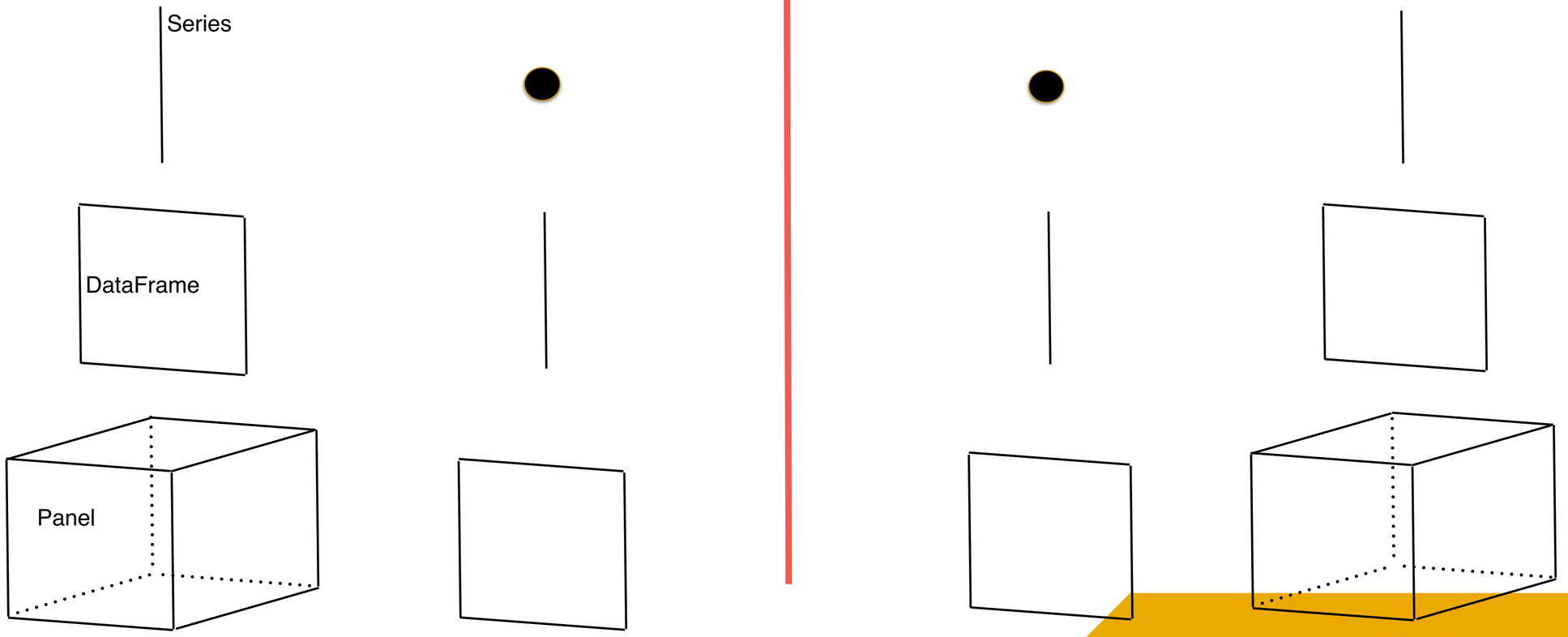
- `data.apply( <function> )`

IF
Call Apply On...

THEN
Function
Receives

IF
Function
Returns

THEN
Apply Returns...



In Class Exercise

Please complete Worksheet 2.1: Exploring Two Dimensional Data

Exercise 1

#Data1


```
{“first_name":{“0":"Robert","1":"Steve","2":"Anne","3":"Alice"},
“last_name":{"0":"Hernandez","1":"Smith","2":"Raps","3":"Muller"},
“birthday":{"0":"5\\/3\\/67","1":"8\\/4\\/84","2":"9\\/13\\/
91","3":"4\\/15\\/75"}
}
```


```
df1 = pd.read_json('../data/data1.json')
```

Exercise 1

#Data2

```
{“0”:{"first_name":"Robert","last_name":"Hernandez","birthday":"5\\/3\
\/67"},
“1":{"first_name":"Steve","last_name":"Smith","birthday":"8\\/4\\/
84"},
“2":{"first_name":"Anne","last_name":"Raps","birthday":"9\\/13\\/91"},
"3":{"first_name":"Alice","last_name":"Muller","birthday":"4\\/15\\/
75"}}
```

```
df2 = pd.read_json('../data/data2.json', orient='index')
```

Exercise 1

```
#Data3
[
{“first_name":"Robert","last_name":"Hernandez","birthday":"5\\/3\\/
67"},
{"first_name":"Steve","last_name":"Smith","birthday":"8\\/4\\/84"},
{"first_name":"Anne","last_name":"Raps","birthday":"9\\/13\\/91"},
{"first_name":"Alice","last_name":"Muller","birthday":"4\\/15\\/75"}]
```

```
df3 = pd.read_json('../data/data3.json',
orient='columns')
```

Exercise 1

#Data4

```
{“columns":["first_name","last_name","birthday"],
“index":[0,1,2,3],
"data":[["Robert","Hernandez","5\\/3\\/67"],["Steve","Smith","8\\/4\\/
84"],["Anne","Raps","9\\/13\\/91"],["Alice","Muller","4\\/15\\/75"]]}
```

```
df4 = pd.read_json('../data/data4.json', orient='split')
```

Exercise 2

```
#Write the functions
def get_os(x):
    user_agent = parse(x)
    return user_agent.os.family

def get_browser(x):
    user_agent = parse(x)
    return user_agent.browser.family
```

Exercise 2

```
#Apply the functions to the dataframe
server_df['os'] =
server_df['request_header_user_agent'].apply( get_os )
server_df['browser'] =
server_df['request_header_user_agent'].apply( get_browser )
```

Exercise 2

```
#Apply the functions to the dataframe
server_df['os'] =
server_df['request_header_user_agent'].apply( get_os )
server_df['browser'] =
server_df['request_header_user_agent'].apply( get_browser )
```

```
#Get the top 10 values
server_df['os'].value_counts().head(10)
```

Exercise 2

```
#Get the top 10 values
server_df['os'].value_counts().head(10)
```

```
Windows 7          2041
Windows Vista       500
Windows XP          423
Windows 8.1         221
Linux               125
Mac OS X             80
Chrome OS            60
Ubuntu                6
```

Exercise 3

```
bots = pd.read_csv('../data/dailybots.csv')
gov_bots = bots[['botfam', 'hosts']][bots['industry'] ==
"Government/Politics"]

gov_bots.groupby('botfam', as_index=False).sum()
```

Questions?

Merging Data Sets

Union

1	2	3
4	5	6
7	8	9

`data.T`

1	4	7
2	5	8
3	6	9

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

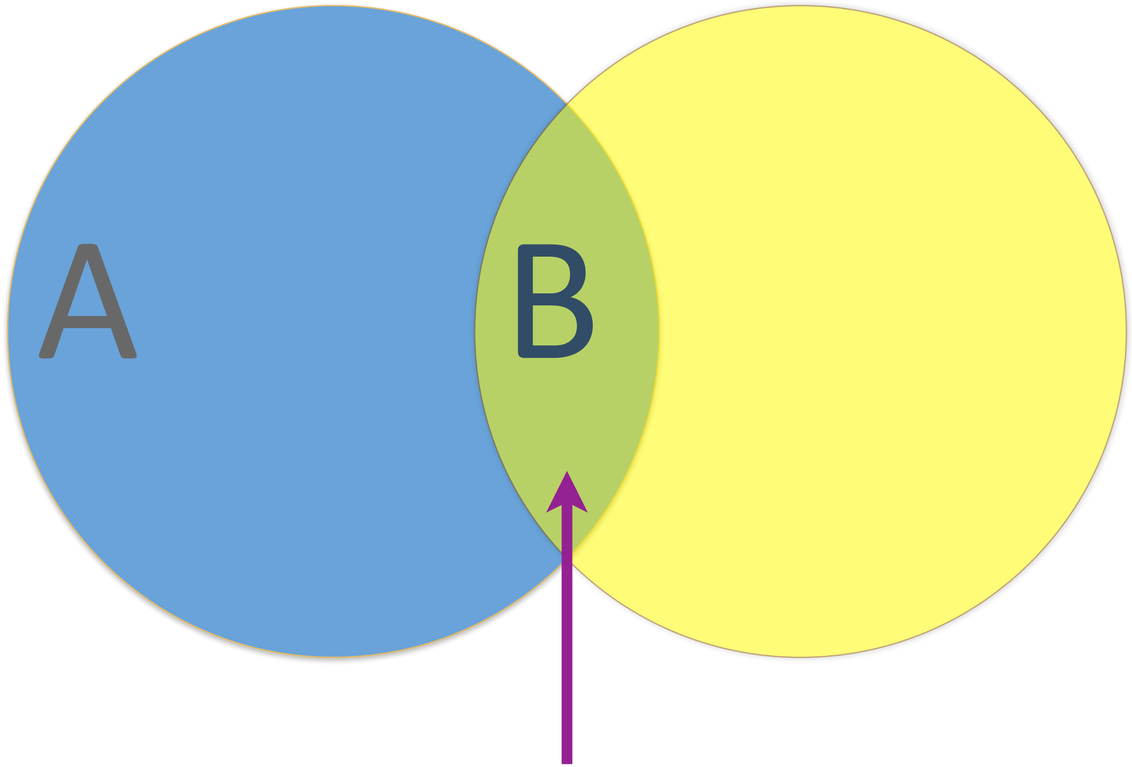
combinedSeries

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 )
```

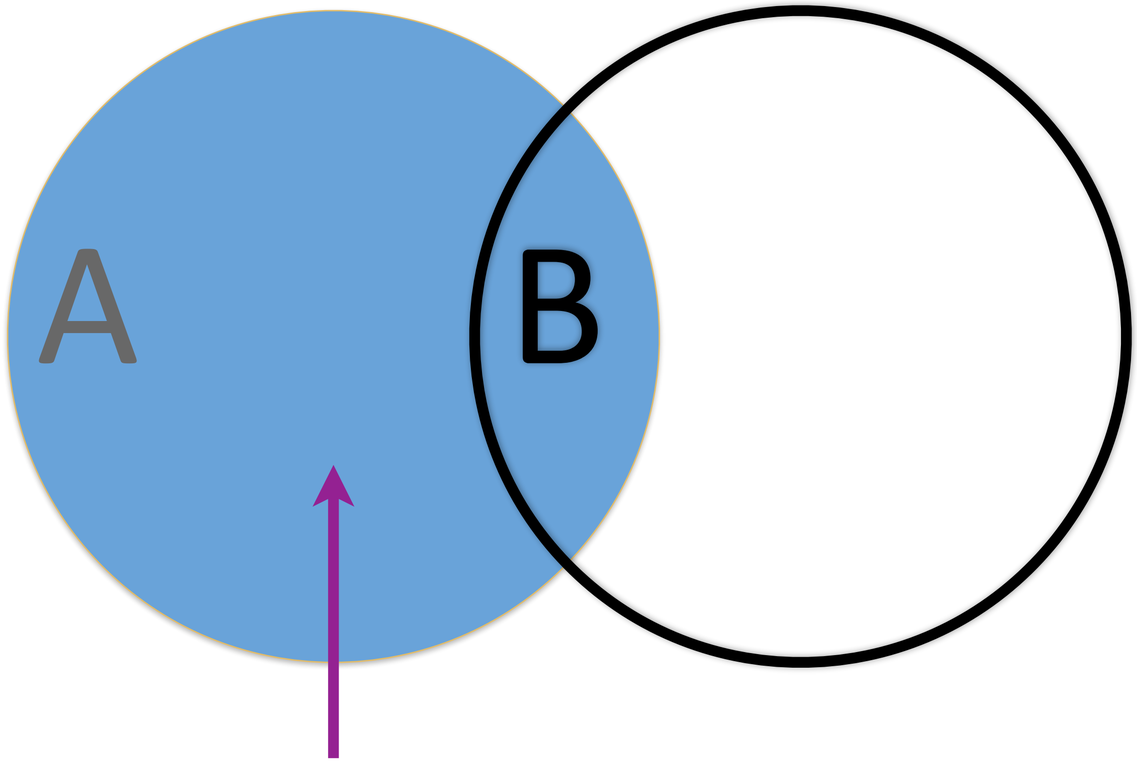
Joins

Inner Join (Intersection)



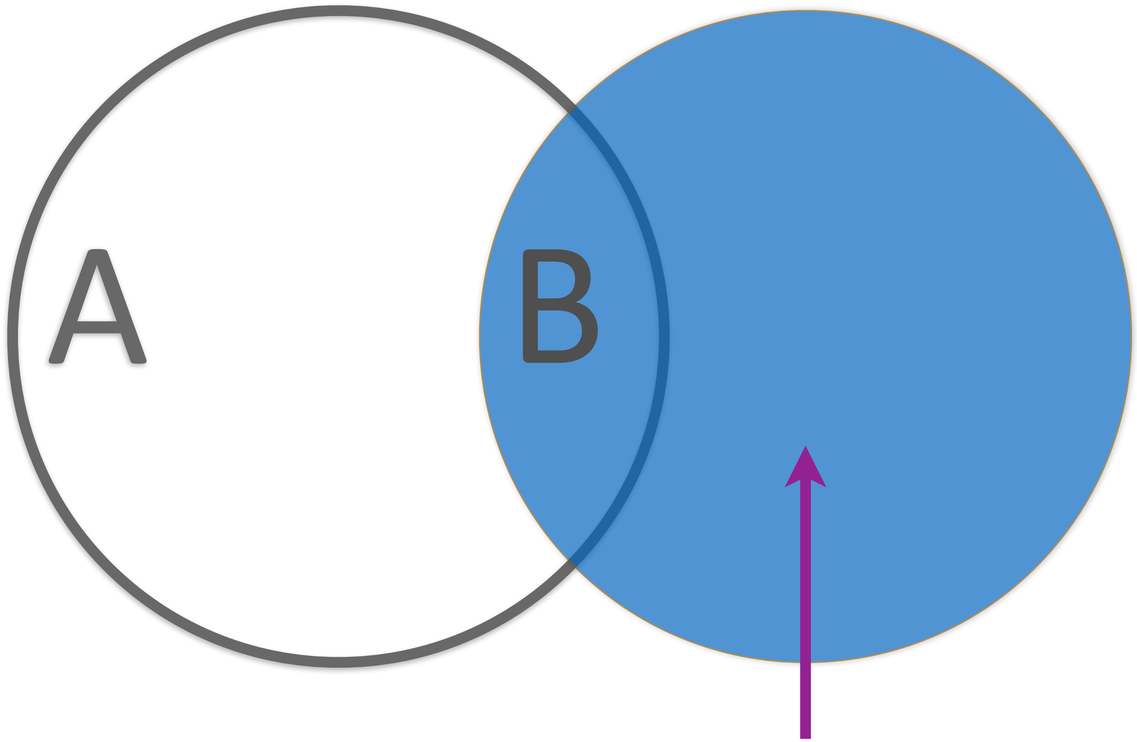
Inner Join

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> )
```

Grouping and Aggregating Data

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

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

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(field or list of fields)
```

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby('src_ip')['port'].count()
```

src_ip	count
192.168.20.1	1
192.168.20.2	3

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(['date','src_ip'])['port'].count()
```

Multi-Index

```
date          src_ip
2018-06-21    192.168.20.1     1
              192.168.20.2     2
2018-06-22    192.168.20.2     1
```

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(['date','src_ip'], as_index=False)
['port'].count()
```

	date	src_ip	port
0	2018-06-21	192.168.20.1	1
1	2018-06-21	192.168.20.2	2
2	2018-06-22	192.168.20.2	1

Grouping Functions

Pivot

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

http://jalammar.github.io/visualizing-pandas-pivoting-and-reshaping/

Grouping Functions

Melt

df3

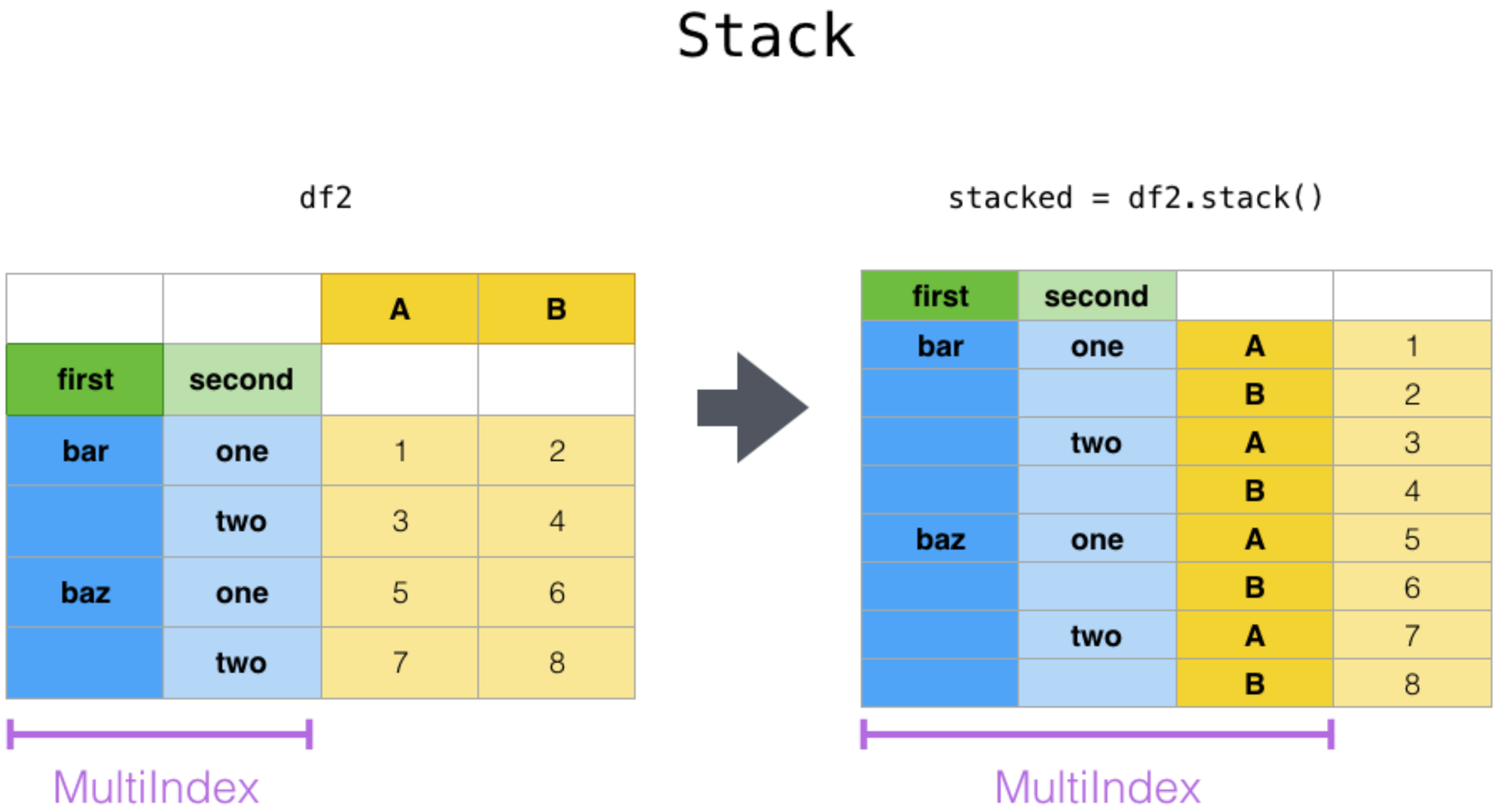
	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150

```
df3.melt(id_vars=['first', 'last'])
```

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

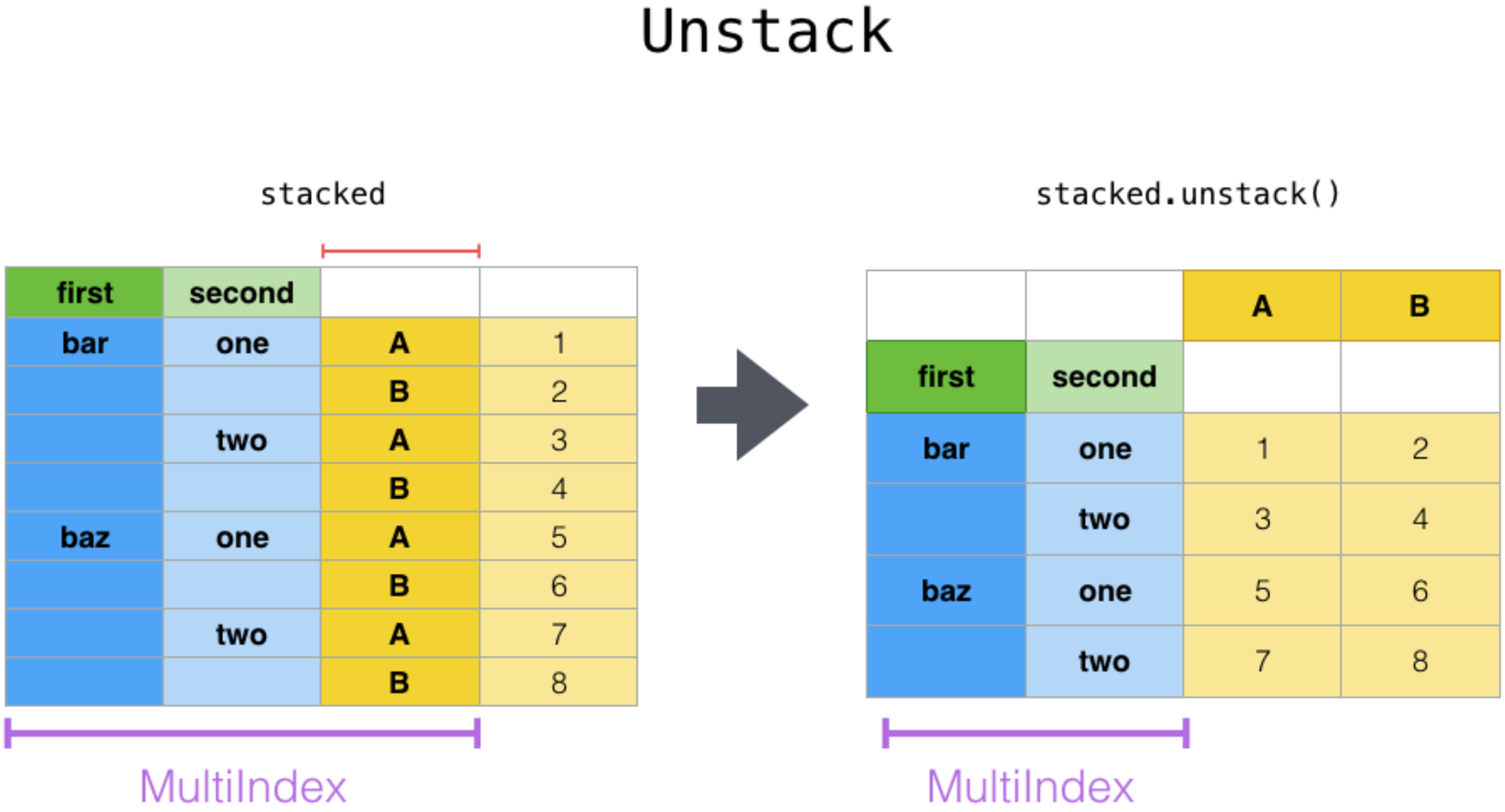
http://jalammar.github.io/visualizing-pandas-pivoting-and-reshaping/

Grouping Functions



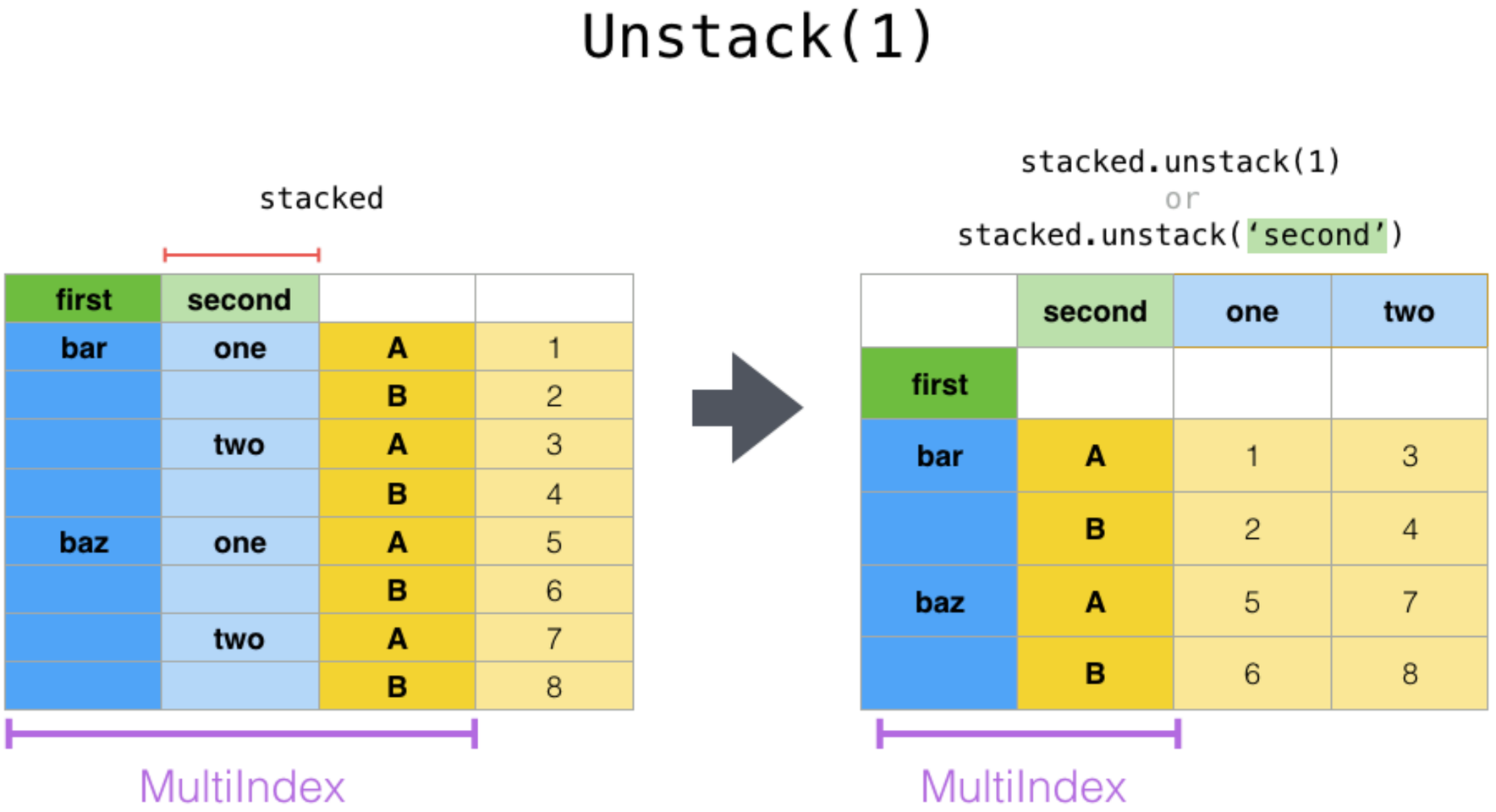
http://jalammar.github.io/visualizing-pandas-pivoting-and-reshaping/

Grouping Functions



http://jalammar.github.io/visualizing-pandas-pivoting-and-reshaping/

Grouping Functions



http://jalammar.github.io/visualizing-pandas-pivoting-and-reshaping/

In-Class Exercise

Please complete Worksheet 2.2: Exploratory Data Analysis

Life After Pandas?
What's next? … well

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

– https://www.coursera.org/course/ml

“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

Artificial Intelligence

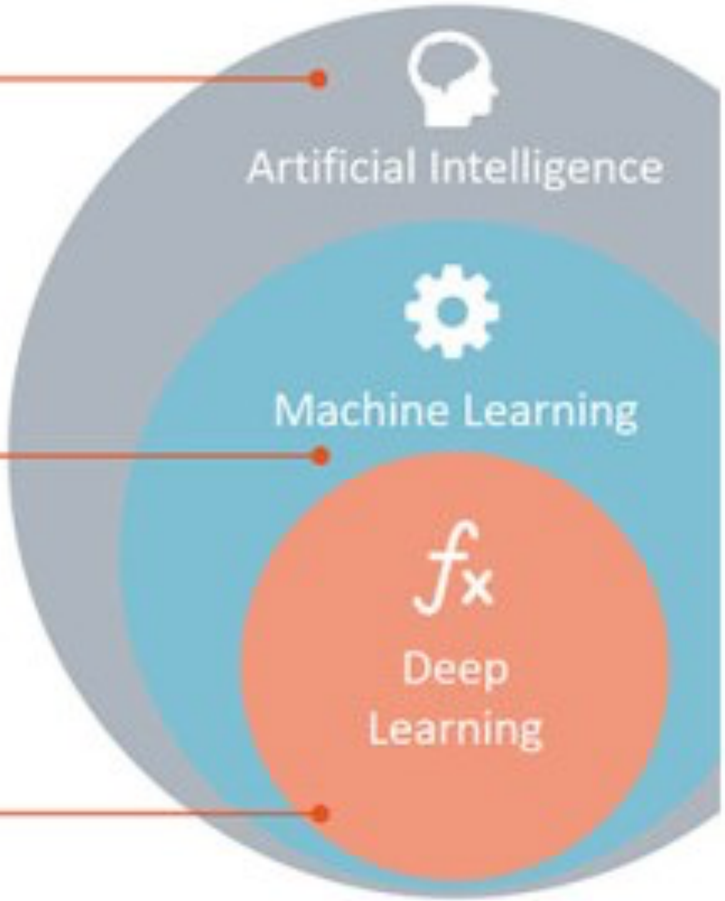
Any technique which enables computers to mimic human behavior.

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning

Subset of ML which make the computation of multi-layer neural networks feasible.



https://www.kdnuggets.com/2017/07/rapidminer-ai-machine-learning-deep-learning.html

Machine Learning Problems

- **Supervised Learning:** Supervised Learning is a class of Machine Learning in which a model is "trained" using a set of pre-existing labeled data.
- **Unsupervised Learning:** A class of Machine Learning algorithms in which a model is built without the use of labeled data.

Machine Learning Problem Types

- **Classification**: Assigning or predicting a observation's membership in discrete class
- **Regression:** Predicting a continuous value based on the observations' features
- **Clustering:** Identifying groupings within a dataset
- **Dimensionality Reduction:** Reducing the number of variables in a feature set

Applications to Security

Regression Example

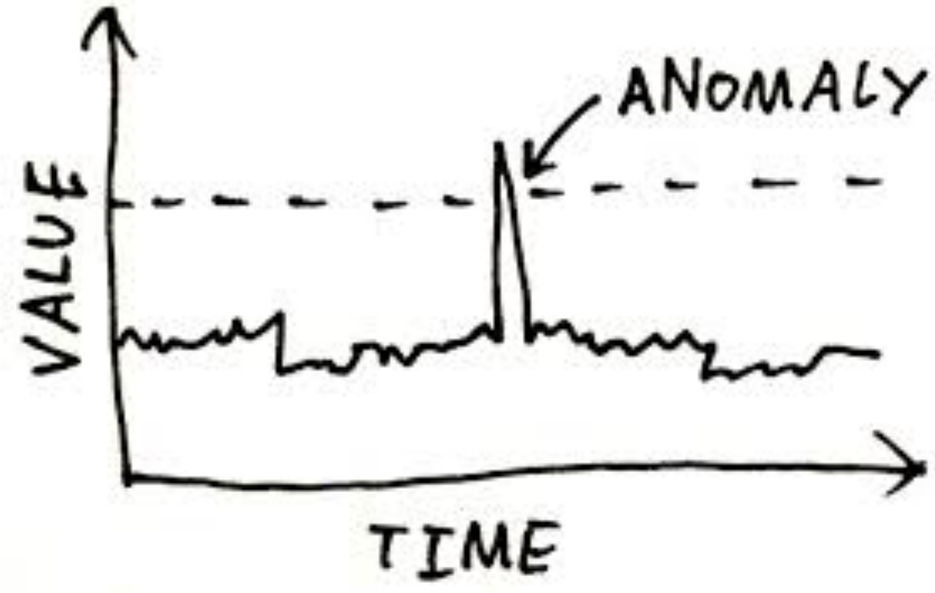
Server Capacity Prediction: Regression analysis can be used to predict a server's capacity (or CPU usage) based on the server's historical performance.

https://www.researchgate.net/publication/256645877_LiRCUP_Linear_Regression_based_CPU_Usage_Prediction_Algorithm_for_Live_Migration_of_Virtual_Machines_in_Data_Centers

https://jgreenemi.com/predicting-capacity-with-linear-regression-ml/

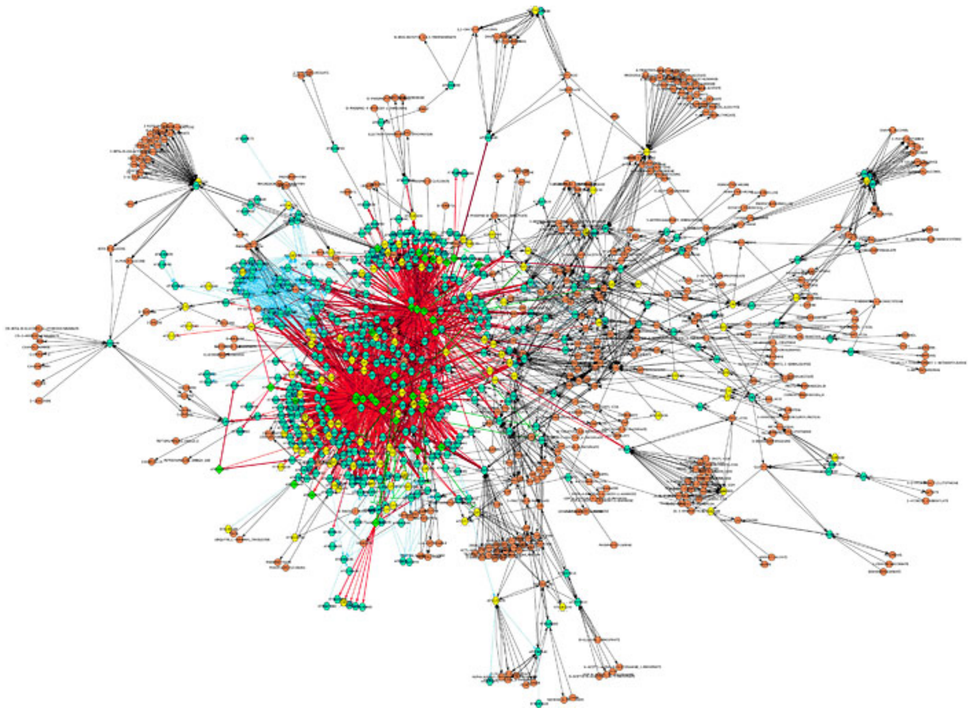
Clustering Example

Anomaly Detection: Clustering techniques can be used to detect anomalous traffic or loads or anything really.



Network-Based Intrusion Detection

- Derive Features from Network Traffic Captures “pcap” at packet level or NetFlow level (tools: tshark, tcpdump, bro…)
- Example Features based on header information: 2s-windowing of connections > duration, protocol, src and dat bytes, service.
- Get data sets: http://www.netresec.com/?page=PcapFiles, https://maccdc.org/, http://www.westpoint.edu/crc/SitePages/DataSets.aspx http://www.unb.ca/cic/research/datasets/,



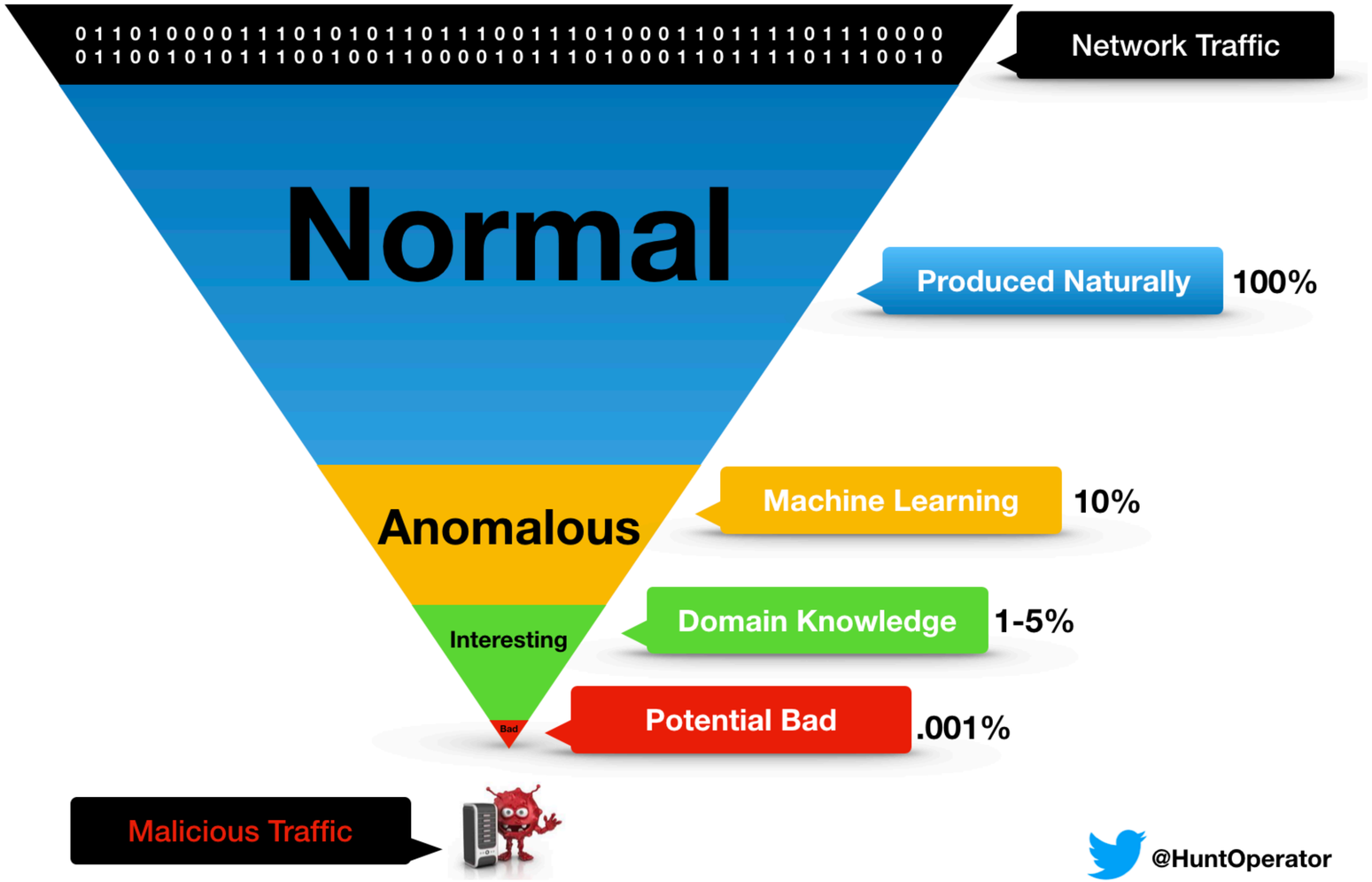
Malware Detection/Classification

- Derive Features from Binary Content and metadata manifest (function calls, string obtained from IDA Disassembler)
- Example Features: opcode count (n-grams), segment count, asm pixel intensity, n-gramming of bytes, function name.
- Featureless Deep Learning with word2vec embedding
- Get open source malware samples: Vx Heaven, Virus Share, Maltrieve, Open Malware

Security Applications of Machine Learning

- Domain Generation Algorithm (DGA) Detection (Classification)
- Malicious URL Detection (Classification)
- Network Traffic: Beaconing Detection (Classification/Clustering)
- Detection of new classes of malware (Classification/Clustering)
- General Network Traffic Anomaly Detection (Classification/Clustering)
- Log Analysis - Anomaly Detection (Classification/Clustering)
- Phishing Detection (Classification)
- Identifying SQL Injection (Classification)
- Identifying XSS cross-site scripting (Classification)
- DOS/DDOS Detection (Classification)
- Authentication (Classification)

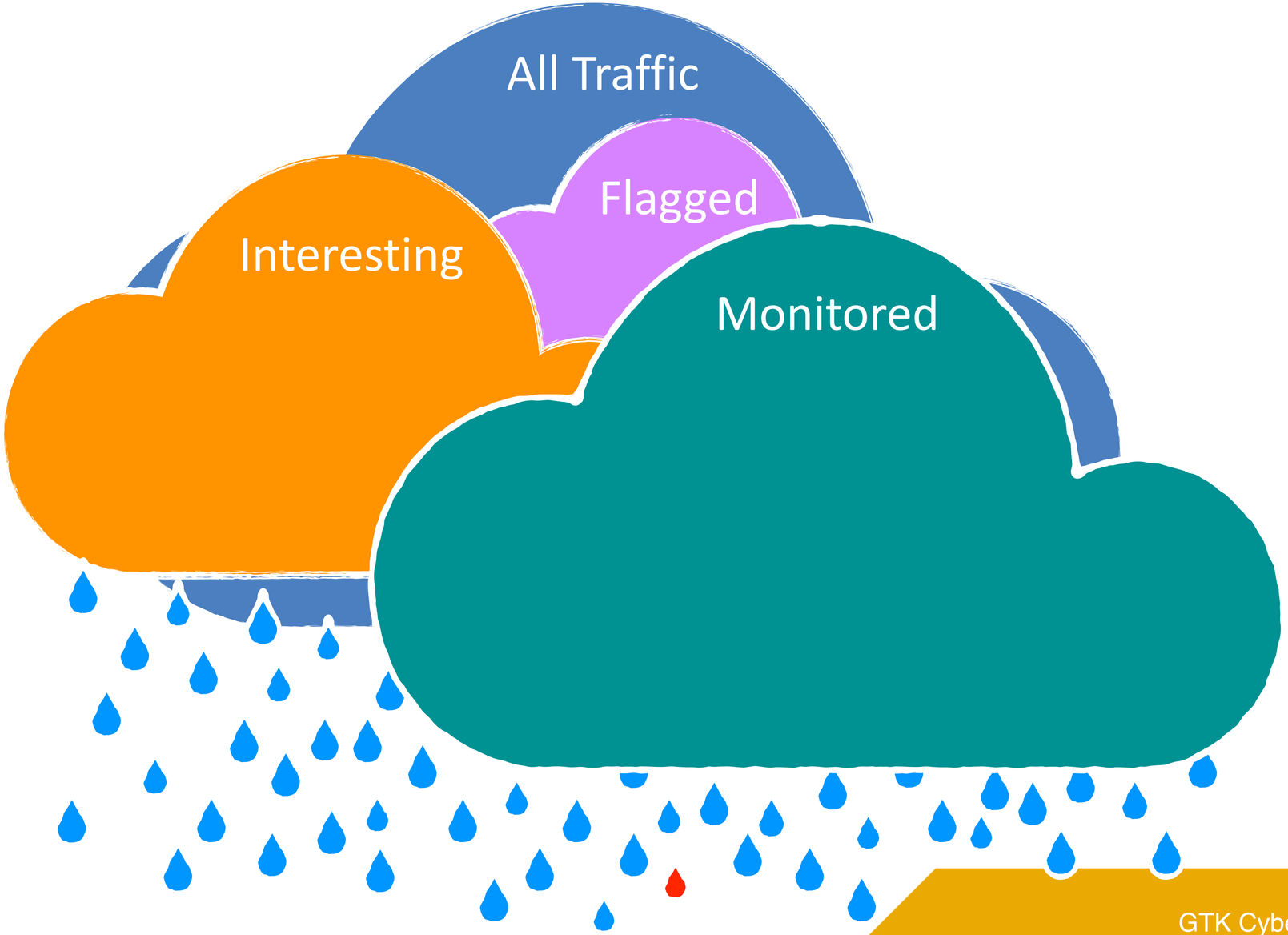
Data Science Hunting Funnel



@HuntOperator

http://www.austintaylor.io/network/traffic/threat/data/science/hunting/funnel/machine/learning/domain/expertise/2017/07/11/data-science-hunting-funnel/

Data Science Hunting



labs.gtkcyber.com

Username: student_0x

Join the Slack Channel at: http://bit.ly/30IvONs