



# Applied Data Science for Cyber Security

Presented by Charles Givre, CISSP

[charles.givre@gtkcyber.com](mailto:charles.givre@gtkcyber.com)

Join the Slack Channel at: <https://gtkcyberstudents.slack.com>

# Course Agenda

- Introduction: What is Data Science?
  - Overview of Machine Learning & Cyber Applications
- Manipulating and Exploring Data
  - Exploratory Data Analysis in 1 Dimension
  - Exploratory Data Analysis in 2 Dimensions
- Data Visualization
- Introduction to Supervised Machine Learning
- Threat Hunting with Data Science

# 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

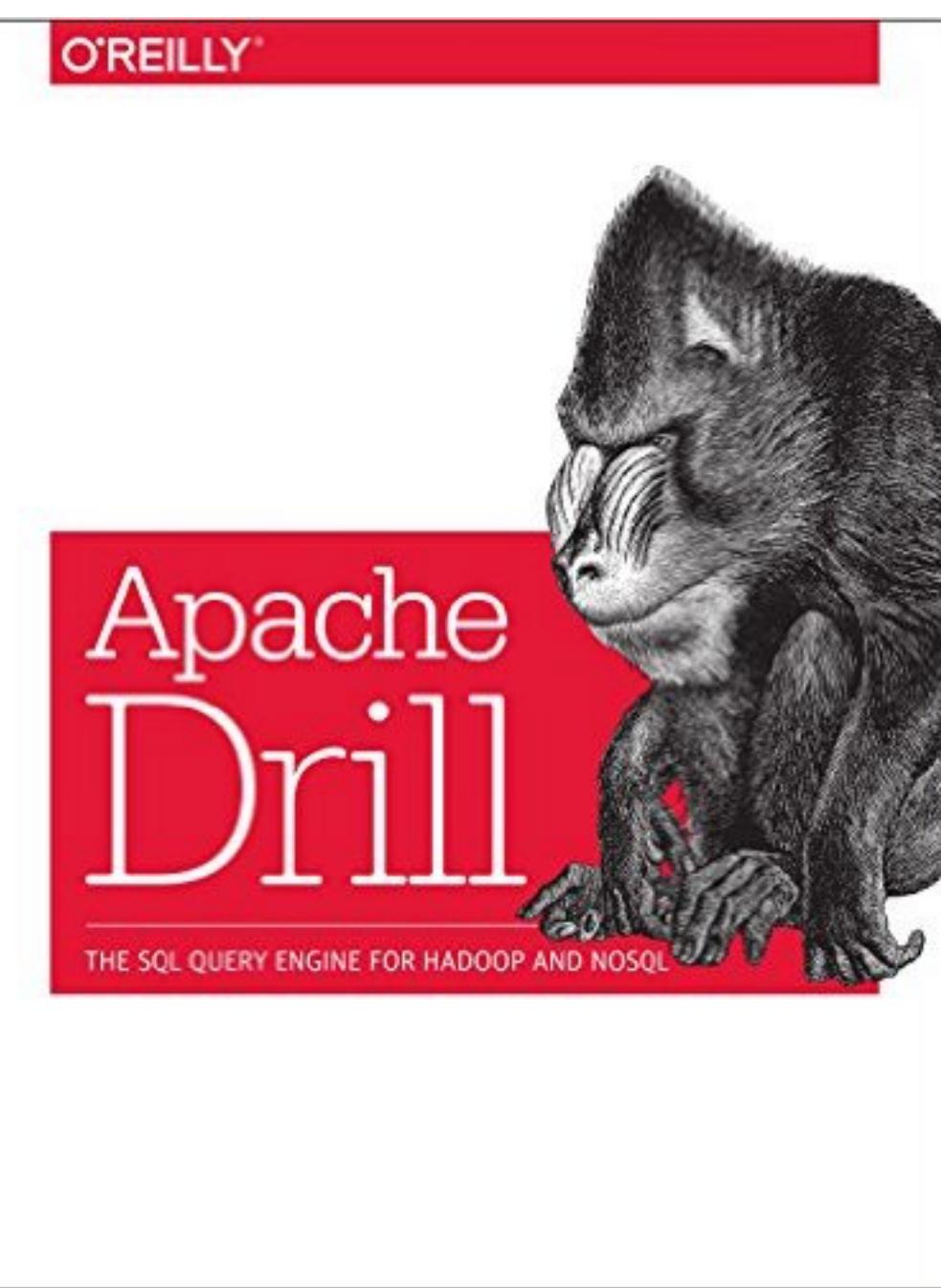


All materials presented in this training and those provided as an adjunct to the program are copyrighted 2019 by GTK Cyber LLC.

They are intended solely for the use of registered program participants and may not be reproduced or redistributed in any manner for any other reason.

# Charles Givre, CISSP

- Lead Data Scientist at Deutsche Bank
- Program Chair Strategic Analytics, Brandeis University
- Committer to Apache Drill Project
- Senior Lead Data Scientist @ Booz Allen
- 5 Years @ CIA
- Undergraduate in Comp.Sci & Music



# **What is Data Science?**

**Data Science is the automated  
extraction of information from raw data.**

**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**

Booz Allen Hamilton, Field Guide to Data Science, Pg. 17

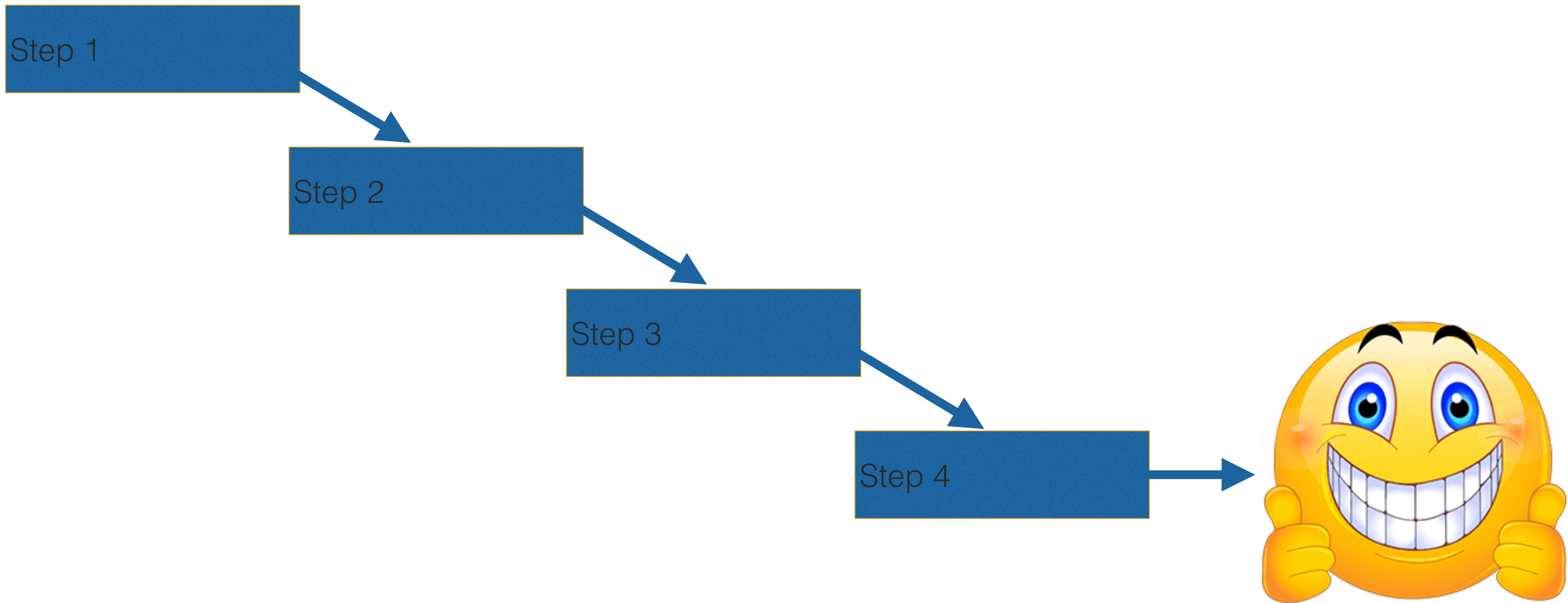
Analyst



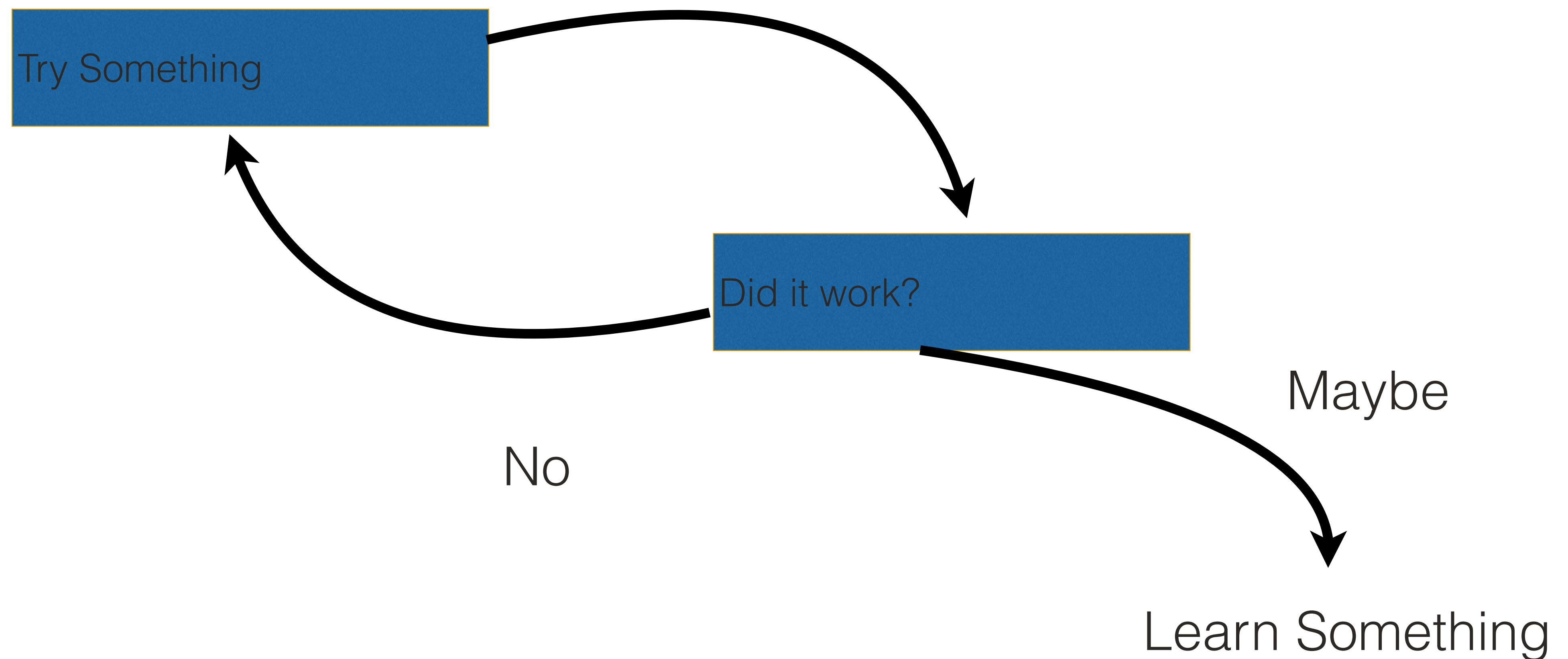
Developer

Analyst + Developer

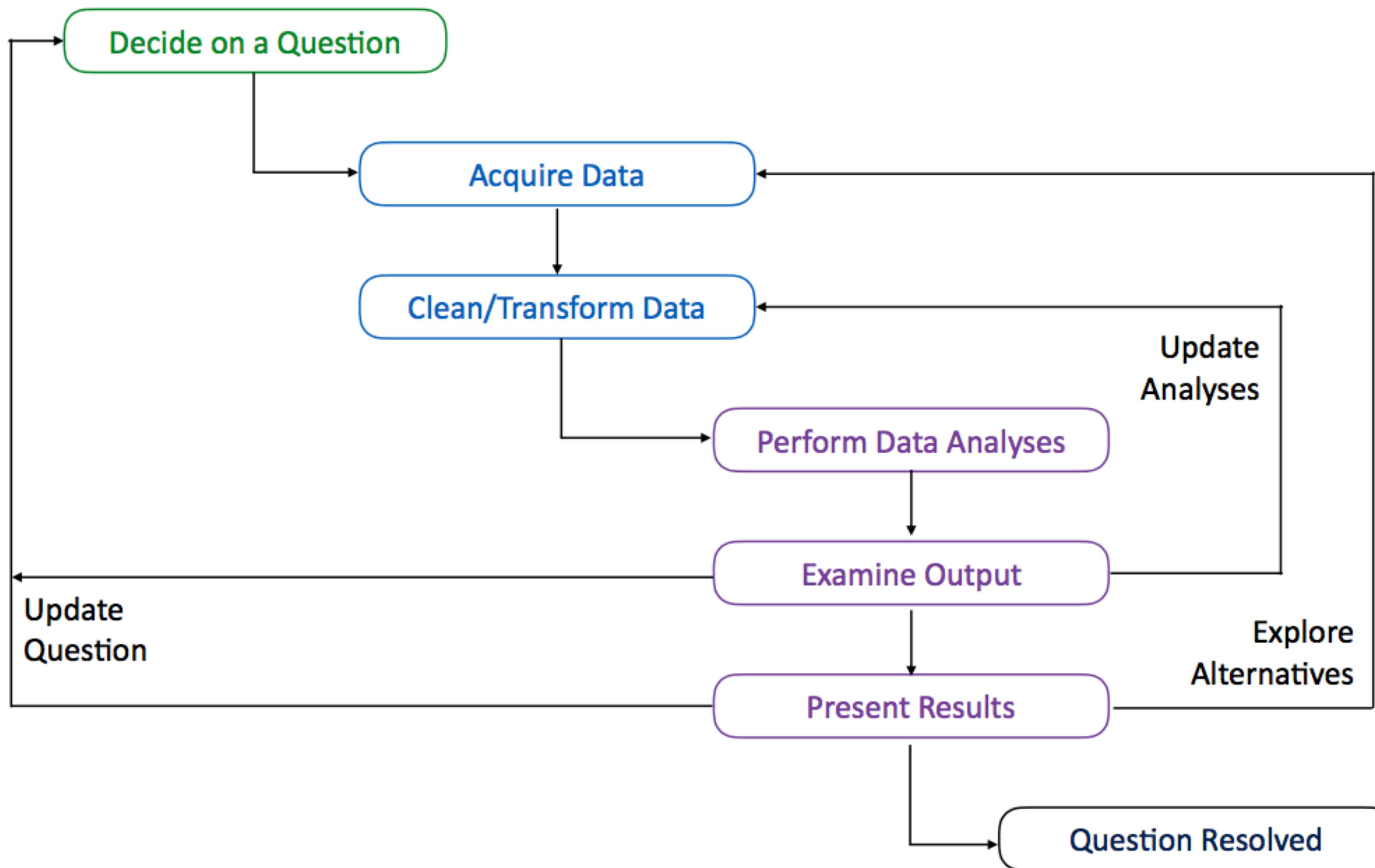
# What Data Science is Not



# What Data Science is

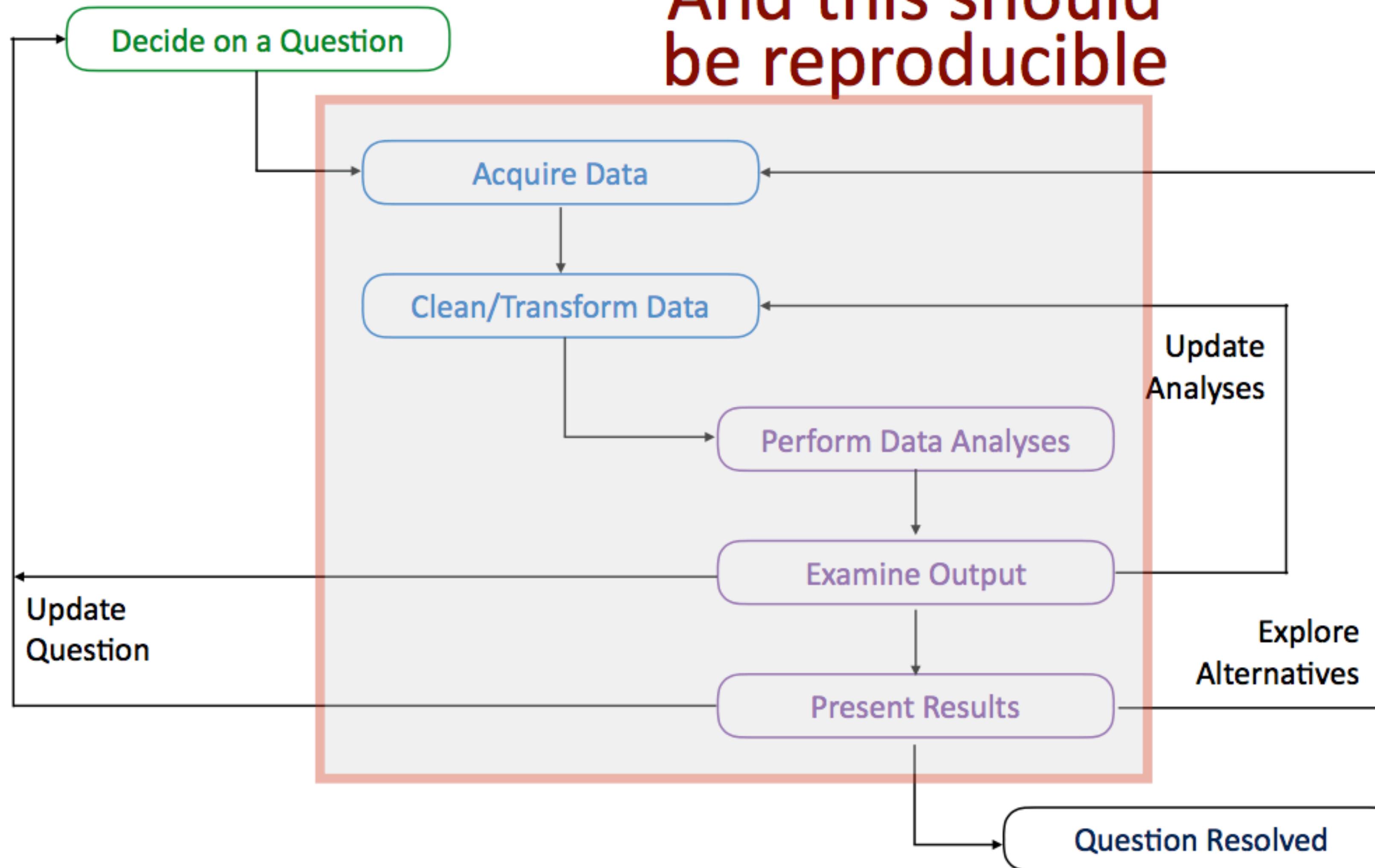


# Research Process



# Research Process

And this should  
be reproducible



"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.**

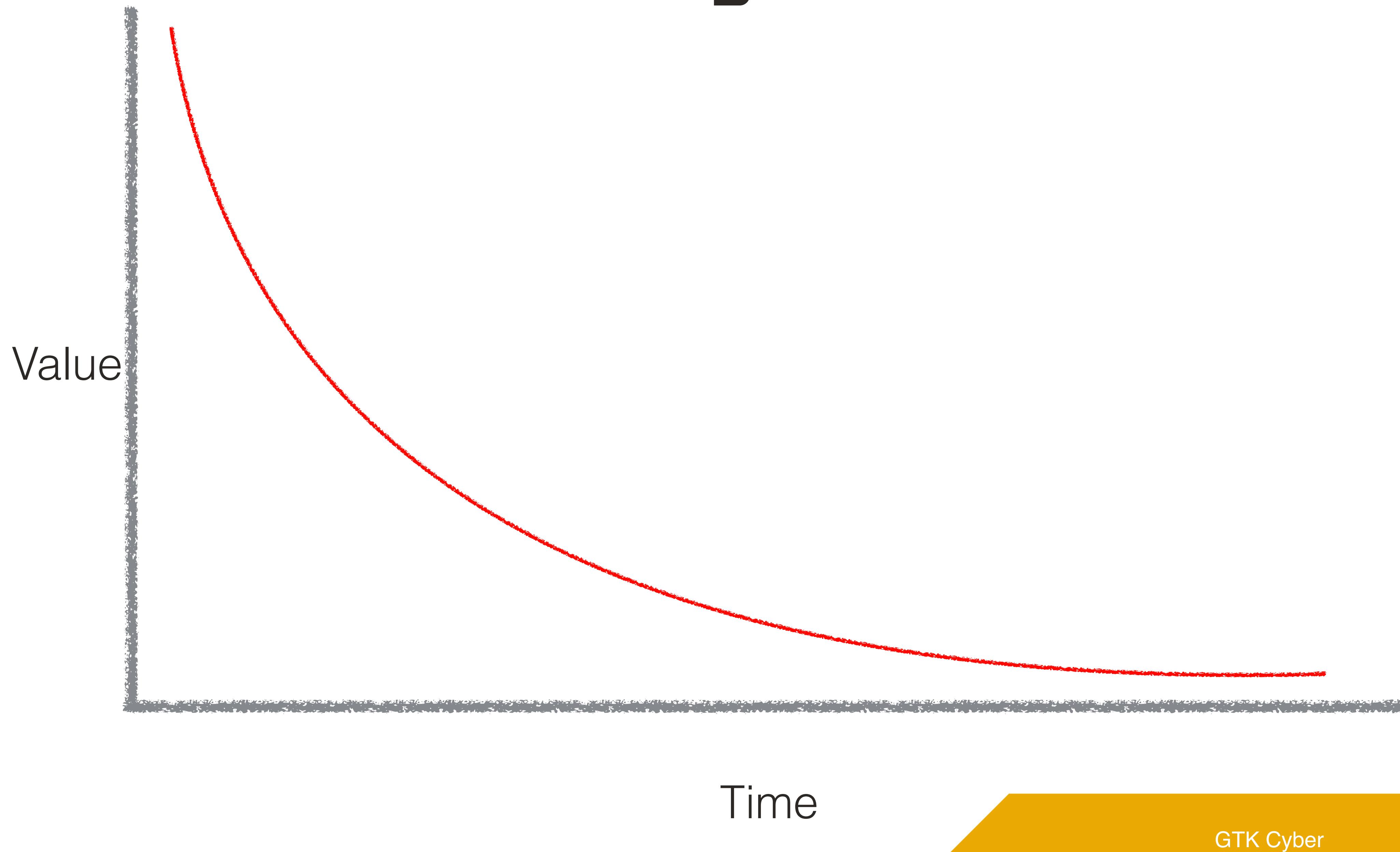
–Andrew Brust, “Data scientists don’t scale”, <http://www.zdnet.com/article/data-scientists-dont-scale/>

# Time to Insight

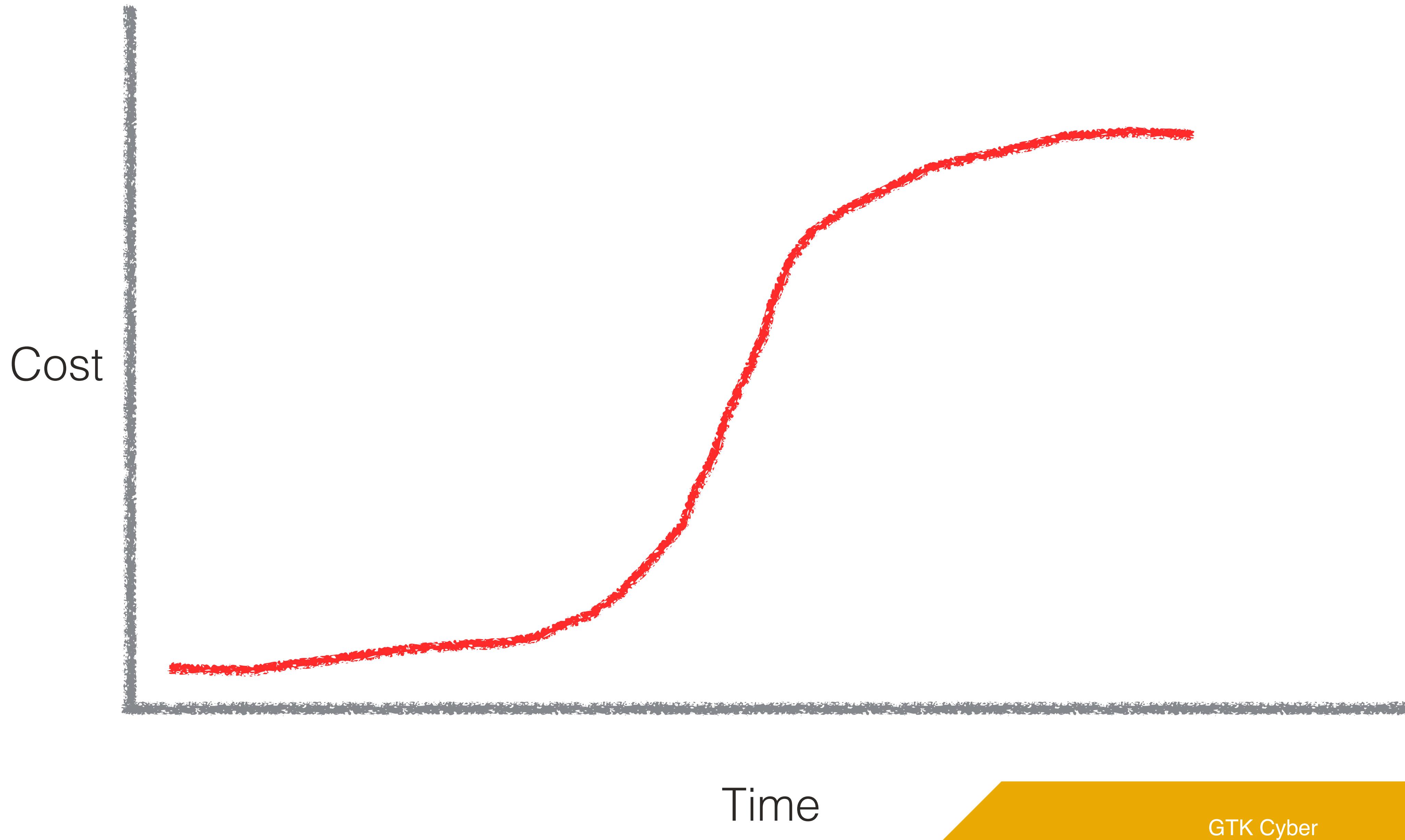
# Time to Insight

Time = **\$\$**

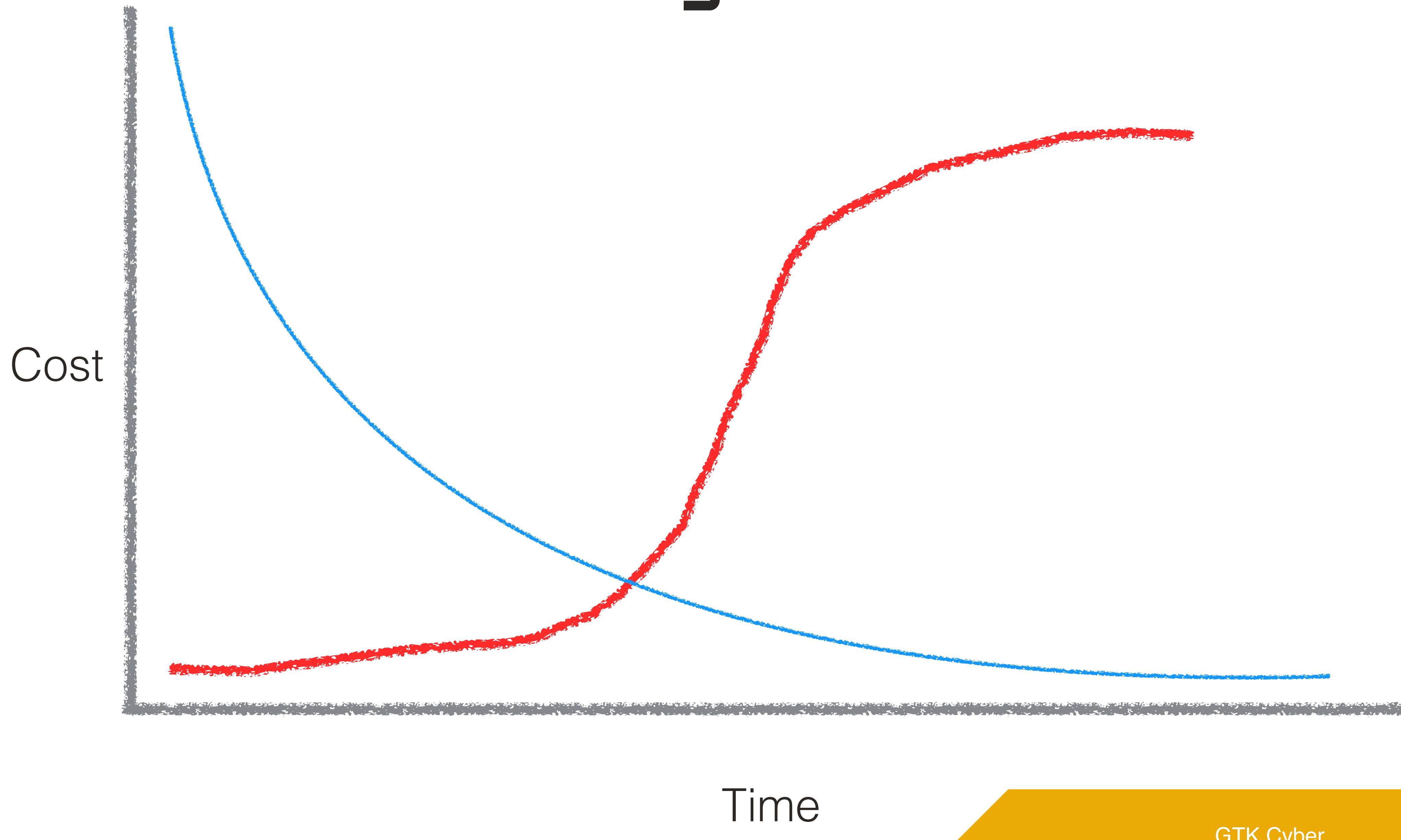
# Value of Insights over Time



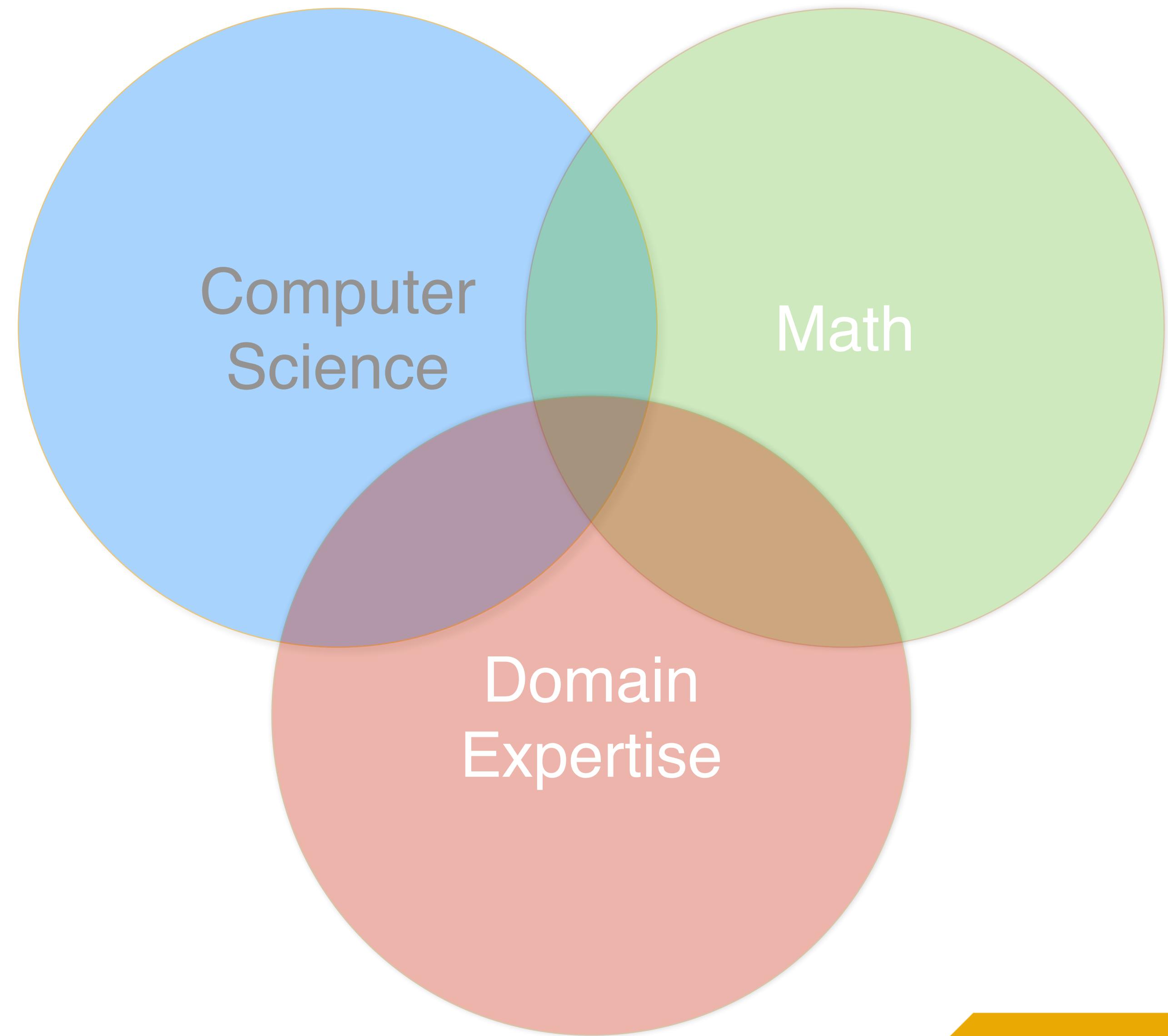
# Cost of Insights over Time

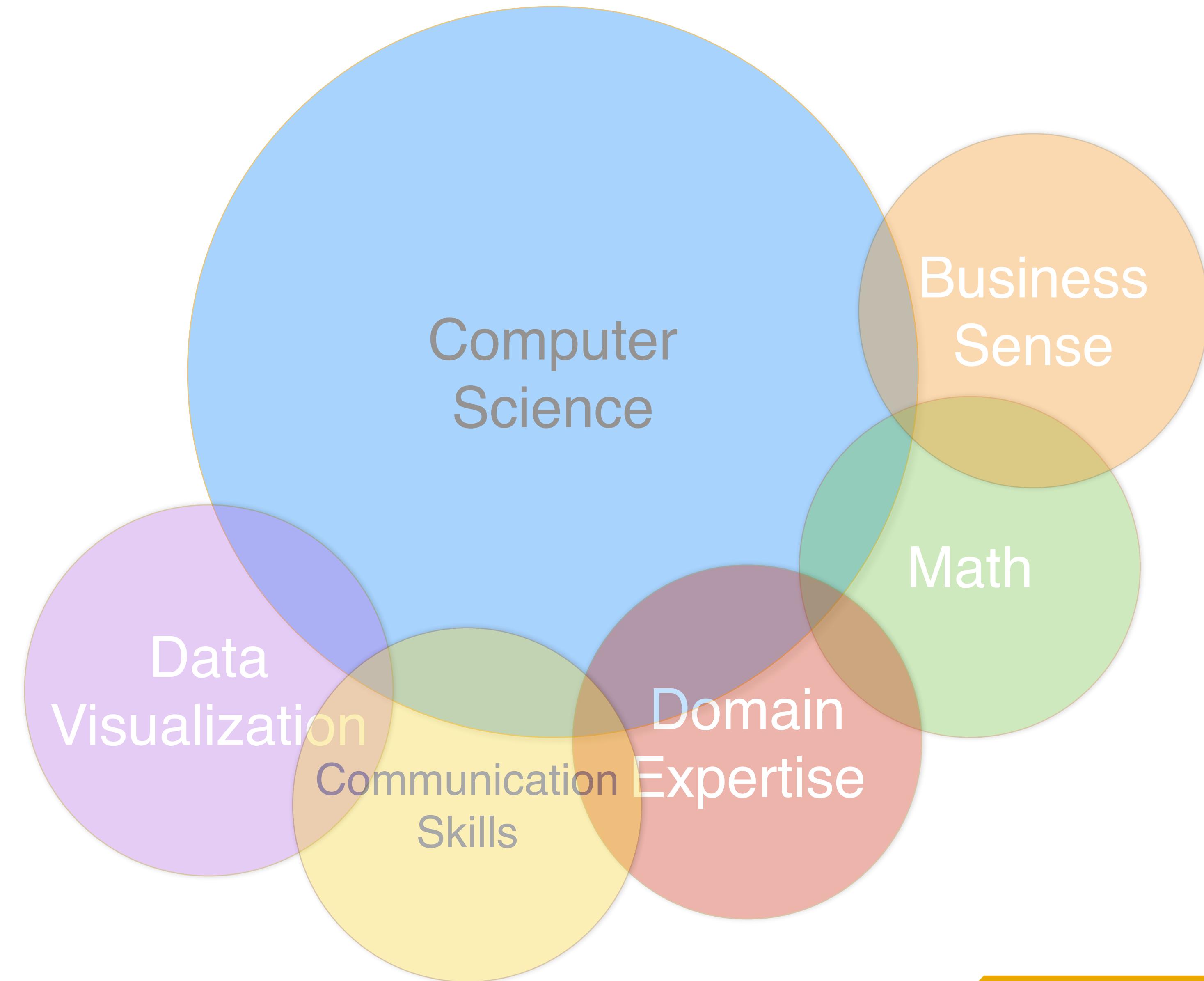


# Cost of Insights over Time

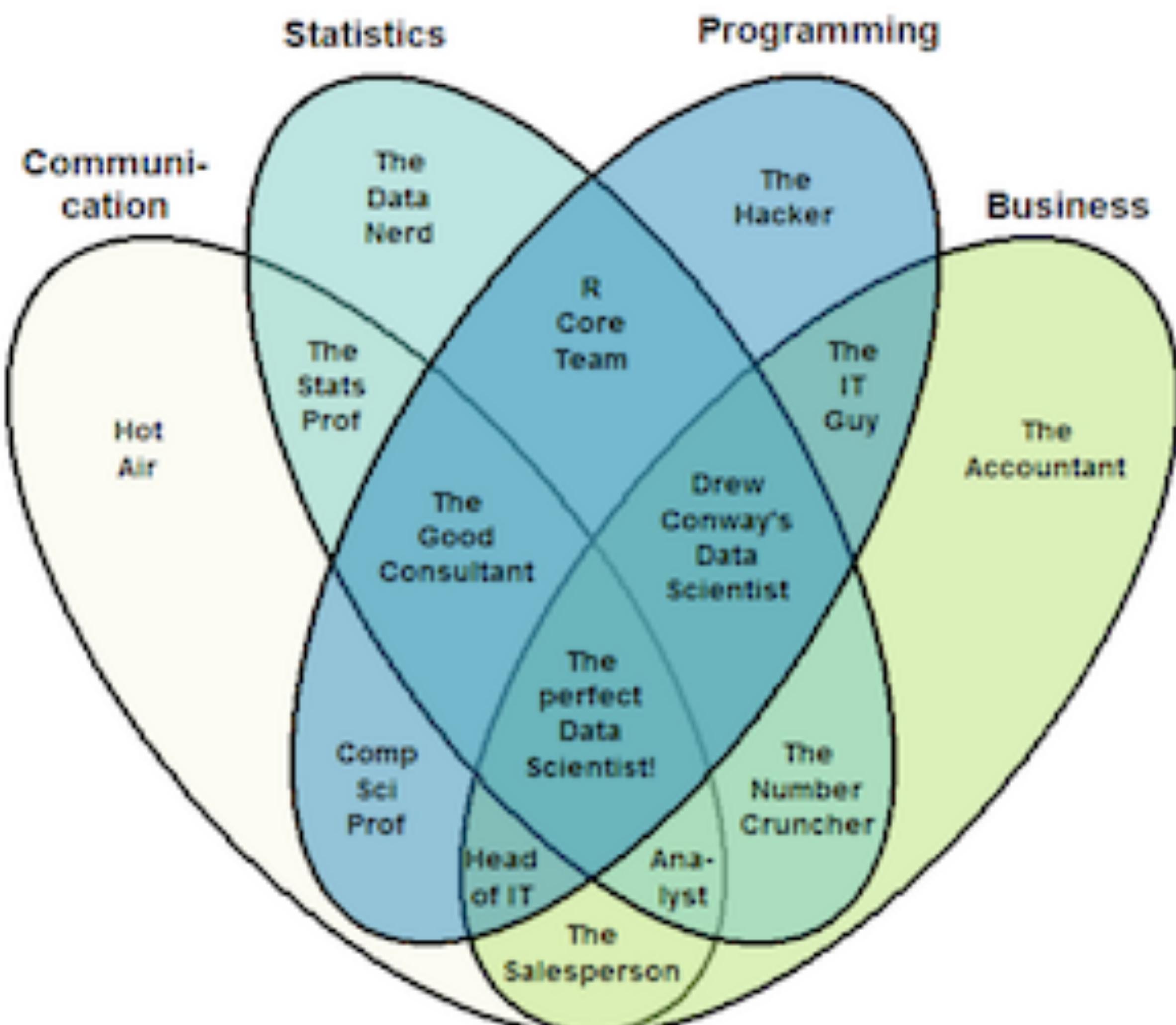


# **What skills does a data scientist need?**





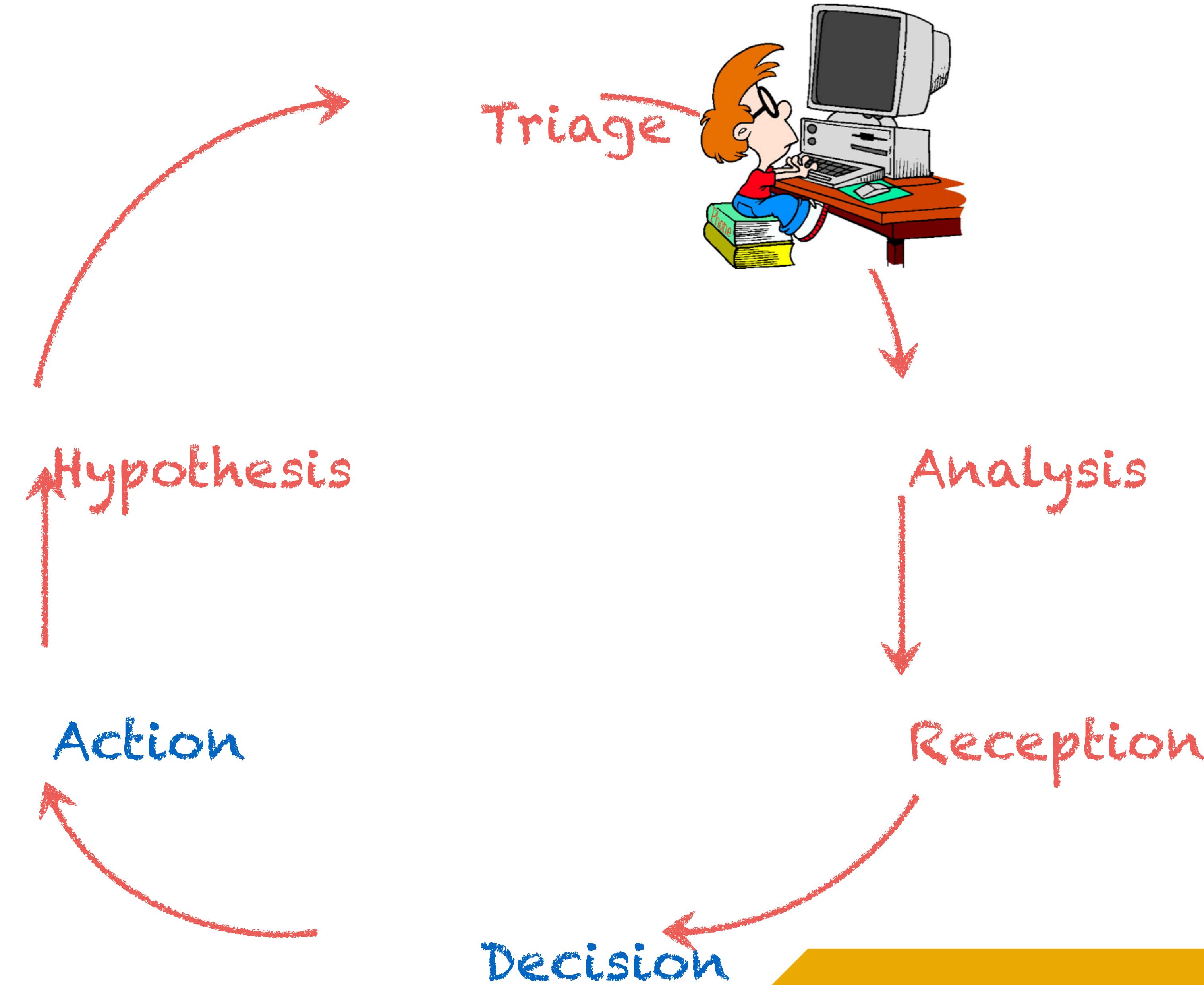
## The Data Scientist Venn Diagram



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

# Data Janitors





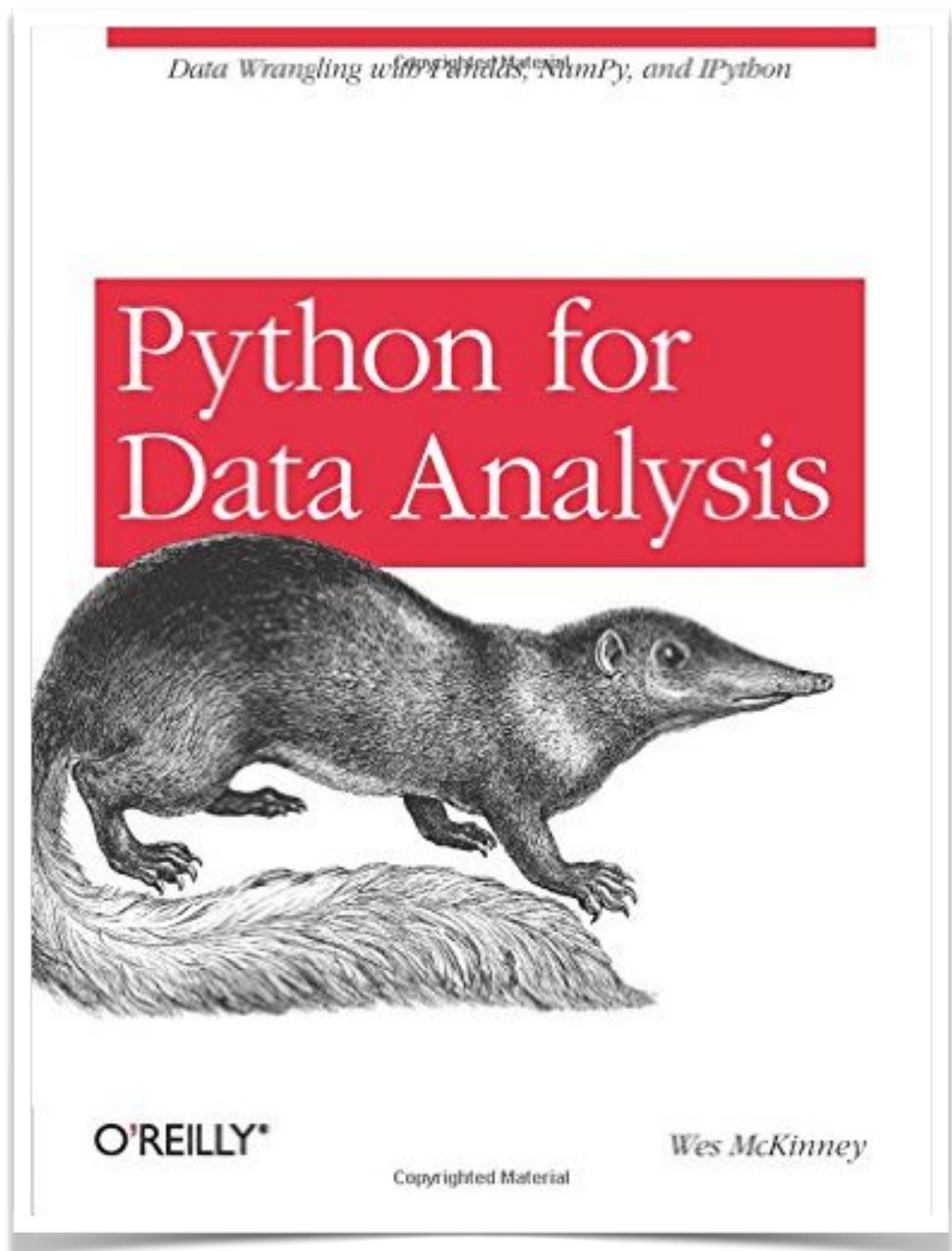
# **By the end of the class you will be able to:**

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

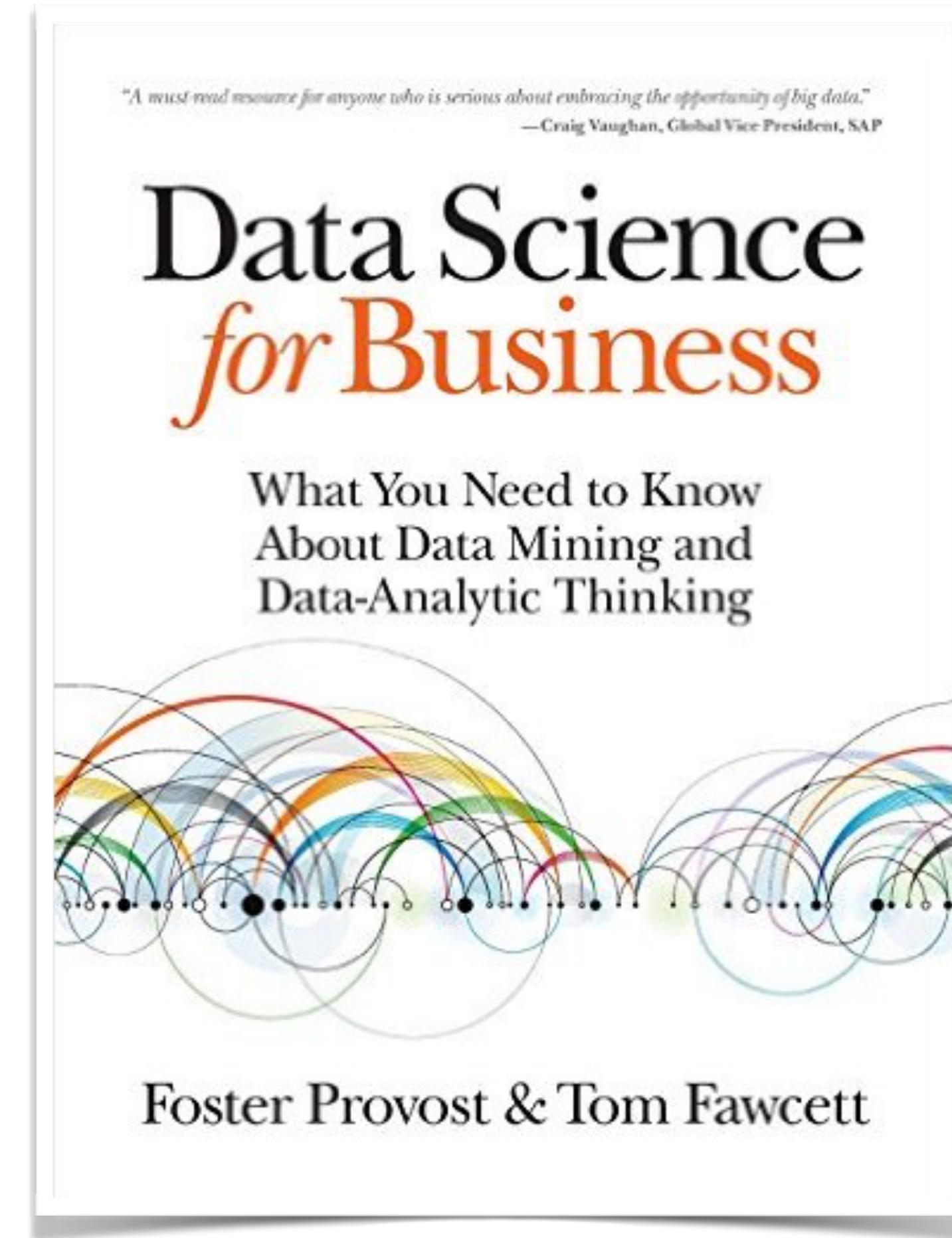




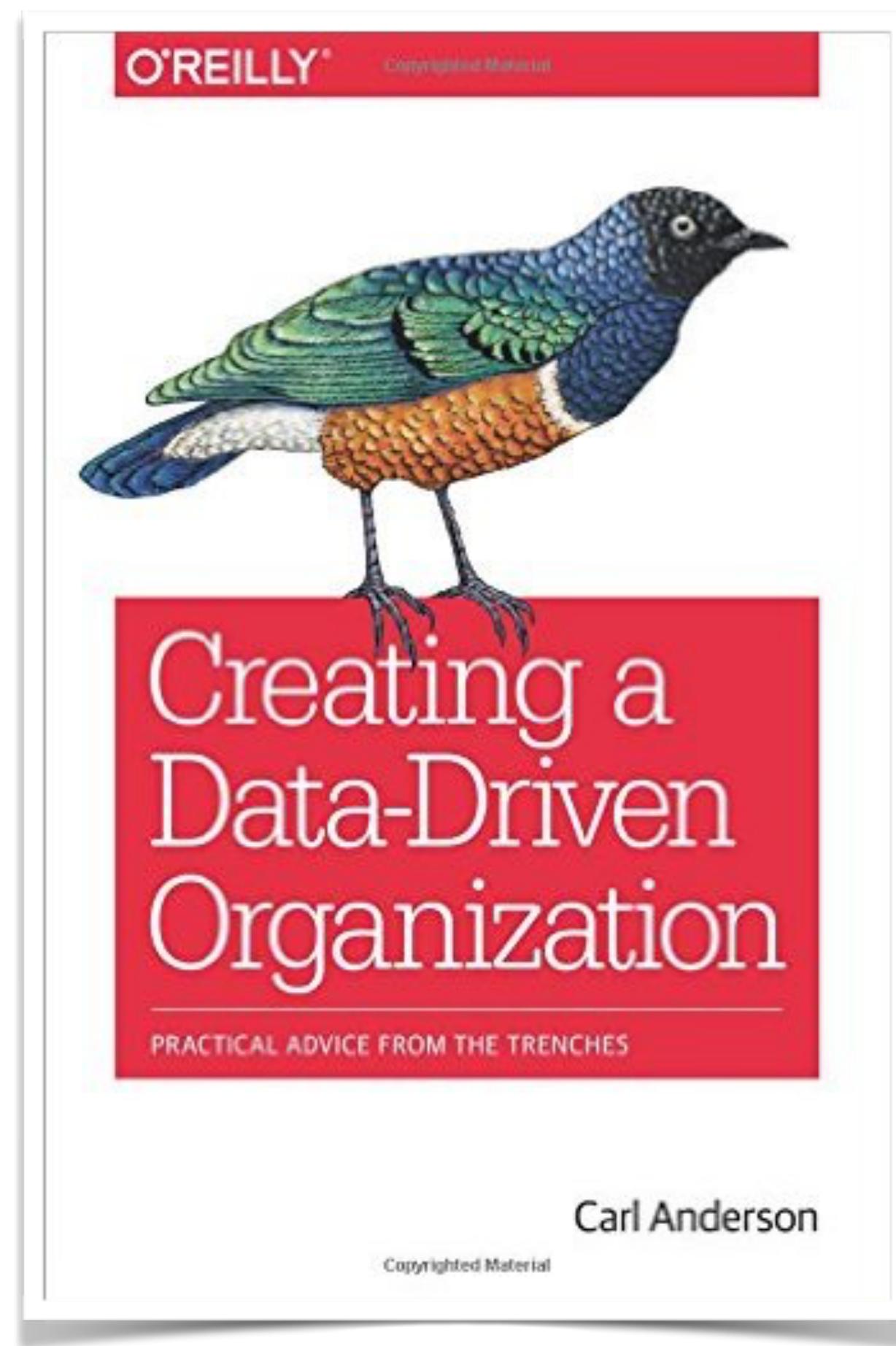
# Recommended Reading



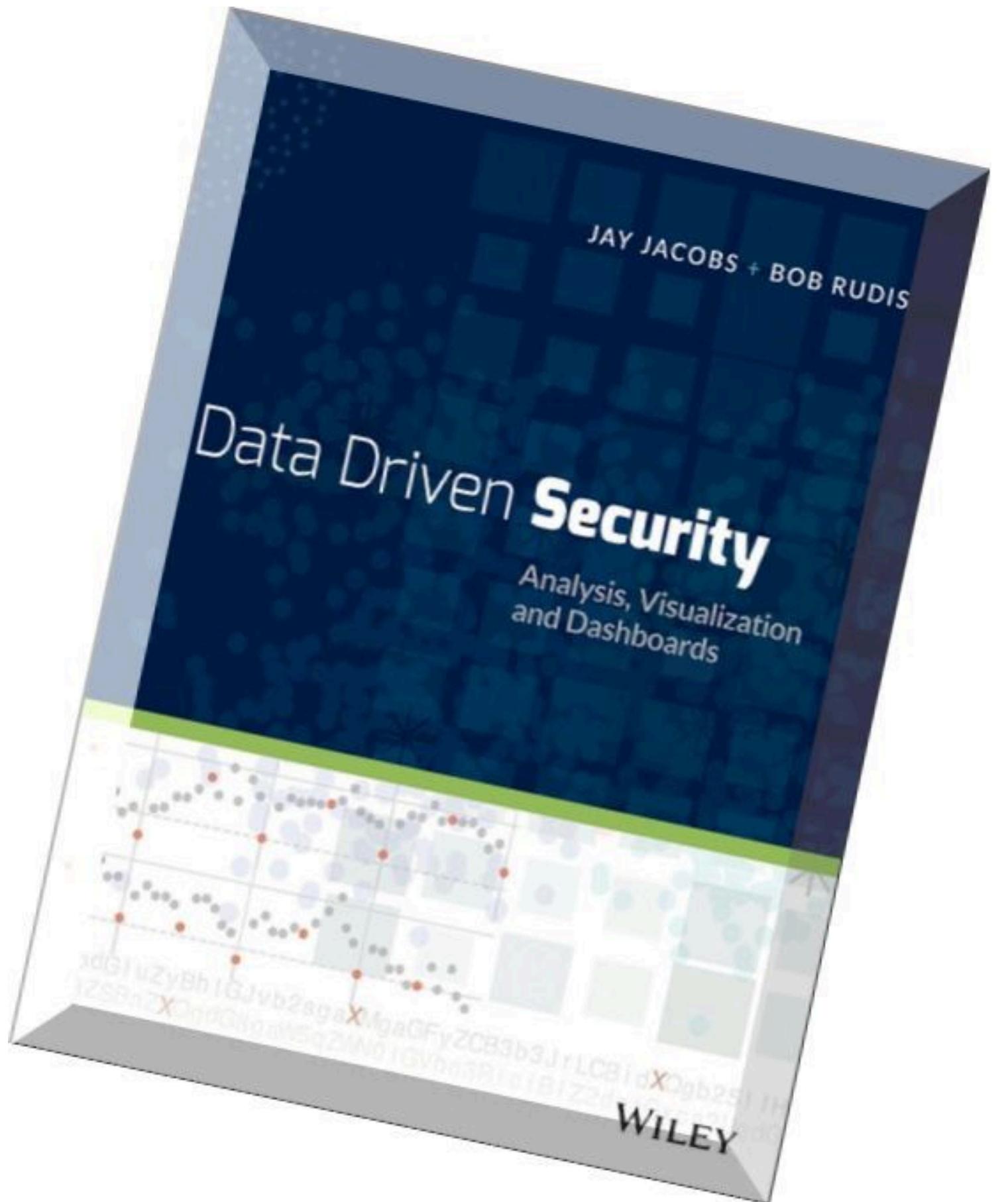
# Recommended Reading



# Recommended Reading

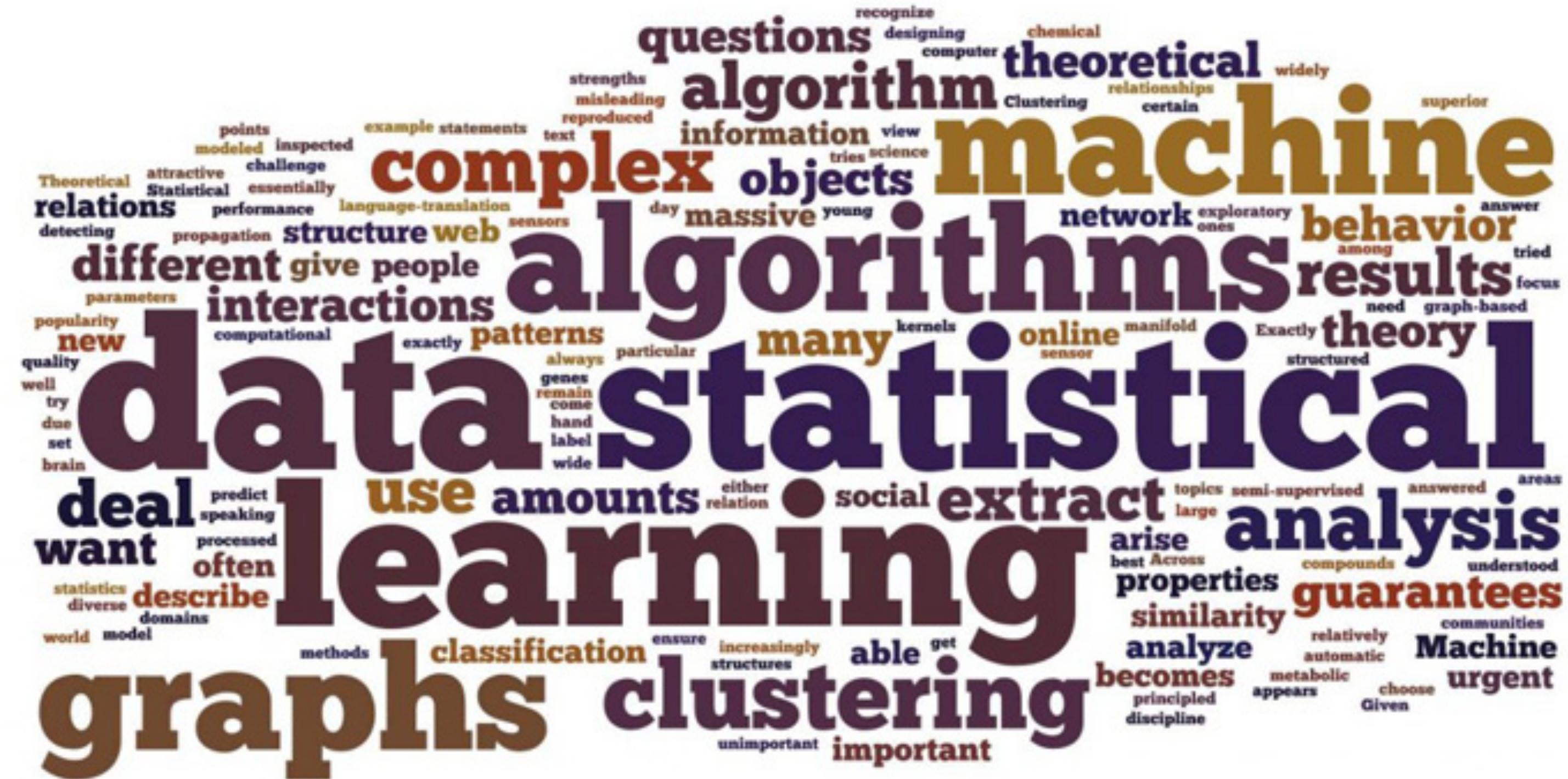


# Recommended Reading



<http://datadrivensecurity.info>

**What is  
Machine Learning (ML)  
Artificial Intelligence (AI)?**



“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](https://en.wikipedia.org/wiki/Machine_learning)



GTK Cyber





- Blacklists
- Simple keyword matching
- Naive Bayesian Classifiers
- Deep Learning

## Artificial Intelligence

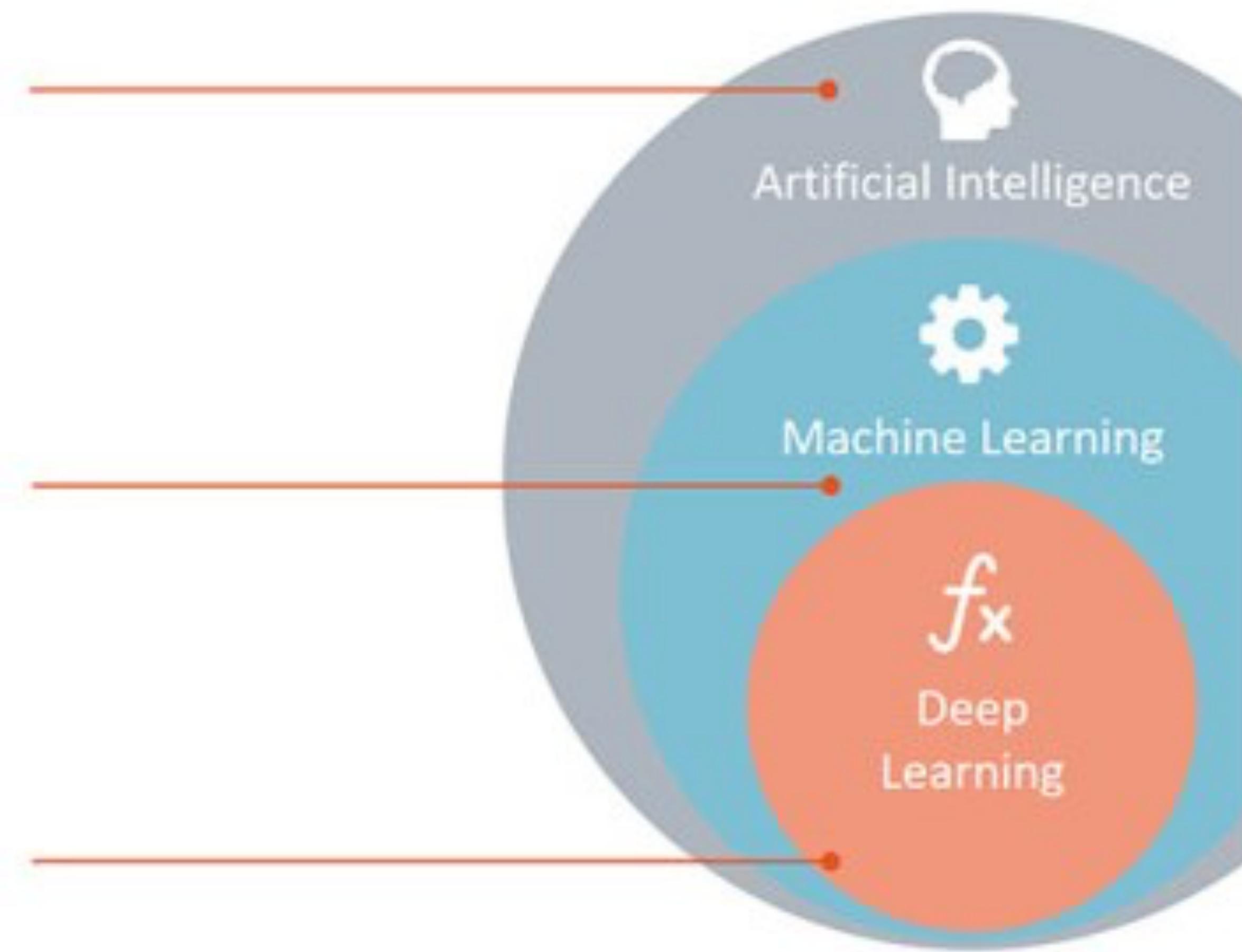
Any technique which enables computers to mimic human behavior.

## 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.



@katherinebailey Because marketing? Every time someone calls simple linear regression “AI” Gauss turns over in his grave.

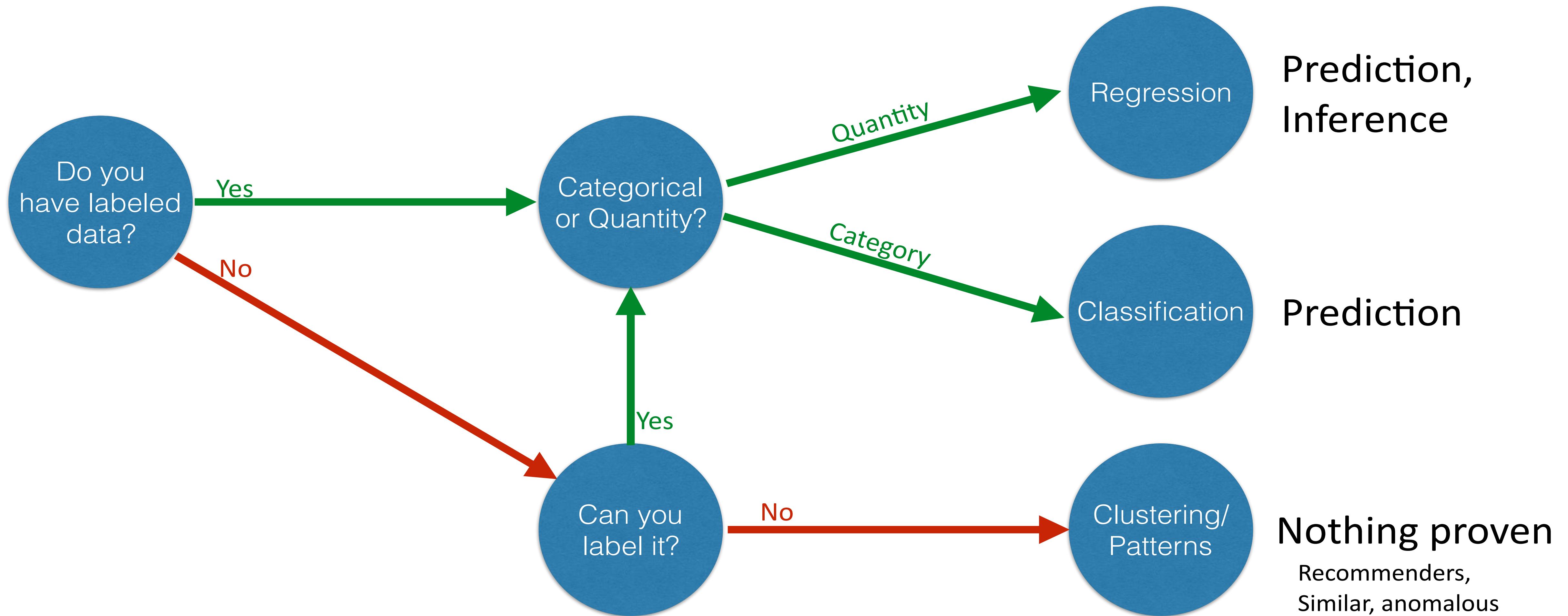
# Machine Learning Problems

- **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.

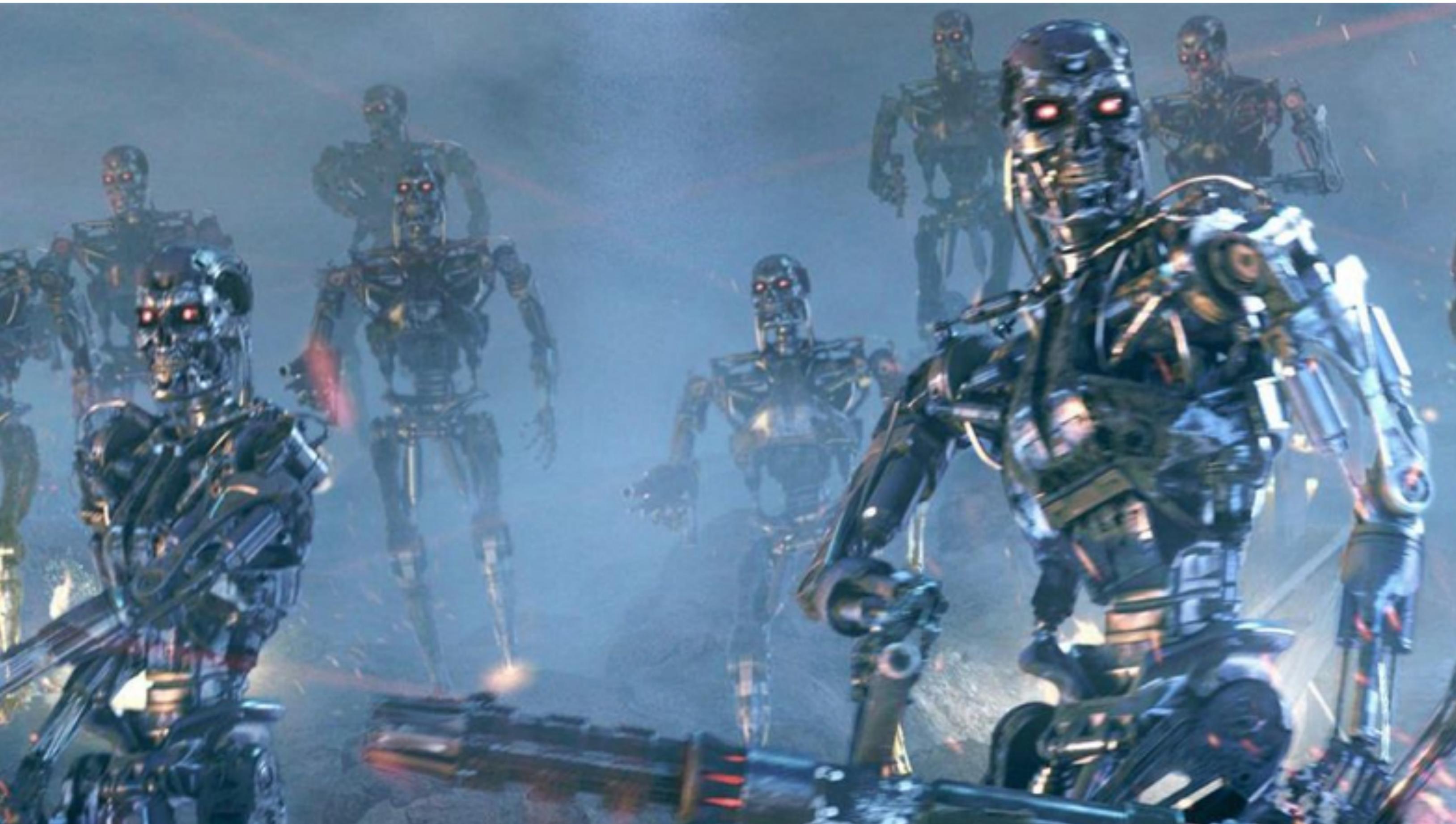
# Machine Learning Problem Types

- **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

# What Problem am I solving?

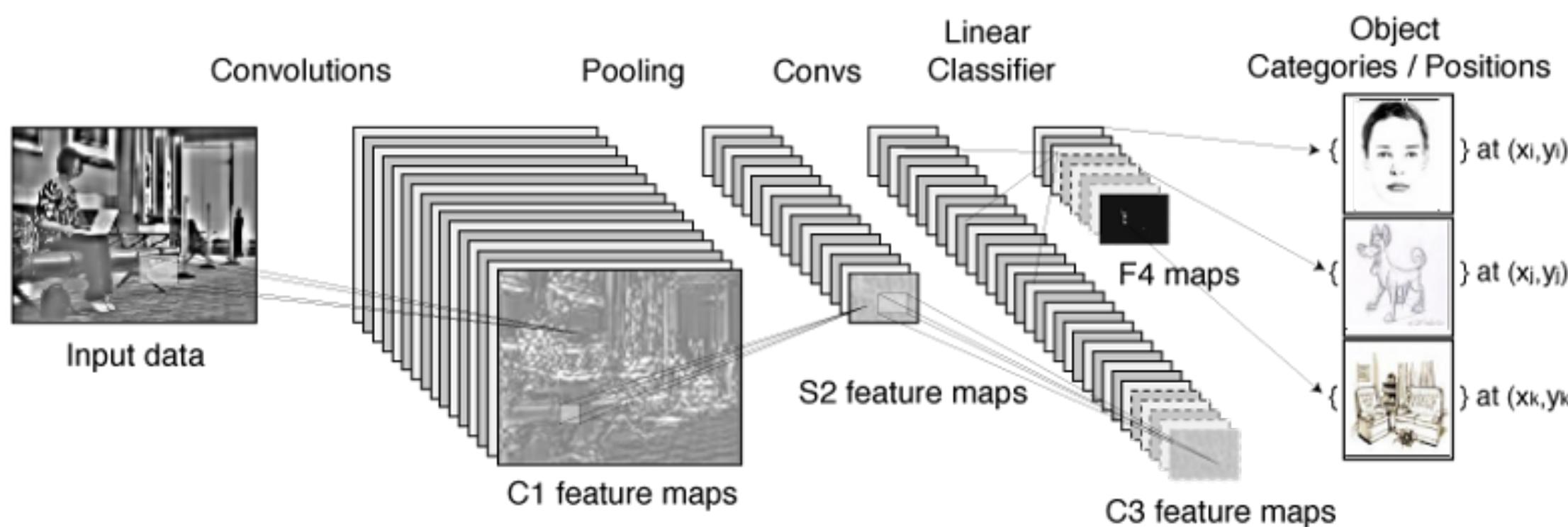


# What it is not



# Path to Artificial Intelligence?

- Hard-coding knowledge failed (e.g. Cyc 1989).
- **Machine Learning: Gathering knowledge from experience** eliminates the need to formally specify all knowledge the computer needs.
- **AI** needs to acquire own knowledge by **extracting pattern from raw data**.
- Success depends heavily on **representation of data, aka features**.



# **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.

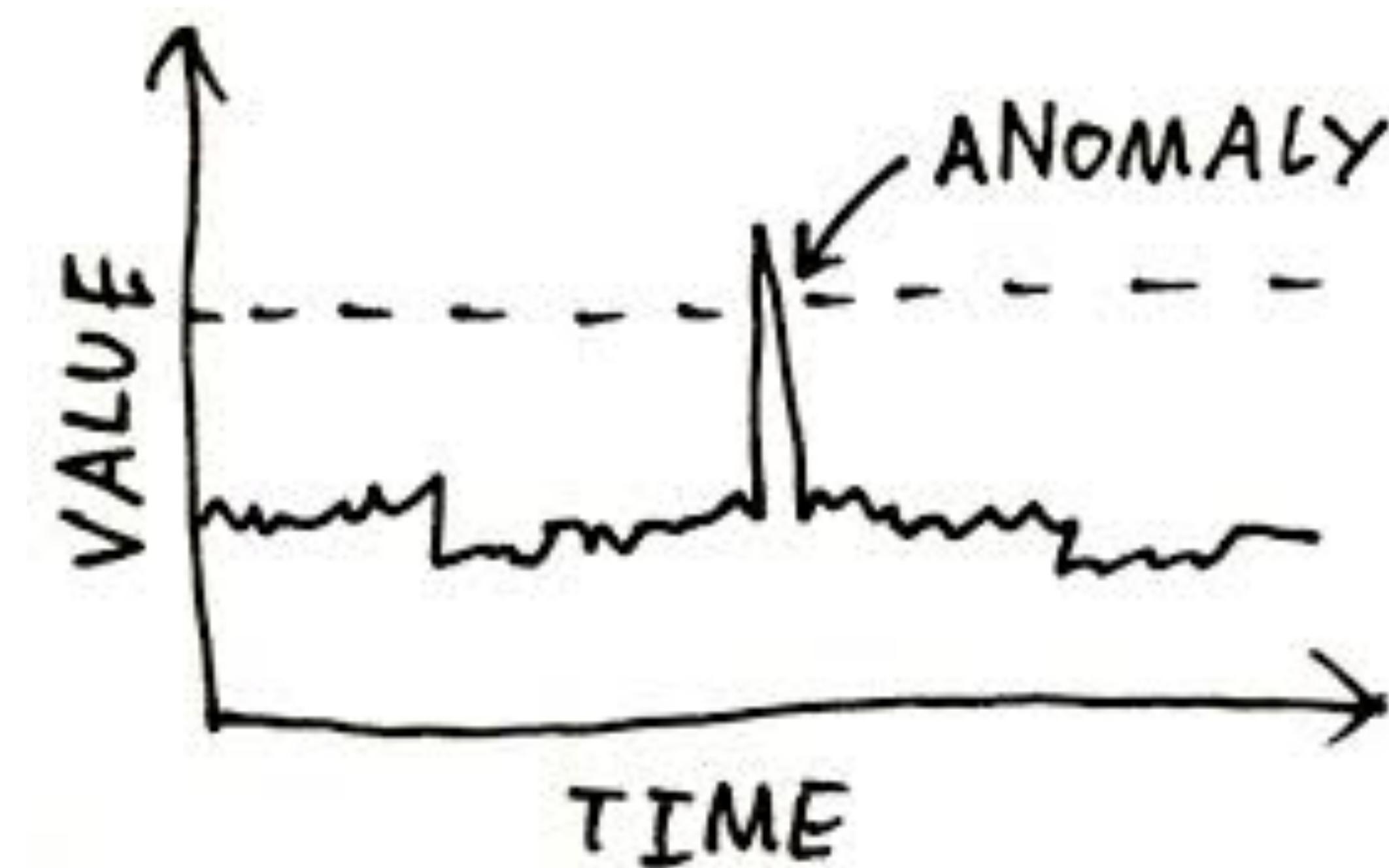


[https://www.researchgate.net/publication/256645877\\_LiRCUP\\_Linear\\_Regression\\_based\\_CPU\\_Usage\\_Prediction\\_Algorithm\\_for\\_Live\\_Migration\\_of\\_Virtual\\_Machines\\_in\\_Data\\_Centers](https://www.researchgate.net/publication/256645877_LiRCUP_Linear_Regression_based_CPU_Usage_Prediction_Algorithm_for_Live_Migration_of_Virtual_Machines_in_Data_Centers)

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

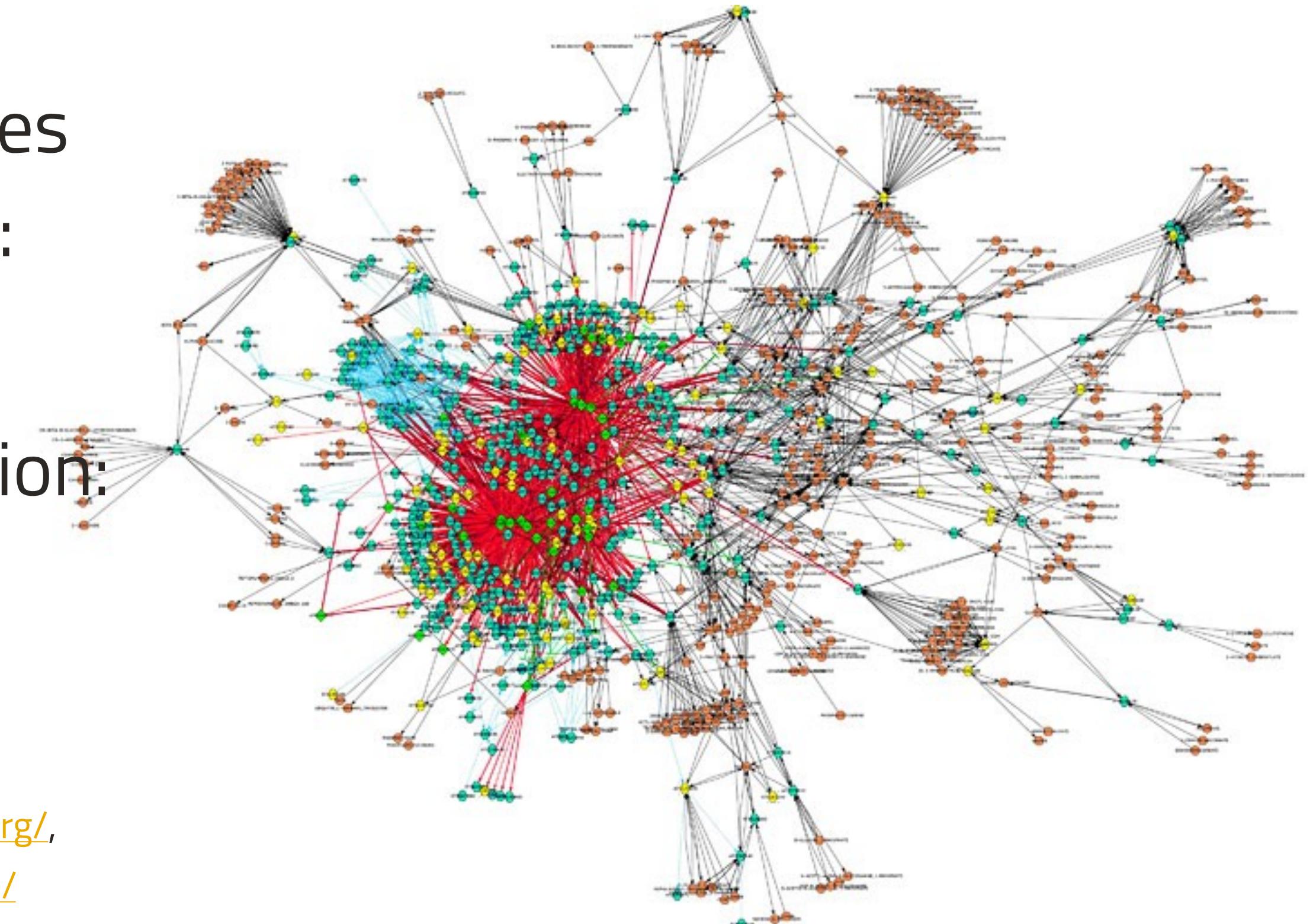
# Clustering Example

**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

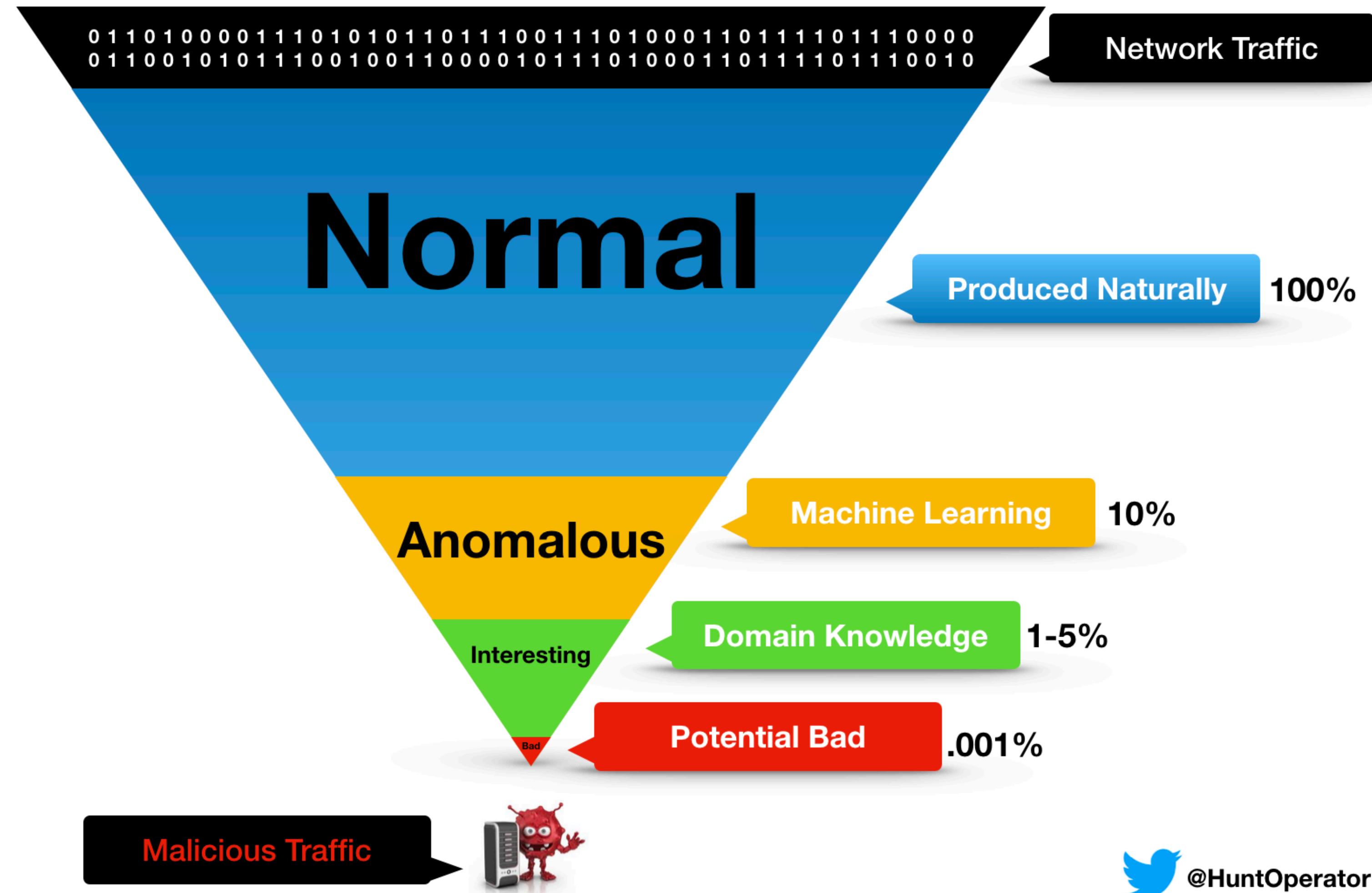
- Derive Features from Binary Content and metadata manifest (function calls, string obtained from IDA Disassembler)
- Example Features: opcode count (n-grams), segment count, asm pixel intensity, n-gramming of bytes, function name.
- Featureless Deep Learning with word2vec embedding
- Get open source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

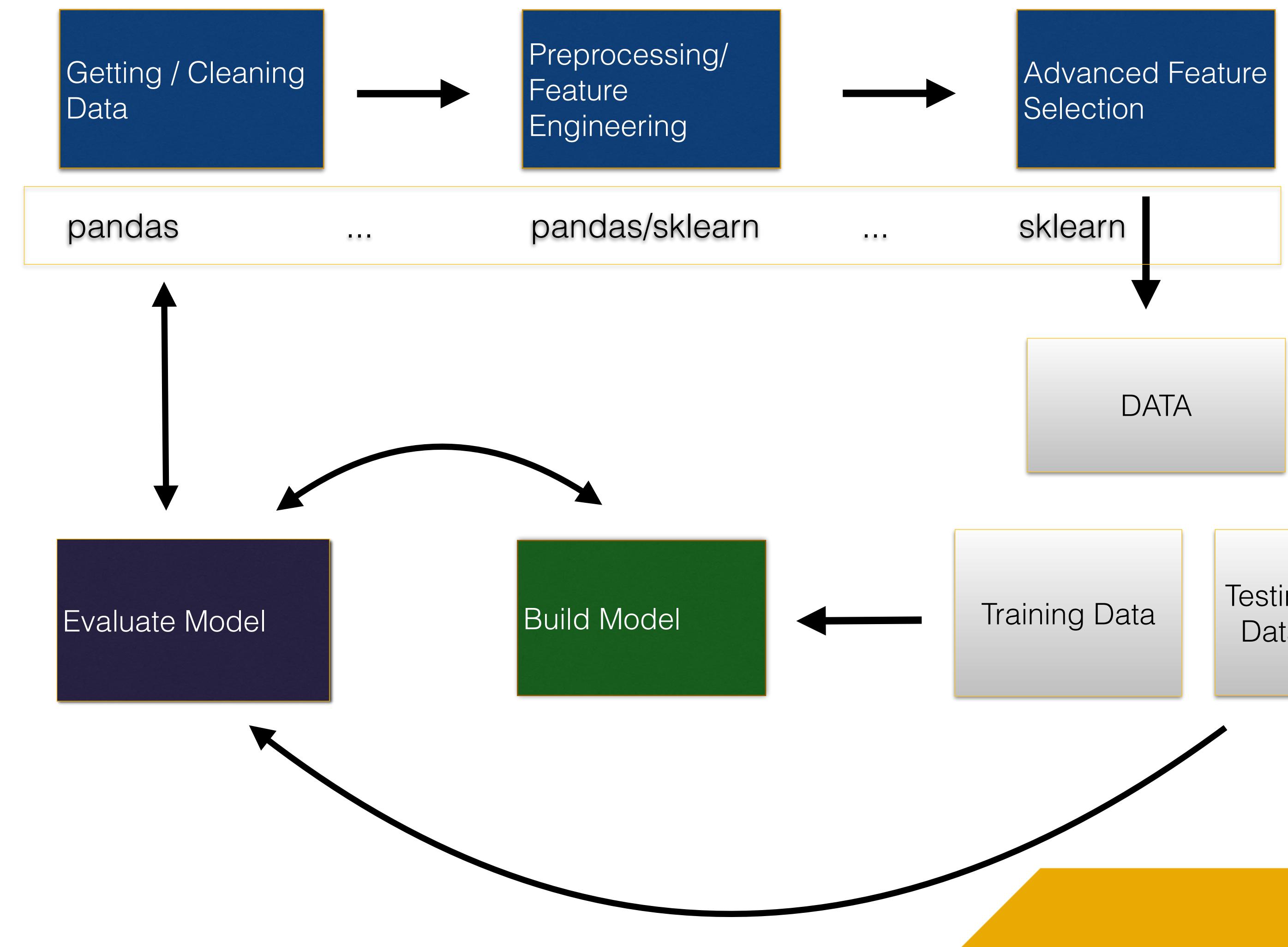
# Data Science Hunting Funnel



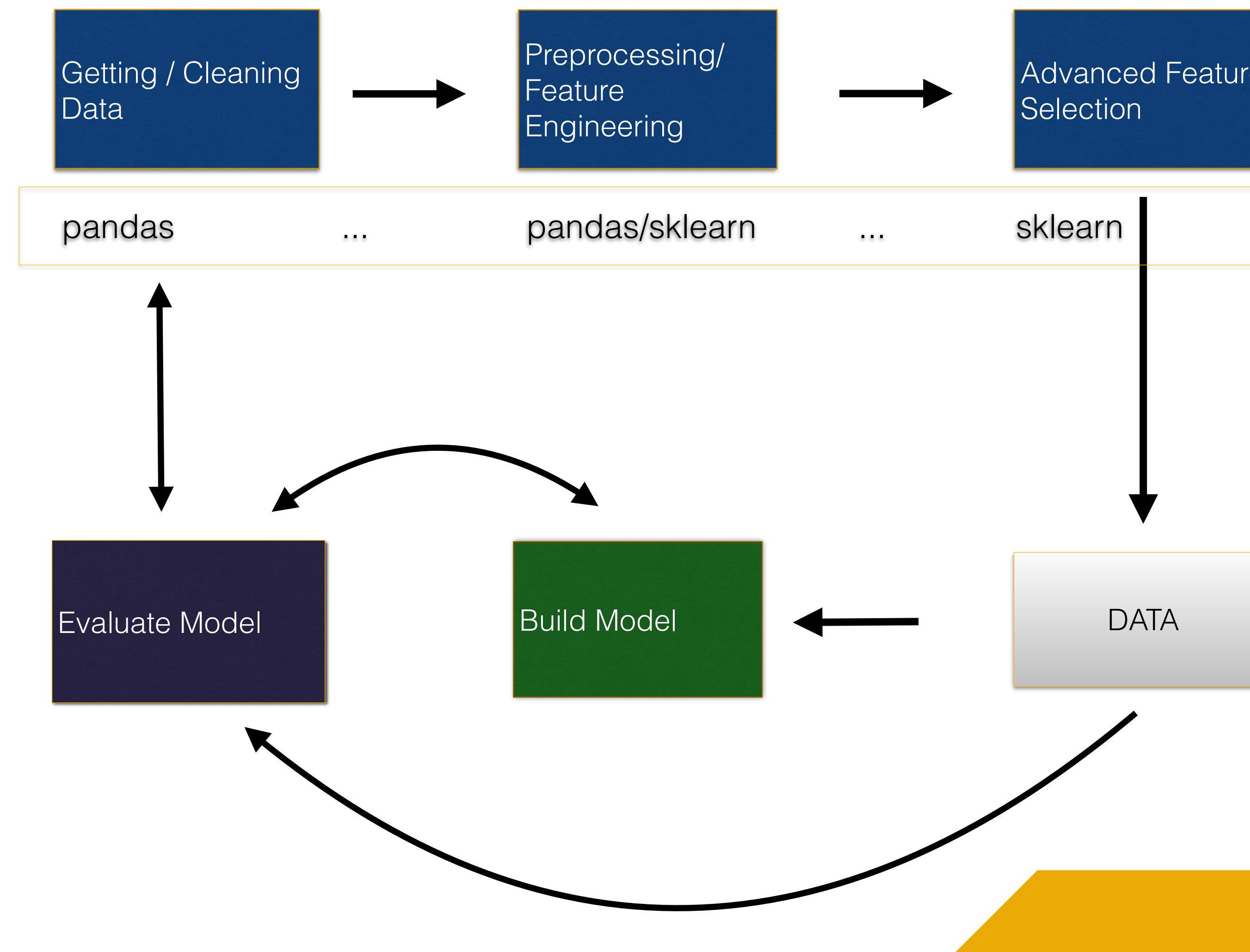
<http://www.austintaylor.io/network/traffic/threat/data/science/hunting/funnel/machine/learning/domain/expertise/2017/07/11/data-science-hunting-funnel/>

# The Machine Learning Process

# Supervised Machine Learning Process



# Unsupervised Machine Learning Process



**First, define your analytic  
question.**

**what are you trying to do?**

**How do you define success?  
What are you measuring?**

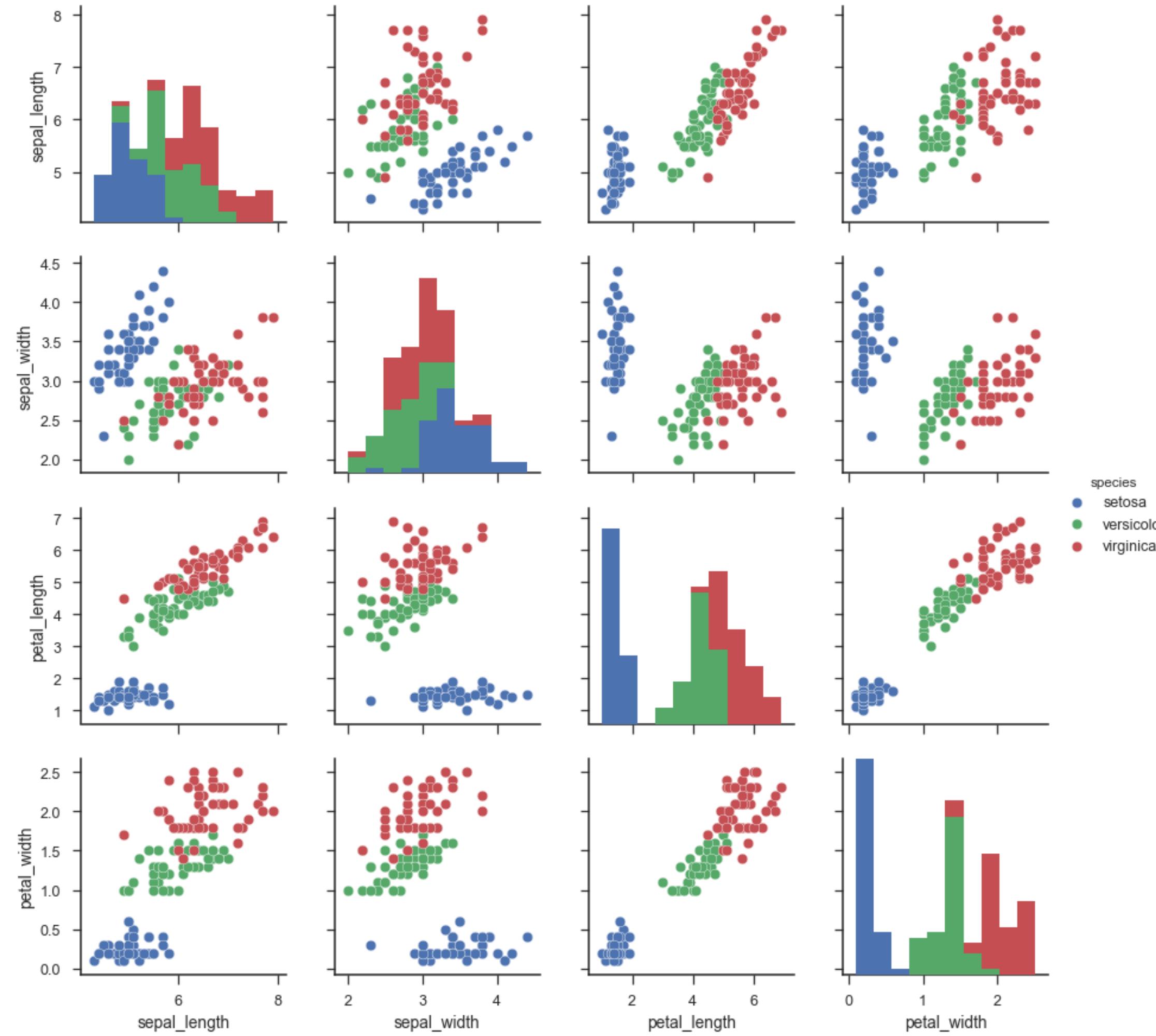
# Choose data sources

- What is available?
- Is it enough?
- Is the data reliable/clean/consistent?
- What other data could you use?

# Other Considerations

- Policies
- Legal constraints
- Biases in Data
- Latency
- Data size

# Gather and Explore Your Data



Is the data good enough?

What are the rules governing its use?

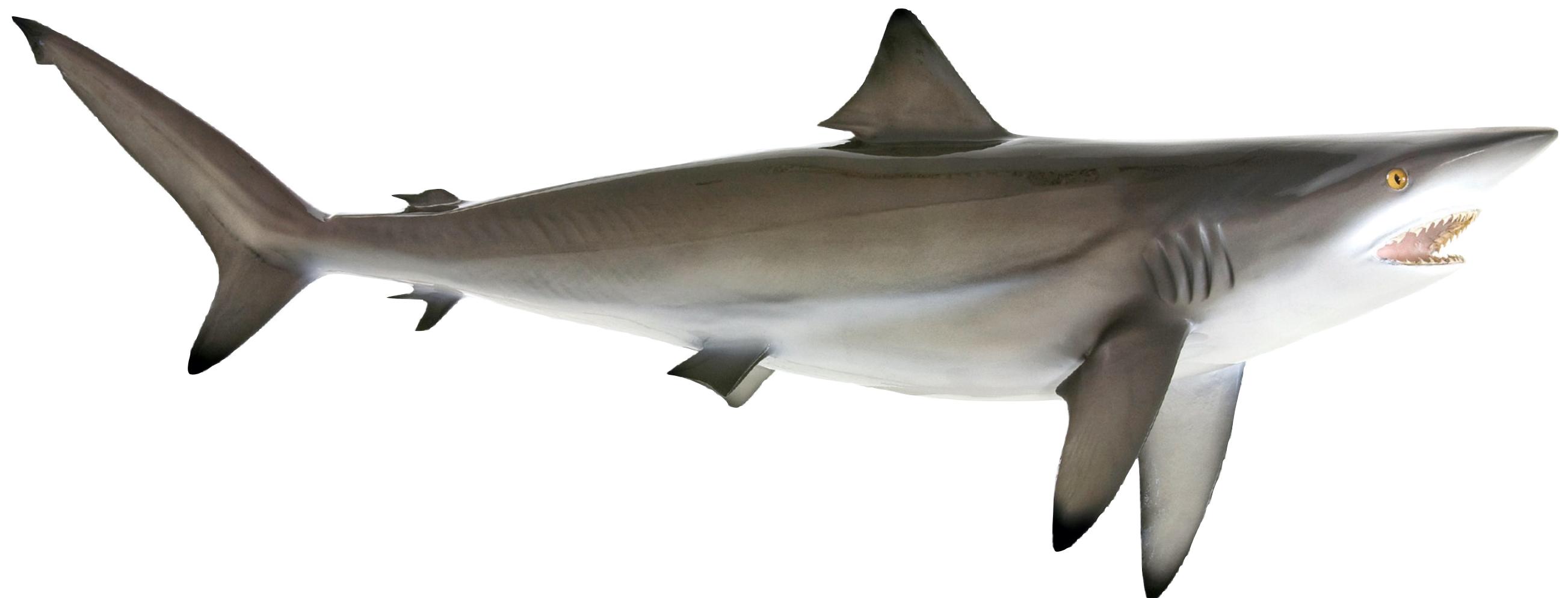
Do I have enough?

Do problems or biases exist in the data that could cause problems?

# Feature Engineering

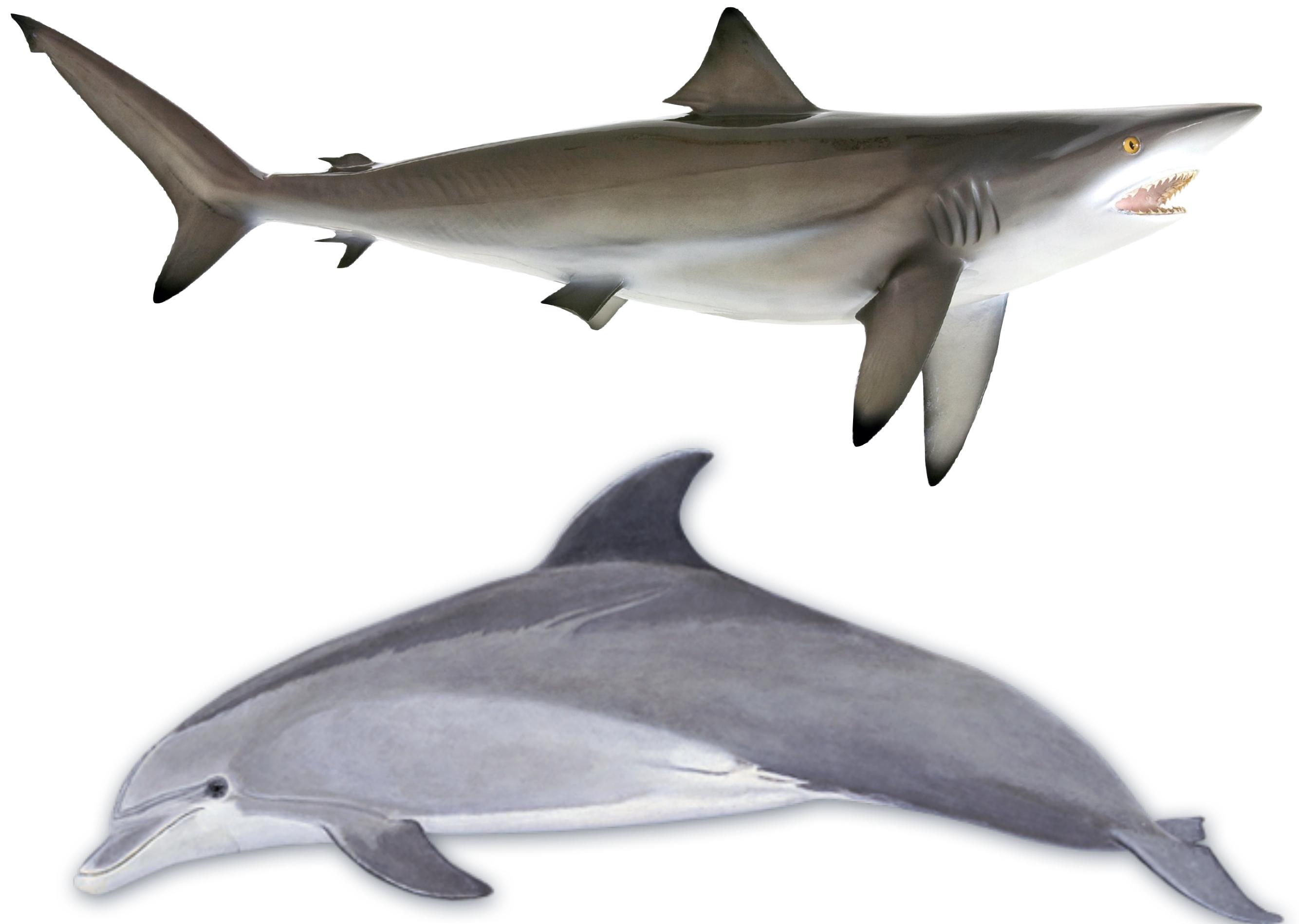
- Define what you are trying to measure. These will become the observations or rows of your final dataset
- Define how you will mathematically represent your data. This will be come the features or columns of your final dataset.

# Feature Engineering



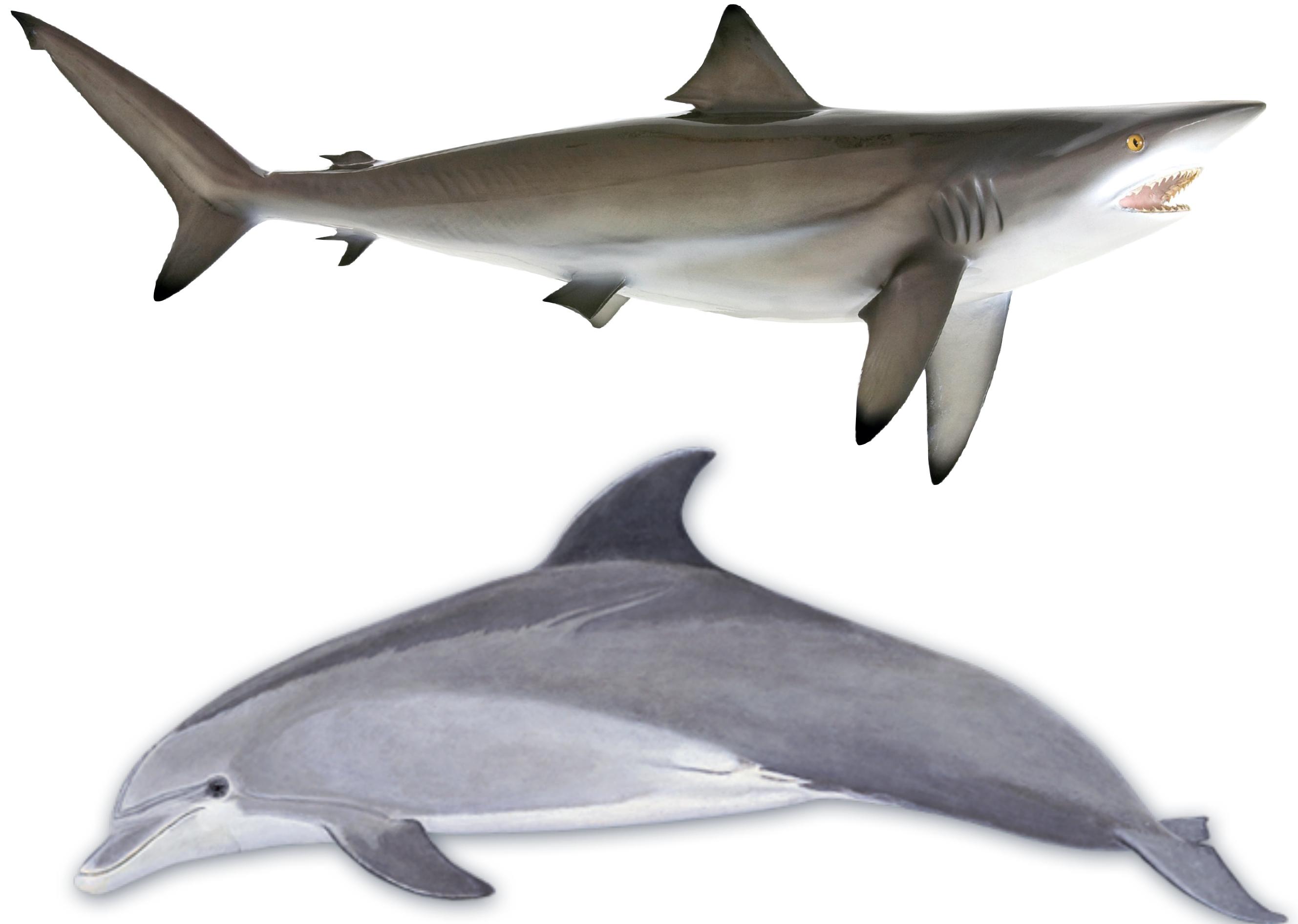
Feature	Value
Color	Gray
Fins	7
Predator	TRUE

# Feature Engineering



Feature	Value
Color	Gray
Fins	7
Predator	TRUE

# Feature Engineering



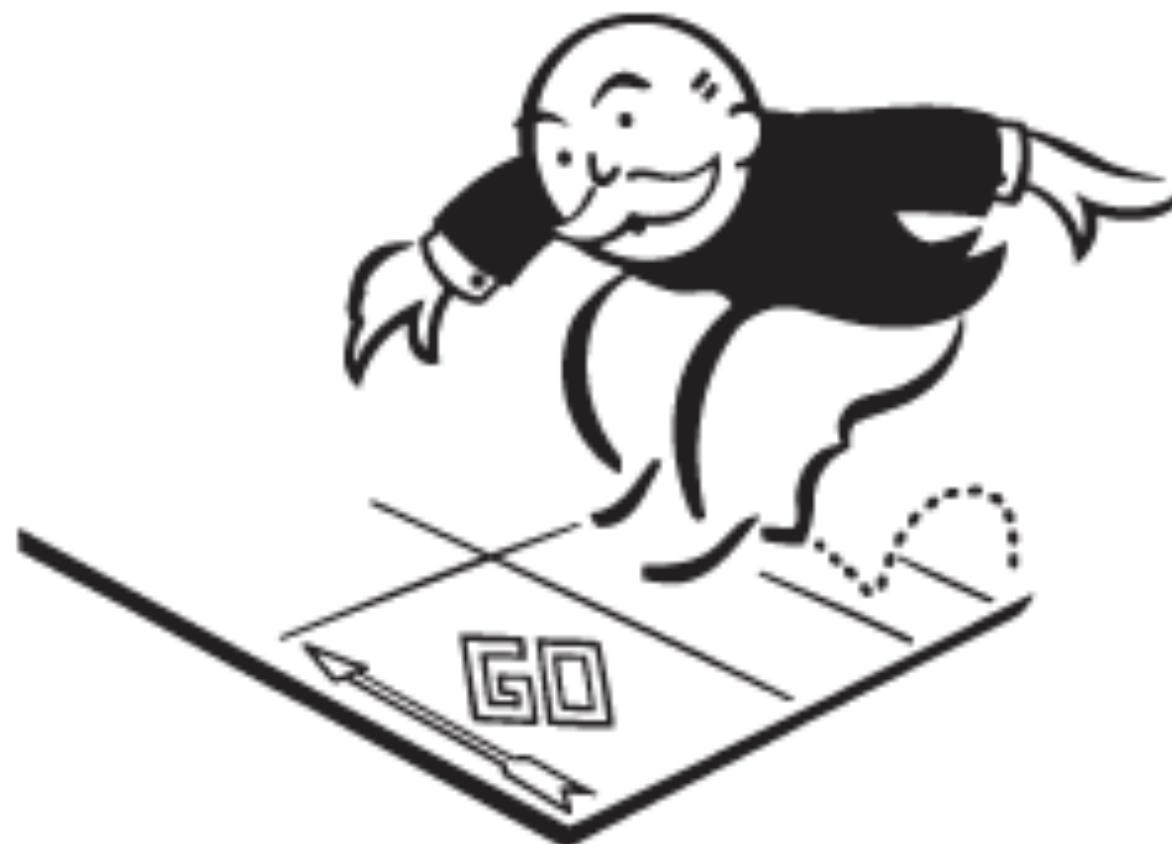
Feature	Value
Color	Gray
Fins	7
Predator	TRUE
<b>Mammal</b>	<b>TRUE</b>

# Build and Tune your Model

- Believe it or not, this is the easy part.
- Most of this is **done using libraries** like scikit-learn, mllib, tensorflow, caret or keras, and **many steps can be automated**.
- You can even do it in Splunk.

# Evaluate Performance

- Use various scoring methods, or write your own to determine model performance.
- Go back to step 1 and repeat! (Do not pass go, do not collect \$200)



# Discussion

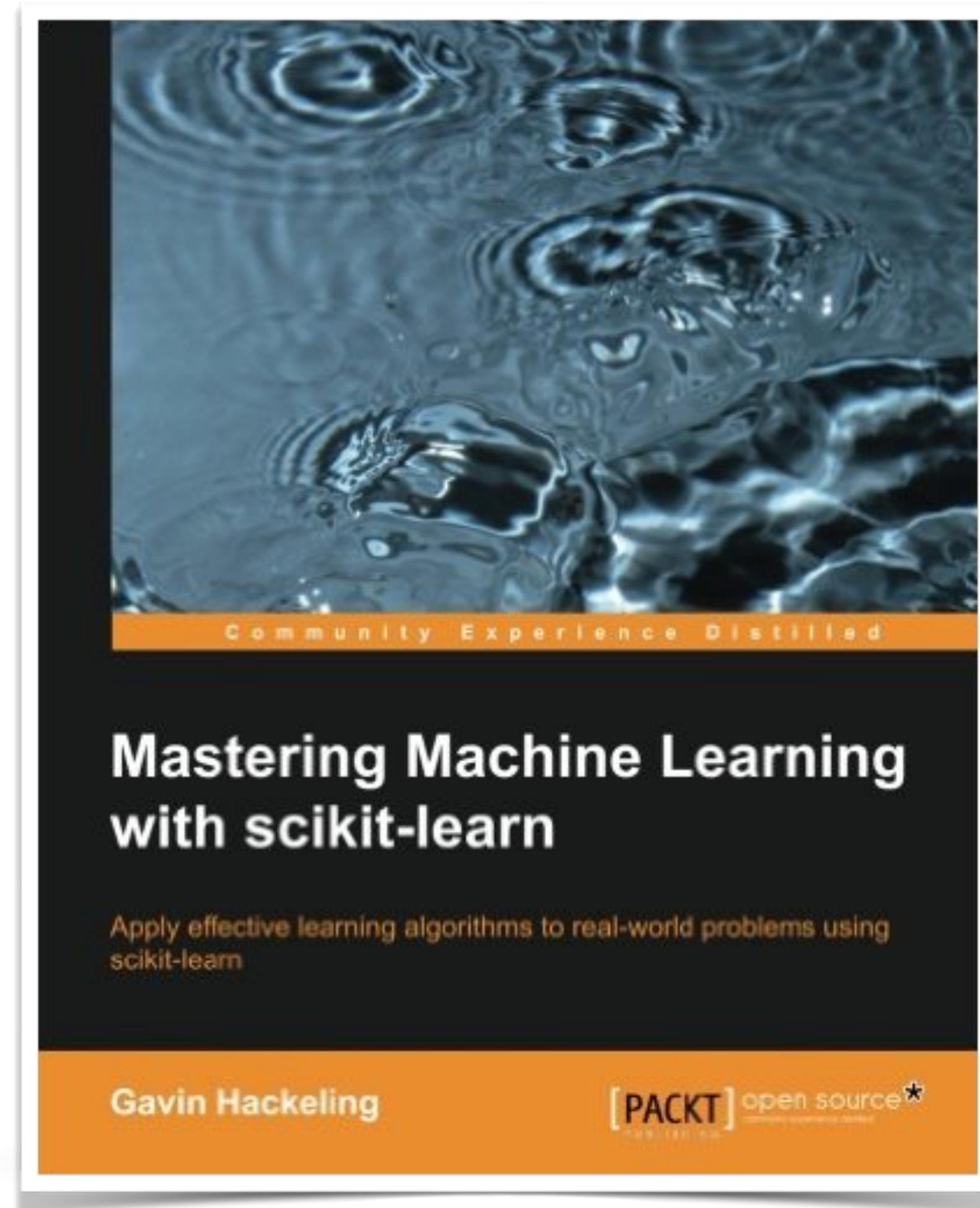
- Consider that you are building a system to identify fraudulent credit card transactions. What are some features that you would want to capture?
- What are the challenges you could foresee in this problem?

# The Python Data Science Ecosystem

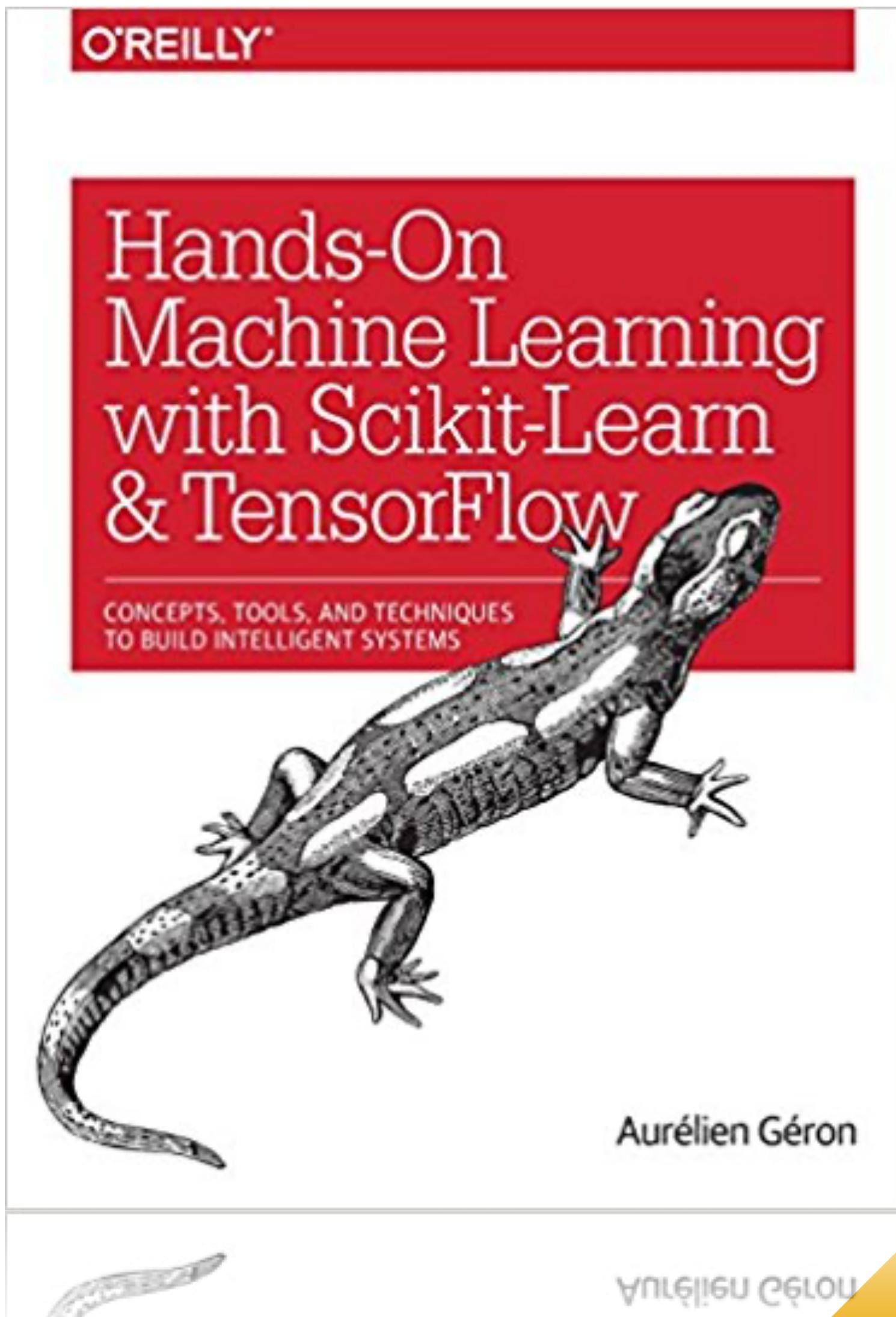
# Machine Learning Ecosystem

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

# Recommended Reading

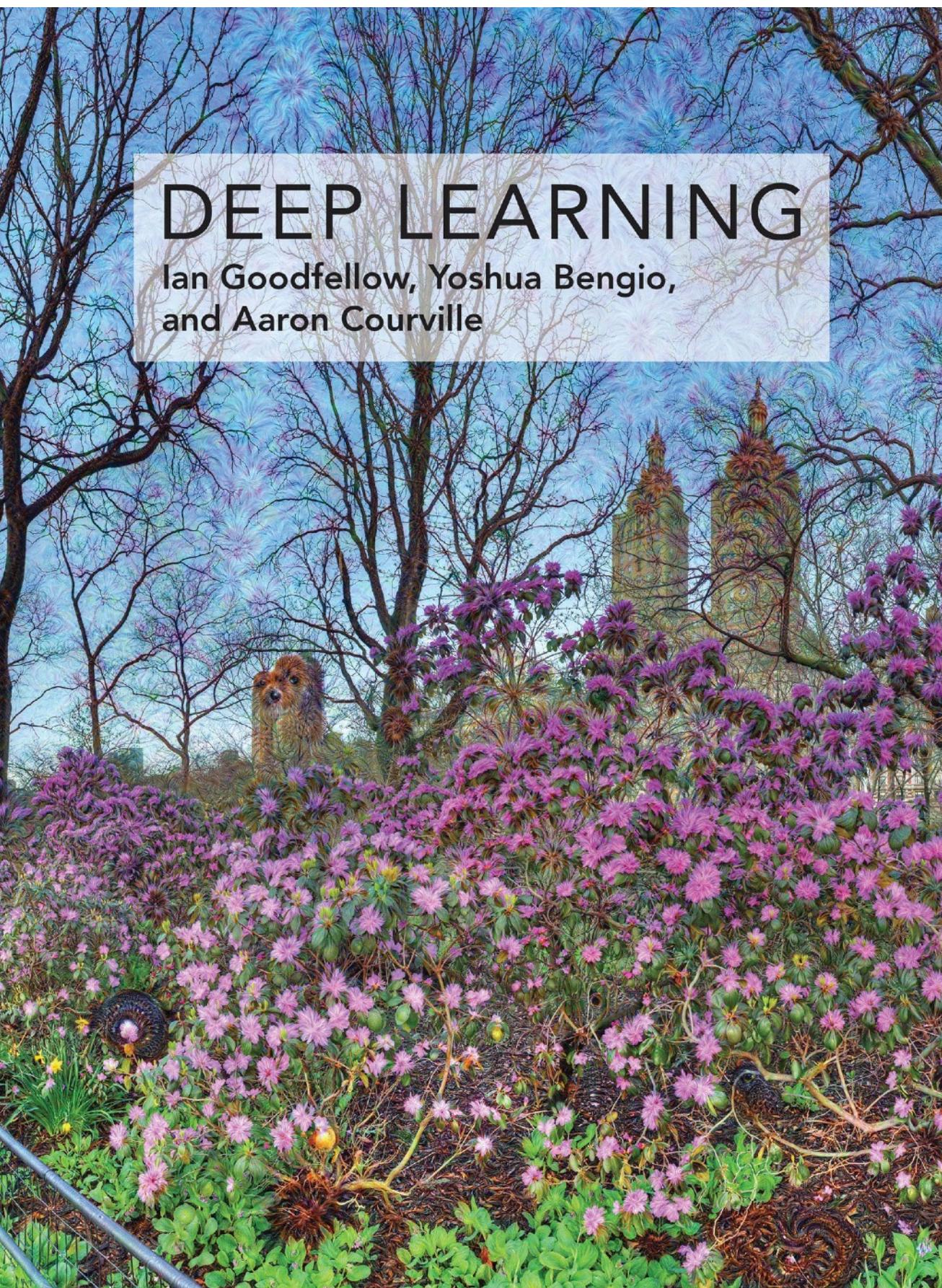


# Recommended Reading



GTK Cyber

# Recommended Reading



<http://www.deeplearningbook.org/>

O'REILLY®

# Machine Learning & Security

PROTECTING SYSTEMS WITH DATA AND ALGORITHMS



Clarence Chio & David Freeman

GTK Cyber

O'REILLY®



# Feature Engineering for Machine Learning

PRINCIPLES AND TECHNIQUES FOR DATA SCIENTISTS

Alice Zheng & Amanda Casari

GTK Cyber

O'REILLY



# Learning Apache Drill

QUERY AND ANALYZE STRUCTURED DATA

Charles Givre & Paul Rogers

GTK Cyber

# The Virtual Machine: Griffon

```
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)
```

**Do Data Science, Not Sysadmin**

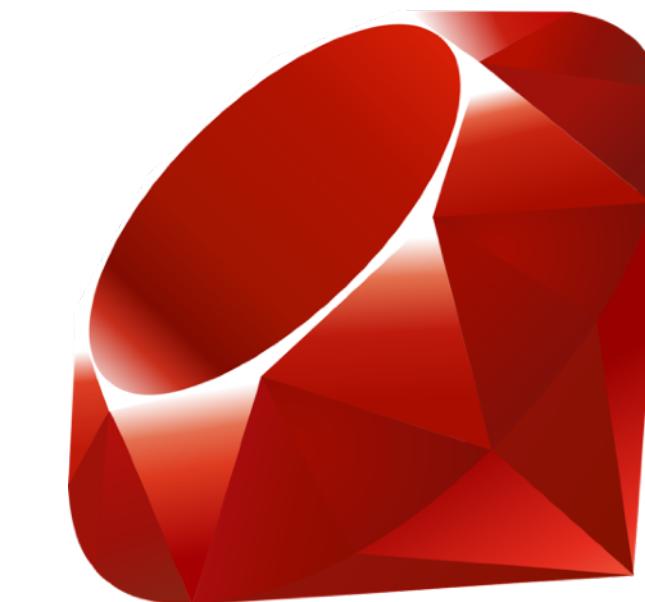
**Built on Ubuntu MATE**

# **Easy to Use**

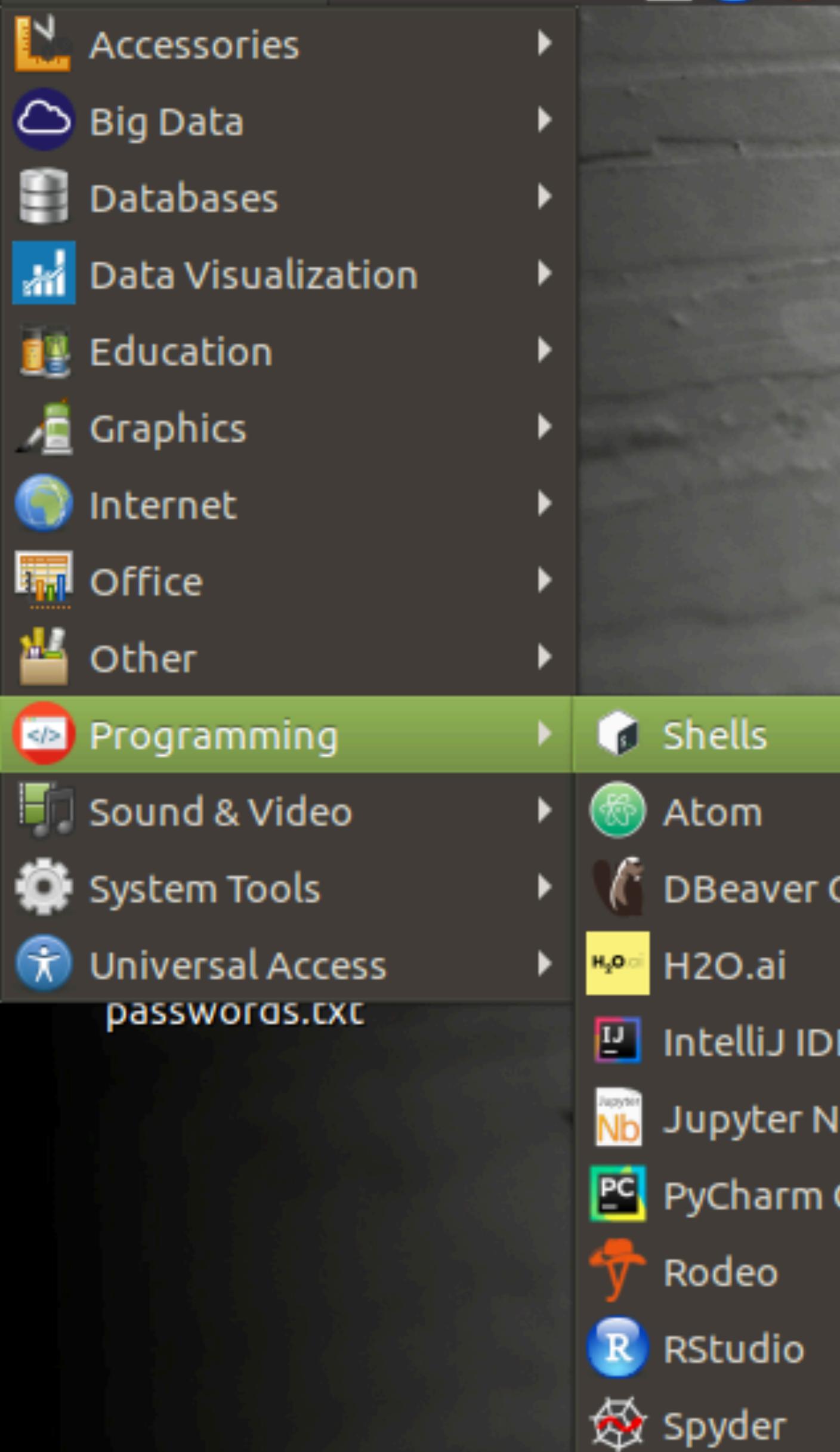
# Programming Languages

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

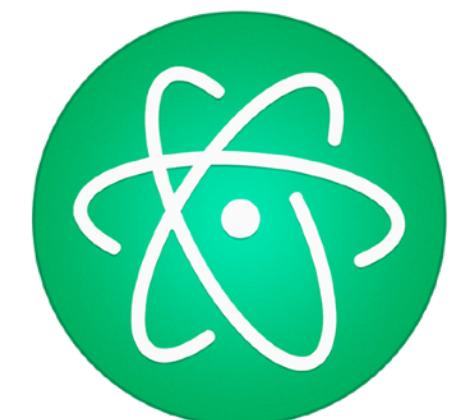
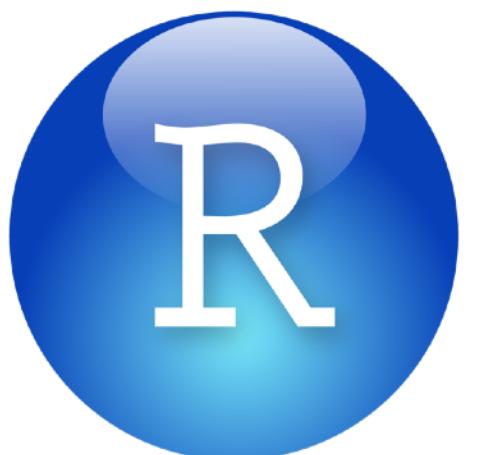
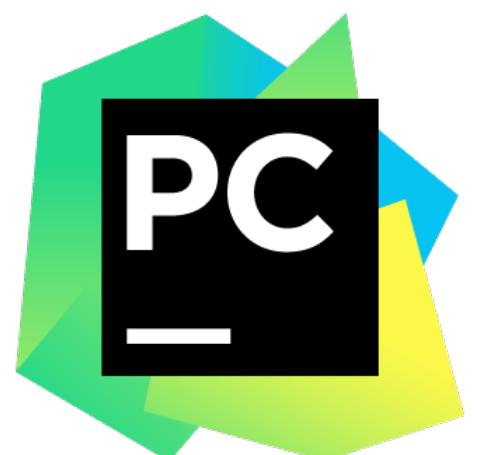
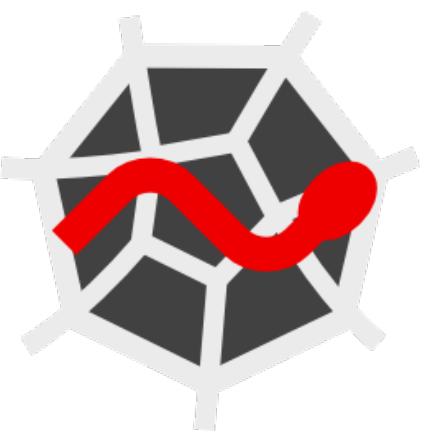
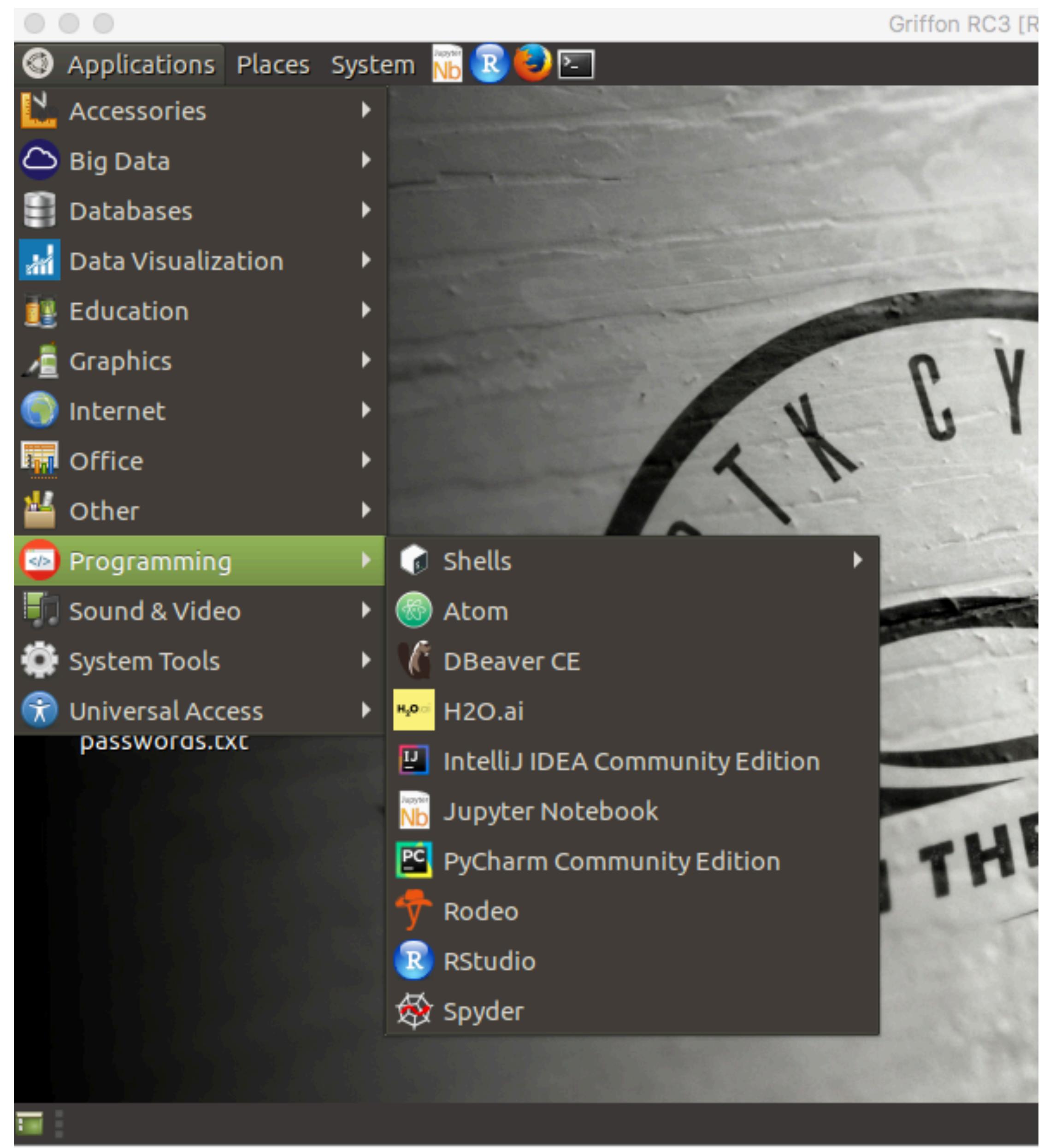
# Scripting Languages

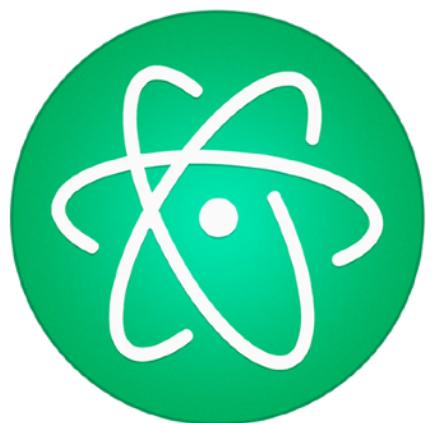


Applications Places System



# **Editors & Notebooks**





# Atom

Project Merlin Beta 3 (Small updates) [Running]

Applications Places System R Jupyter Nb

python\_demo.py — /usr/share/atom/resources — Atom

File Edit View Selection Find Packages Help

resources

python\_demo.py

```
1 import pandas as pd
2 df = pd.DataFrame([2,3,4,5,6,7,8])
```



Project Merlin Development Version (Alpha 0.2) [Running]

Tue May 10, 23:47

Applications Places System Firefox R Nb

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

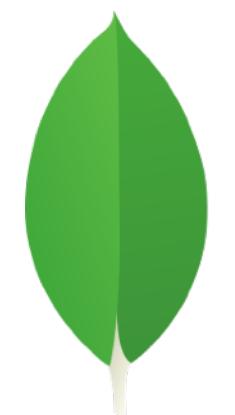
node\_modules

A screenshot of a Linux desktop environment showing a Jupyter Notebook interface. The title bar indicates "Project Merlin Development Version (Alpha 0.2) [Running]" and the date "Tue May 10, 23:47". The desktop panel includes icons for Applications, Places, System, Firefox, R, and Nb. A terminal window titled "Home - Mozilla Firefox" is open, displaying the Jupyter Notebook dashboard at "localhost:8888/tree". The dashboard shows the "jupyter" logo and tabs for "Files", "Running", and "Clusters". Below the tabs, a message says "Select items to perform actions on them." A sidebar on the right lists options for "Upload" and "New" (with "Text File", "Folder", and "Terminal" listed), followed by a list of kernels: "Notebooks", "Julia 0.4.2", "Python 2", "Python 3", "R", "Ruby 2.1.5", "Scala 2.11", and "pySpark (Spark 1.6.0)". The main content area shows a file tree with folders like "anaconda3", "Desktop", "Documents", "Downloads", "metastore\_db", "Music", and "node\_modules".



# Jupyter Notebook

- Python 2 & 3
- R
- Ruby
- Scala
- PySpark



mongoDB<sup>®</sup>





# MySQL®

Project Merlin Beta 3 (Before beaker 1.6 update) [Running]

Fri Aug 19, 00:14

MySQL Workbench

Local instance 3306 Local instance 3306

File Edit View Query Database Server Tools Scripting Help

SQL SQL Data Modeler Data Definition Data Manipulation Data Comparison Data Transformation Data Migration Data Cleaning Data Mining Data Science Data Visualization Data Integration Data Quality Data Governance Data Privacy Data Security Data Compliance Data Governance Data Privacy Data Security Data Compliance

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- phpmyadmin
- test
  - Tables

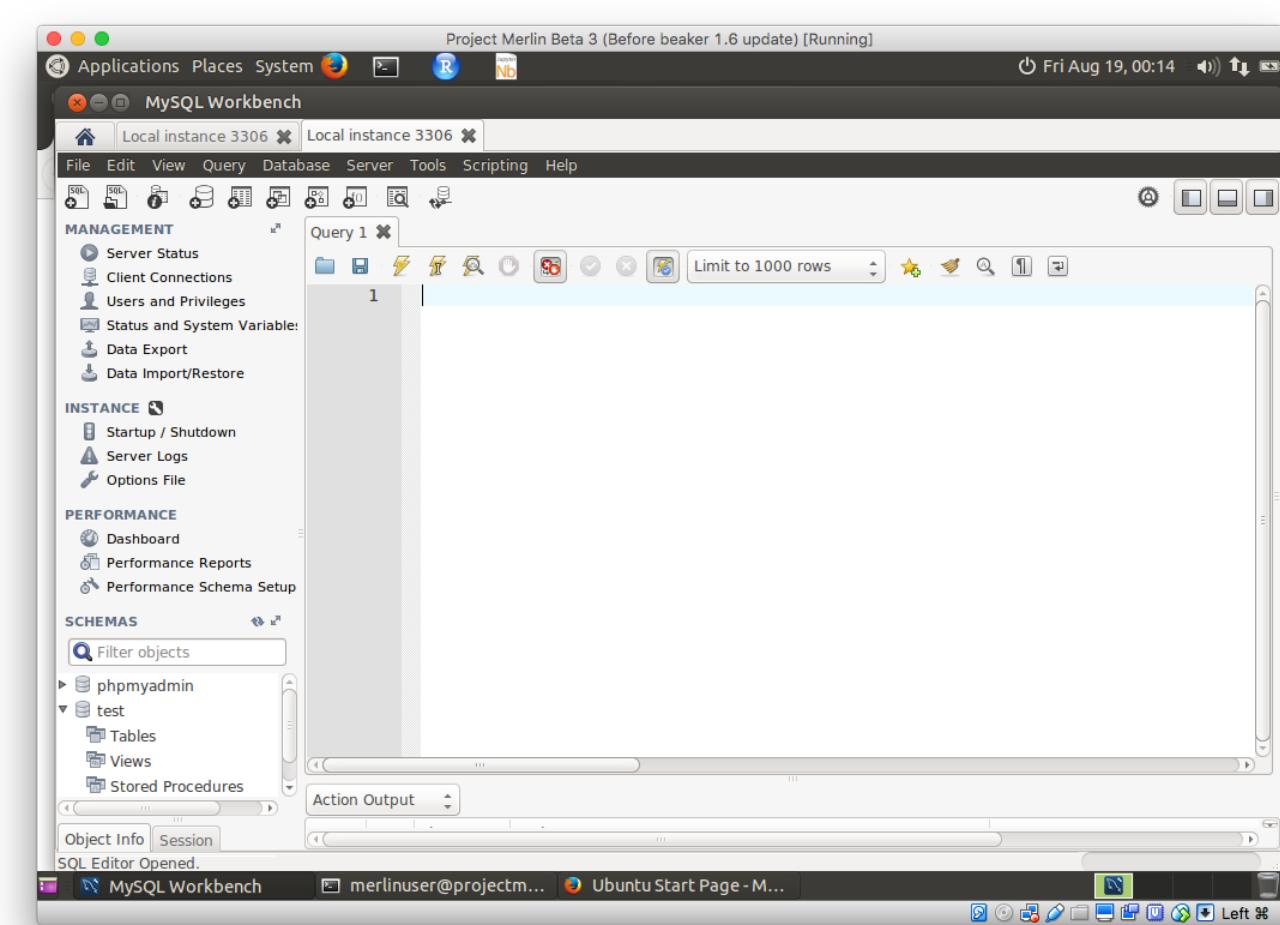
Query 1

Limit to 1000 rows

The screenshot shows the MySQL Workbench interface. The title bar indicates it's running a project named 'Merlin Beta 3' before the 'beaker 1.6 update'. The main window has tabs for 'Local instance 3306' and 'Local instance 3306'. The menu bar includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The toolbar contains various icons for management, data manipulation, and performance monitoring. On the left, there's a sidebar with sections for 'MANAGEMENT', 'INSTANCE', 'PERFORMANCE', and 'SCHEMAS'. The 'SCHEMAS' section shows the 'test' schema expanded, revealing its tables. A query editor window titled 'Query 1' is open at the bottom, with a limit of 1000 rows specified.



# MySQL®



Project Merlin Beta 3 (Before beaker 1.6 update) [Running]

Fri Aug 19, 00:30

localhost / localhost | phpMyAdmin 4.4.13.1deb1 - Mozilla Firefox

localhost / localhost... +

localhost/phpmyadmin/index.php?token=62fb6ffb630fba16f23788ee21dc4e

Search

Applications Places System R Nb

phpMyAdmin

Server: localhost

Databases SQL Status Users Export Import Settings More

General Settings

- Change password
- Server connection collation: utf8mb4\_unicode\_ci

Database server

- Server: Localhost via UNIX socket
- Server type: MySQL
- Server version: 5.6.31-0ubuntu0.15.10.1 - (Ubuntu)
- Protocol version: 10
- User: merlinuser@localhost
- Server charset: UTF-8 Unicode (utf8)

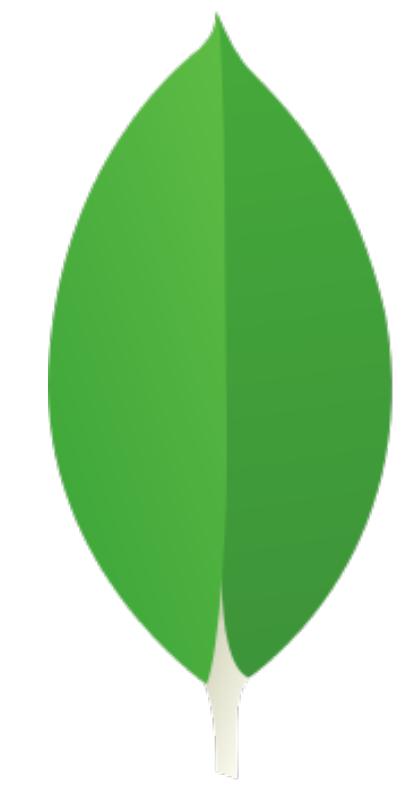
Appearance Settings

- Language: English
- Theme: pmahomme
- Font size: 82%

Web server

- nginx/1.10.1
- Database client version: libmysql - 5.6.31
- PHP extension: mysqli
- PHP version: 5.6.11-1ubuntu3.4

phpMyAdmin



mongoDB®



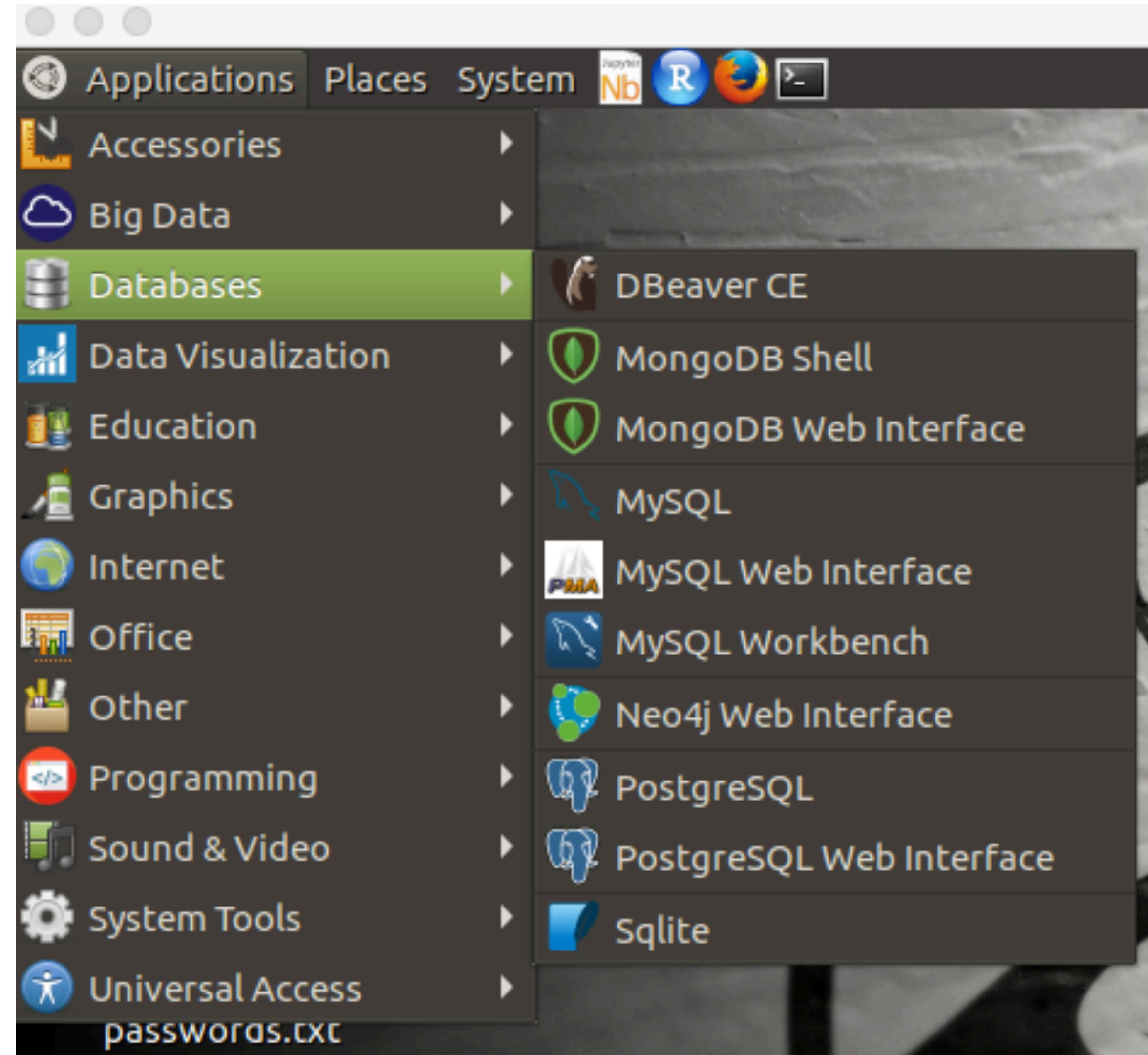
Mongo Express Database: db

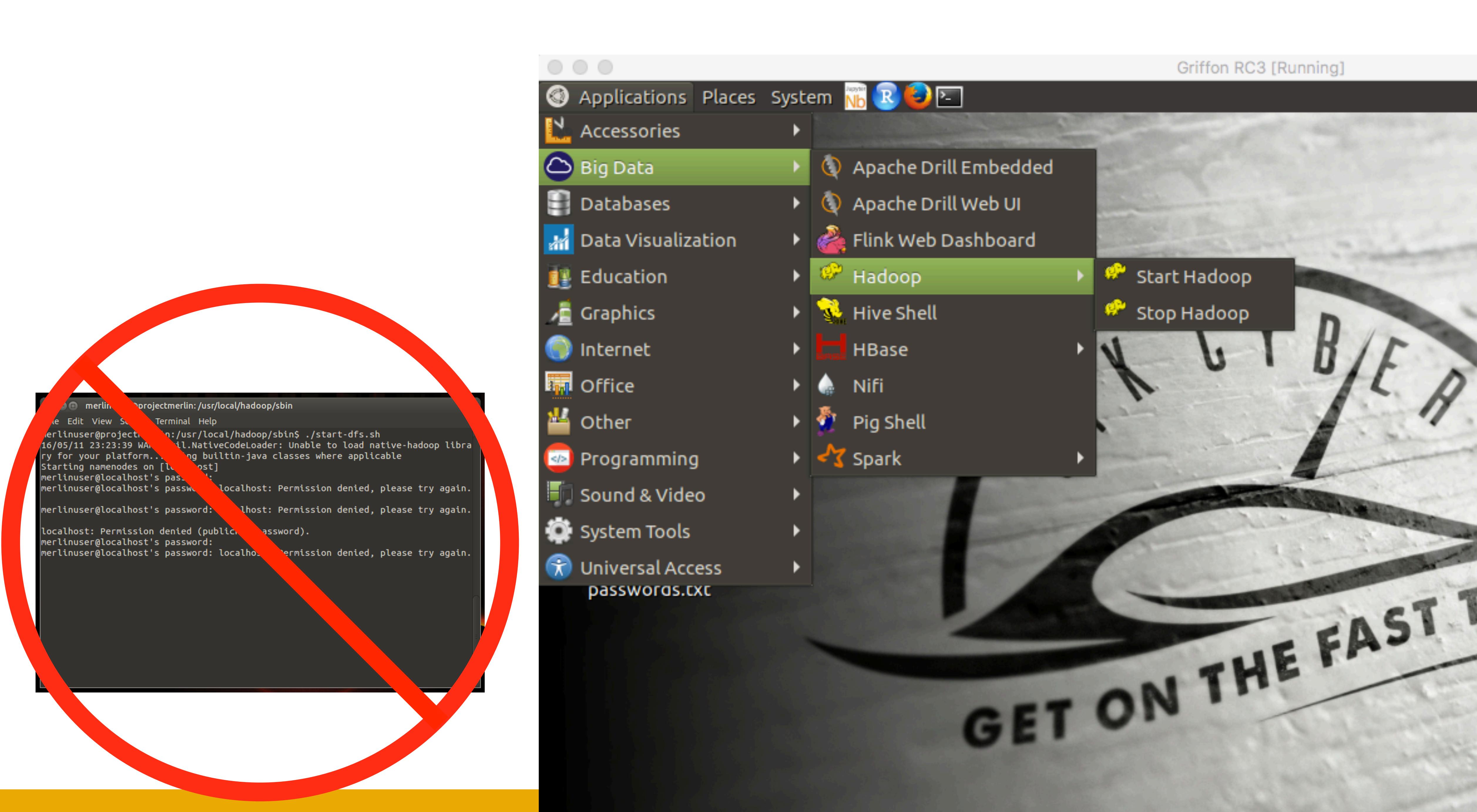
Collection "test" deleted!

Collections

Collection Name [+ Create collection](#)

[View](#) [Export](#) [\[JSON\]](#) system.indexes [Del](#)





# Questions?

# Using Jupyter Notebook

# Using Jupyter Notebook

The screenshot shows the Jupyter Notebook web interface. At the top, there's a navigation bar with links for Files, Running, Clusters, Formgrader, Assignments, BeakerX, and Nbextensions. On the far right of the header are Quit and Logout buttons. Below the header is a message: "Select items to perform actions on them." To the right of this message is a large text area with the heading "Open a notebook" and a black arrow pointing to the "New" button in a dropdown menu. The "New" menu is circled with a black oval. The menu lists various kernel types: Bash, Clojure, Groovy, Java, Javascript (Node.js), Kotlin, PHP, Python 3, R, Ruby 2.5.1, SQL, Scala, and Other: Text File, Folder, Terminal. At the bottom of the menu, the date "14 days ago" is displayed. On the left side of the interface is a file browser showing a directory structure with folders like anaconda3, Desktop, Documents, Downloads, drill, metastore\_db, Music, Pictures, Public, snap, sqldpad, Templates, and Videos.

# Using Jupyter Notebook

jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Navigate Widgets Help Snippets Trust

Run

Demo Notebook

This is a markdown cell. It contains the instructions as to how to do the exercises.

In [1]:

```
1 #This is an executable cell
2 print( "This is a python cell")
```

This is a python cell

In [ ]:

```
1 |
```

A screenshot of the Jupyter Notebook interface. The title bar shows 'jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes)'. The toolbar includes standard options like File, Edit, View, Insert, Cell, Kernel, Navigate, Widgets, Help, and Snippets, along with a 'Trust' button. Below the toolbar is a toolbar with various icons for file operations, code execution, and other functions. A large black arrow points from the text 'Run a cell' down to the 'Run' button in the toolbar, which is circled in black. The main content area is titled 'Demo Notebook' and contains a markdown cell with the text 'This is a markdown cell. It contains the instructions as to how to do the exercises.' Below it is an executable cell with the following Python code:  
In [1]:  
1 #This is an executable cell  
2 print( "This is a python cell")  
This is a python cell  
In [ ]:  
1 |

Run a cell

# Using Jupyter Notebook

The screenshot shows the Jupyter Notebook interface. At the top, there's a toolbar with various icons for file operations, cell execution, and help. Below the toolbar, the title bar says "jupyter Untitled Last Checkpoint: 18 minutes ago (unsaved changes)". On the right side of the title bar are a Python logo icon and a "Logout" button. The main area is titled "Demo Notebook". It contains a markdown cell with the text: "This is a markdown cell. It contains the instructions as to how to do the exercises." Below it is an executable cell with the following code:

```
In [1]: 1 #This is an executable cell  
2 print( "This is a python cell")  
  
This is a python cell
```

Cell In [1] has been run, and its output "This is a python cell" is displayed below the code. Cell In [2] is currently selected, showing the code: "1 x = [4,5,6]".

A large black arrow points upwards from the "Variable Explorer" label towards the toolbar. A black oval highlights the "Variable Explorer" icon in the toolbar, which is a small grid with a plus sign. To the right of the toolbar, a "Variable Inspector" panel is open, showing a table with one entry: X x list 88 [4, 5, 6].

Variable  
Explorer

# Questions?

# **Module 1: Exploratory Data Analysis in One Dimension**

# **Vectorized Data Structures**

# **what are they?**

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

# Advantages of Vectorized Data Structures

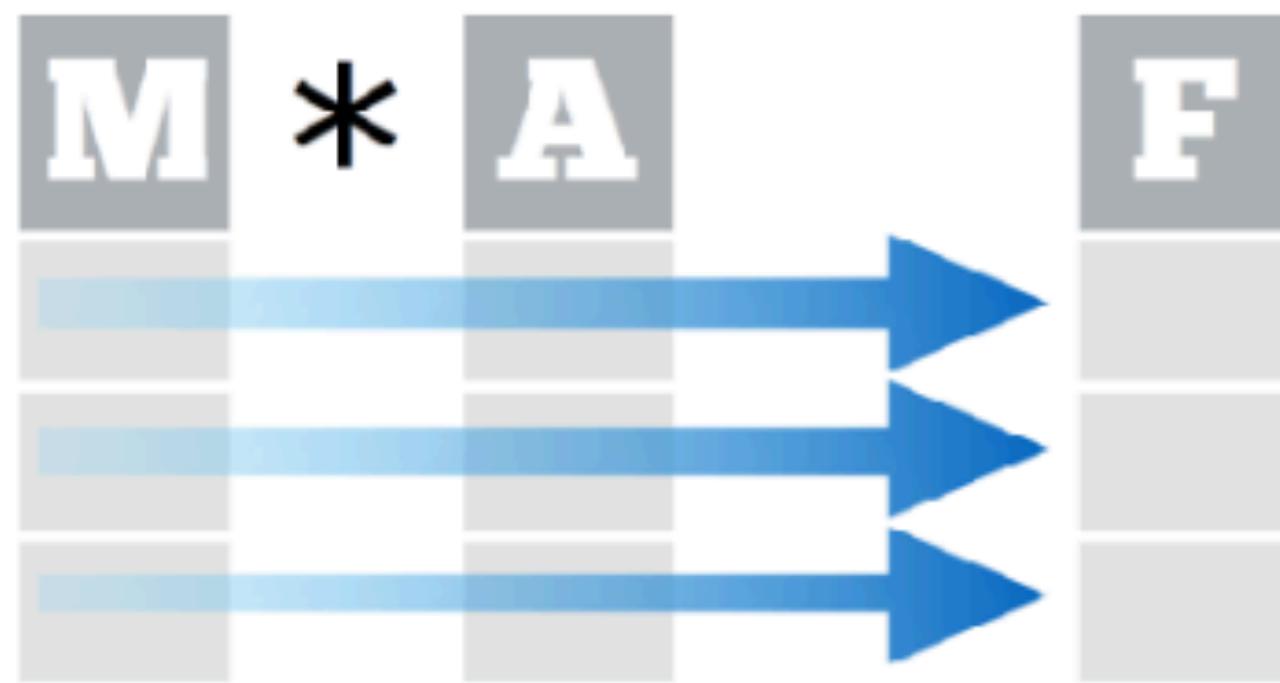
- More readable and simpler code
- Allows the module to optimize for CPU and Memory usage
- Pass serialized objects from one system to another

```
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**





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	<del>A three dimensional mutable data structure</del>

**Columns**

The diagram illustrates the structure of a table. A vertical double-headed arrow labeled "ROWS" points to the five rows of the table. A horizontal double-headed arrow labeled "Columns" points to the three columns: "Regd. No", "Name", and "Marks%".

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

# The Series Object

**A Series is like a list**

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

**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
```

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

# Filtering Data in a Series

```
randomNumbers [ <boolean condition> ]  
randomNumbers [ randomNumbers < 10 ]
```

12	5
21	2
24	1
27	1
29	1

dtype: int64

```
record.str.contains('Cha')
firstname      True
lastname       False
middle         False
dtype: bool
```

```
record[ record.str.contains('Cha') ]  GTK Cyber
Sintax:          Clase:
```

# 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>')
```

# Selected Date/Time Operations

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

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

```
odds.apply( addTwo )
```

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

# Explore Your Data



# 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



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

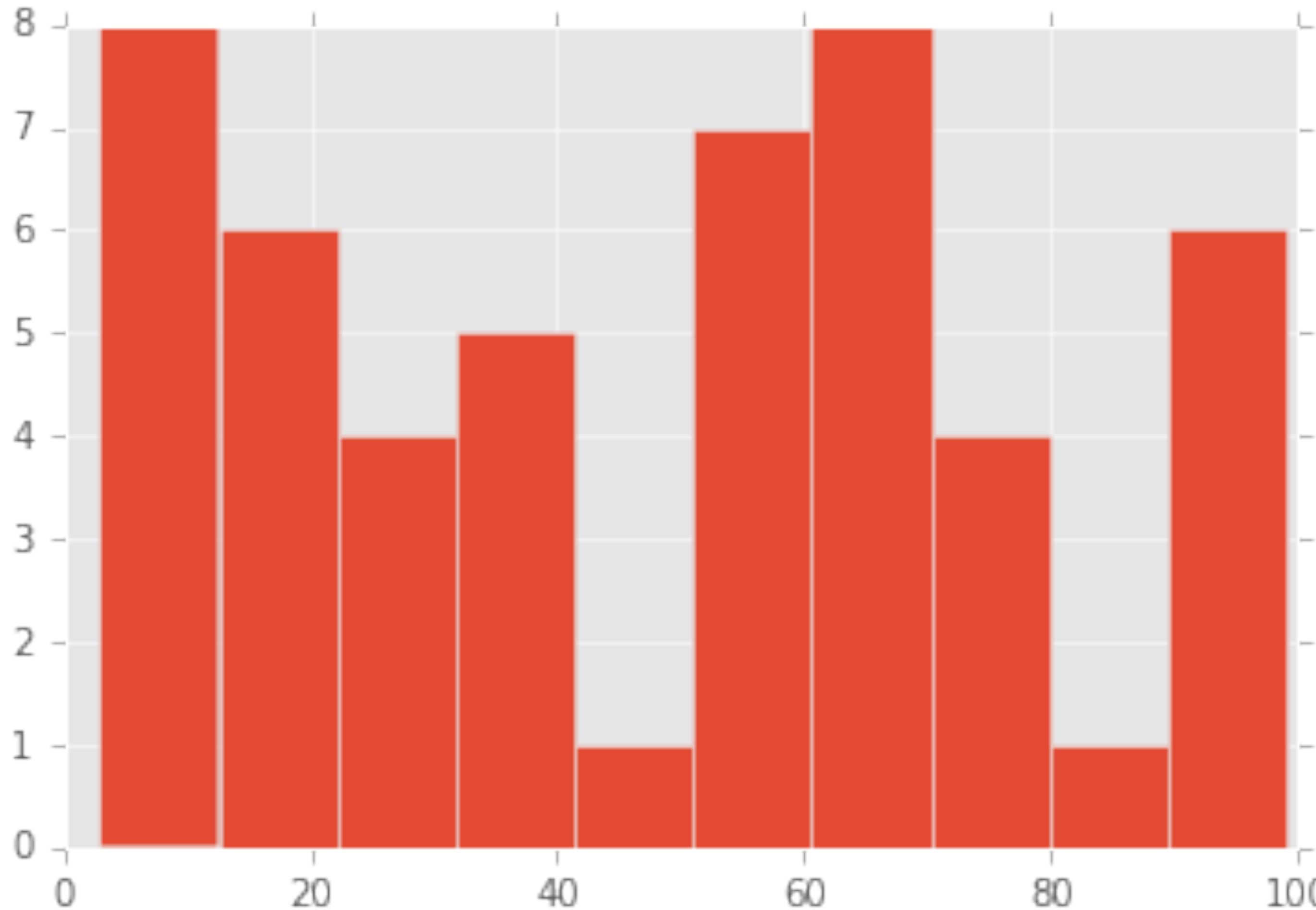
```
series.value_counts(bins=5,  
normalize=True,  
dropna=False)
```

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

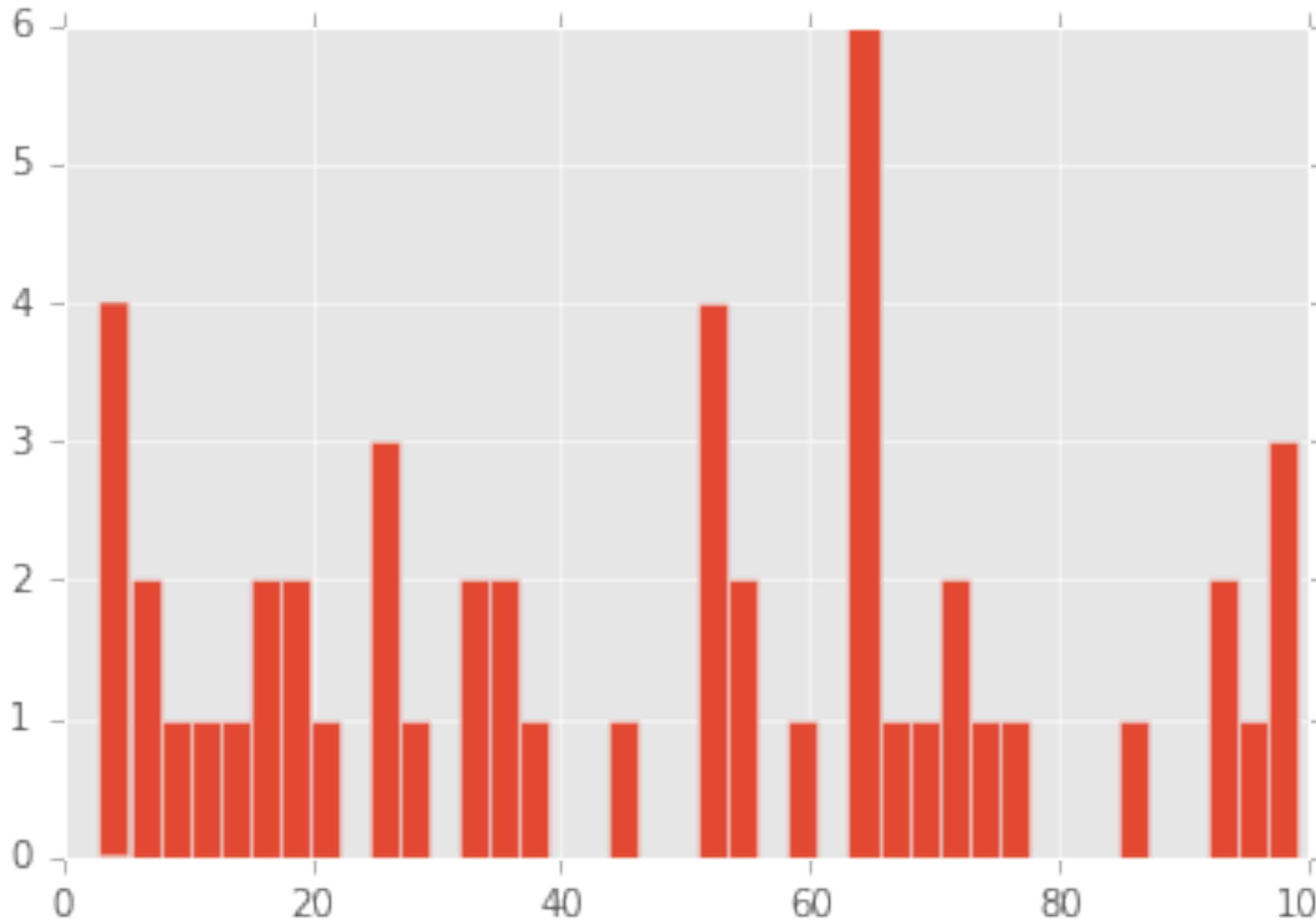
```
Out[7]: 833      7  
        811      5  
        822      5  
        899      3  
        844      3  
        855      2  
dtype: int64
```

# series.hist()

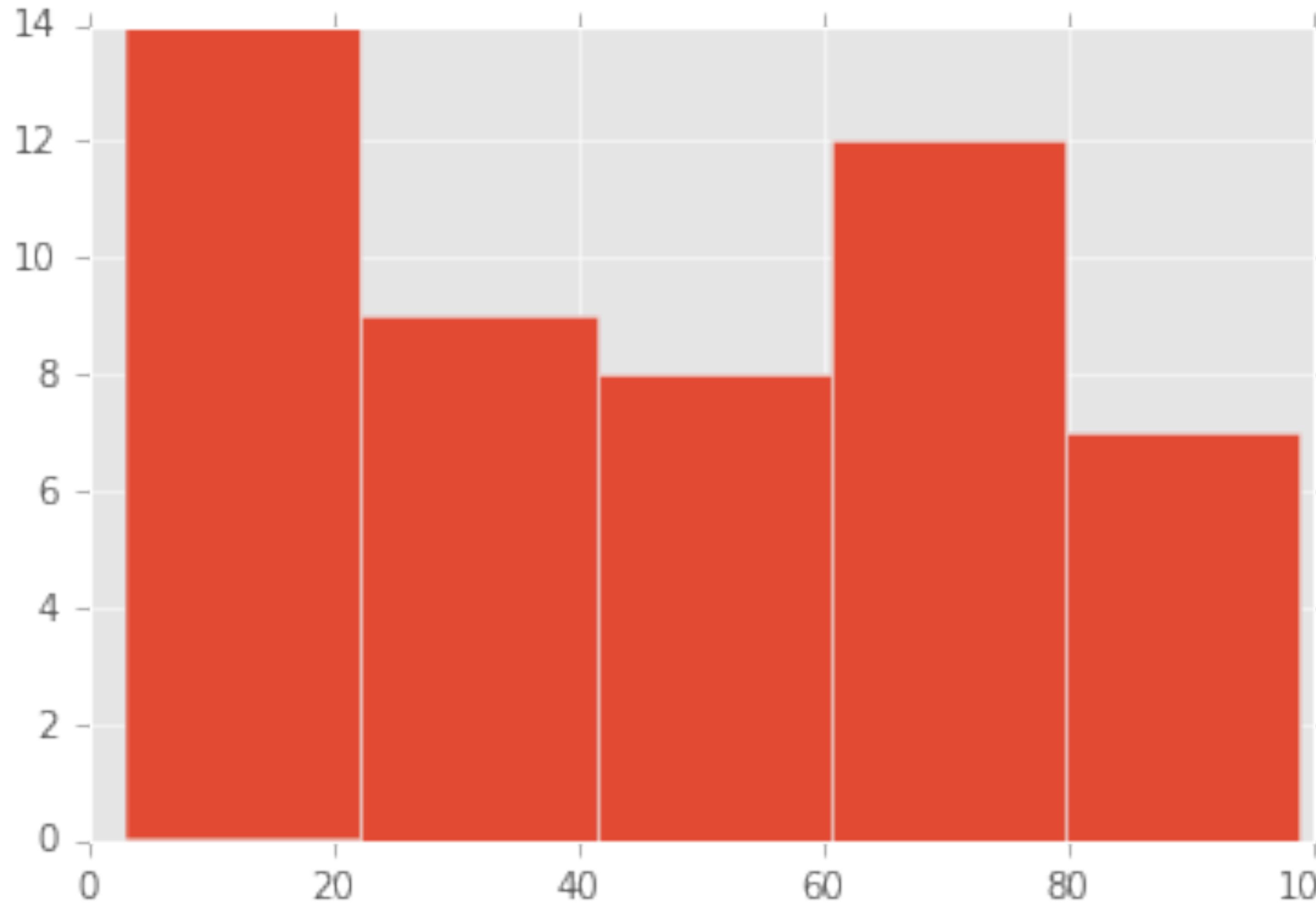
# series.hist()



# series.hist(bins=40)

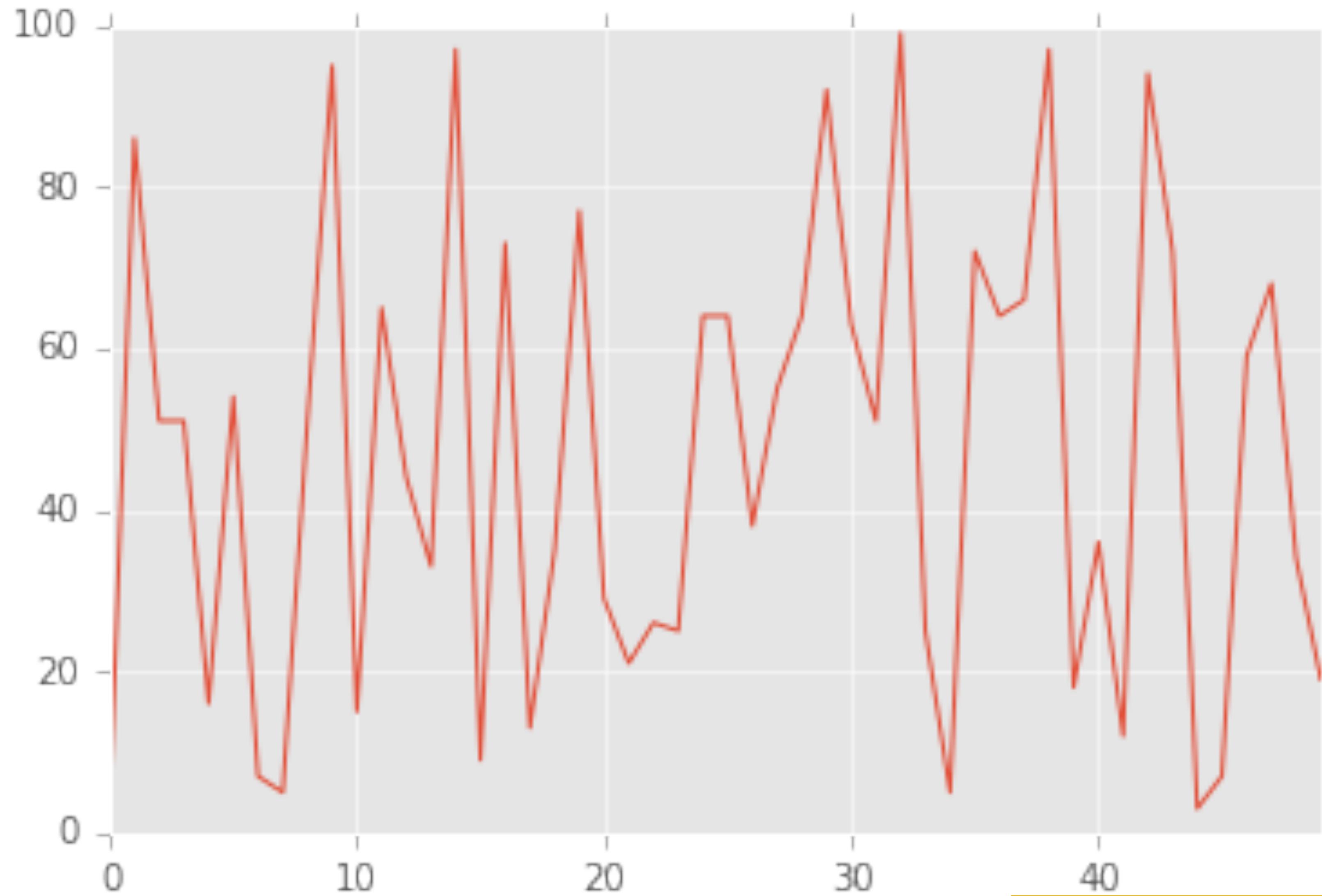


# series.hist(bins=5)

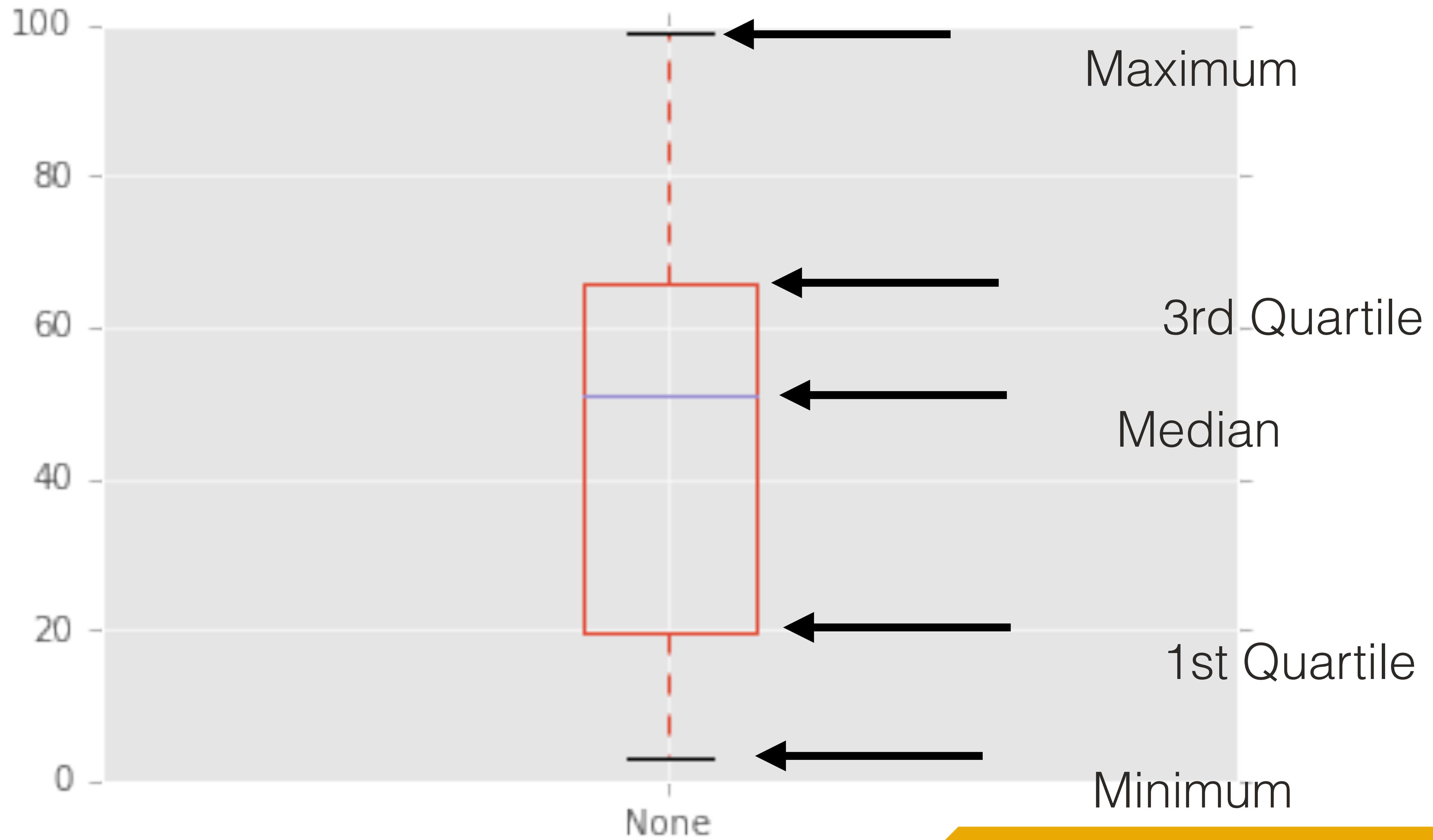


`series.plot()`

# series.plot()

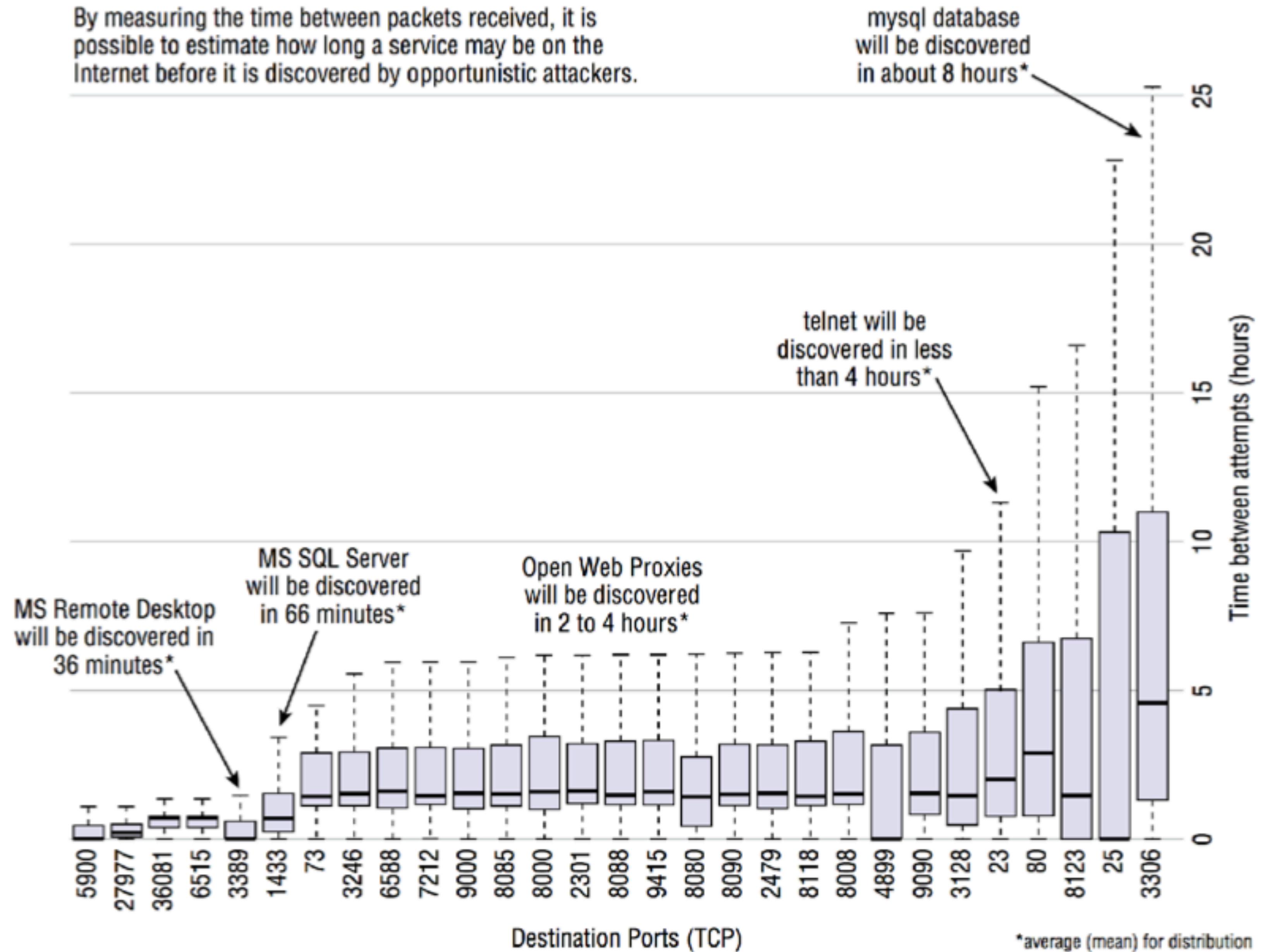


```
series.plot(kind='box')
```



## 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.



# Questions?

# **Module 2: Exploratory Data Analysis**

**Part 2:** Two Dimensional Data

# The Data Frame

```
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=<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

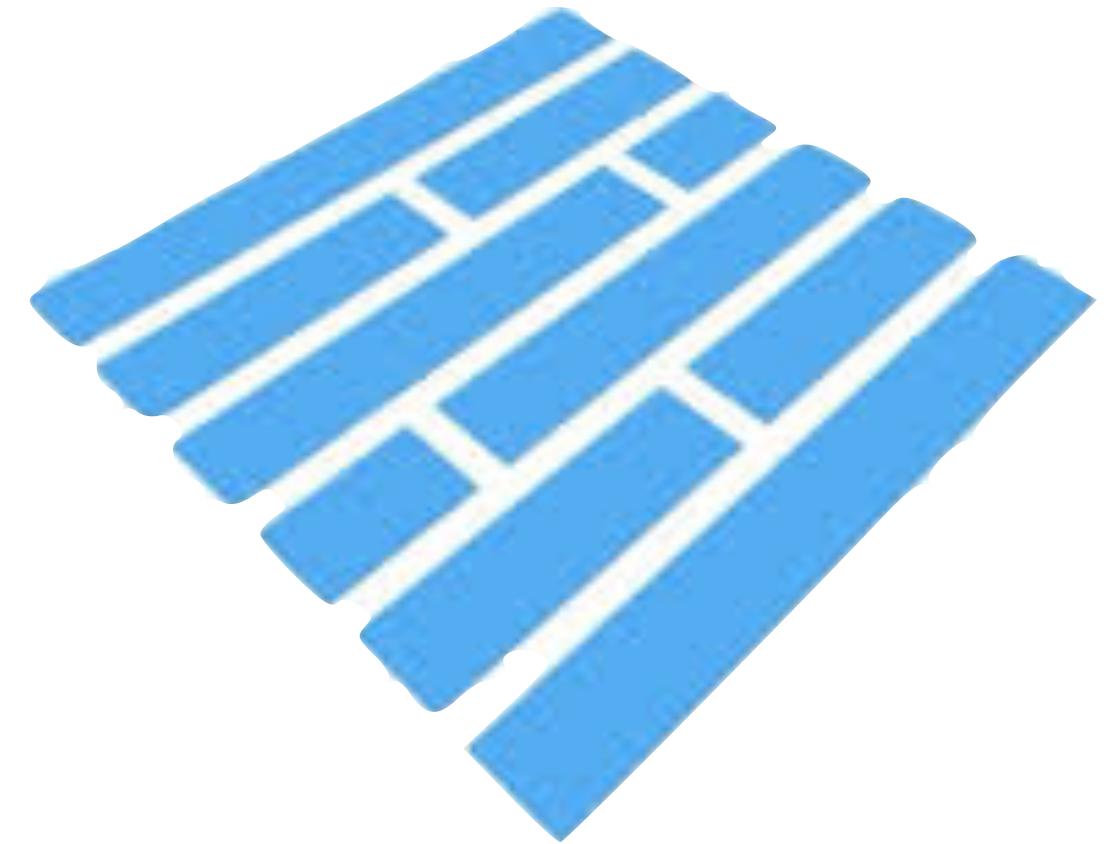
```
df = pd.read_json( <file> )
```

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()
```

# **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@localhost on test1
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\"")
```

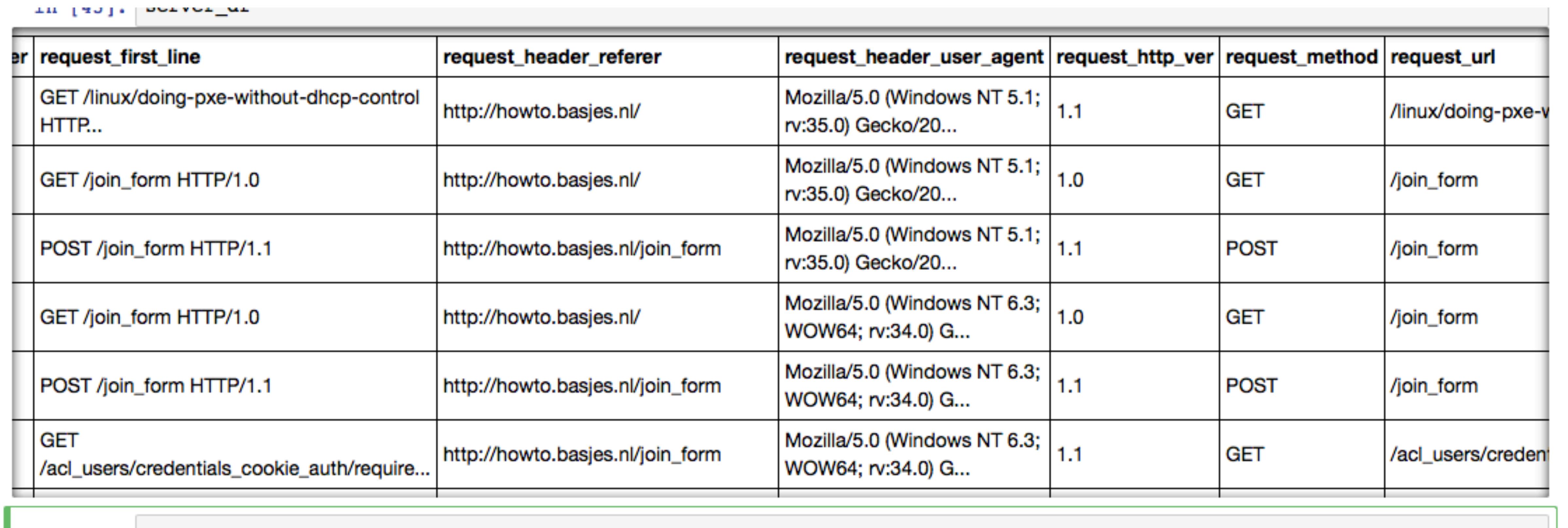
# Web Server Logs

```
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



A screenshot of a web browser window displaying a table of log entries. The table has six columns: request\_first\_line, request\_header\_referer, request\_header\_user\_agent, request\_http\_ver, request\_method, and request\_url. The data in the table is as follows:

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",  
     "checksum": 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, 'checksum': 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, 'checksum': 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, 'checksum': 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, 'checksum': 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, 'checksum': 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



elasticsearch

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()
```

# Two ways of accessing columns

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

```
series = df.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]`



Columns



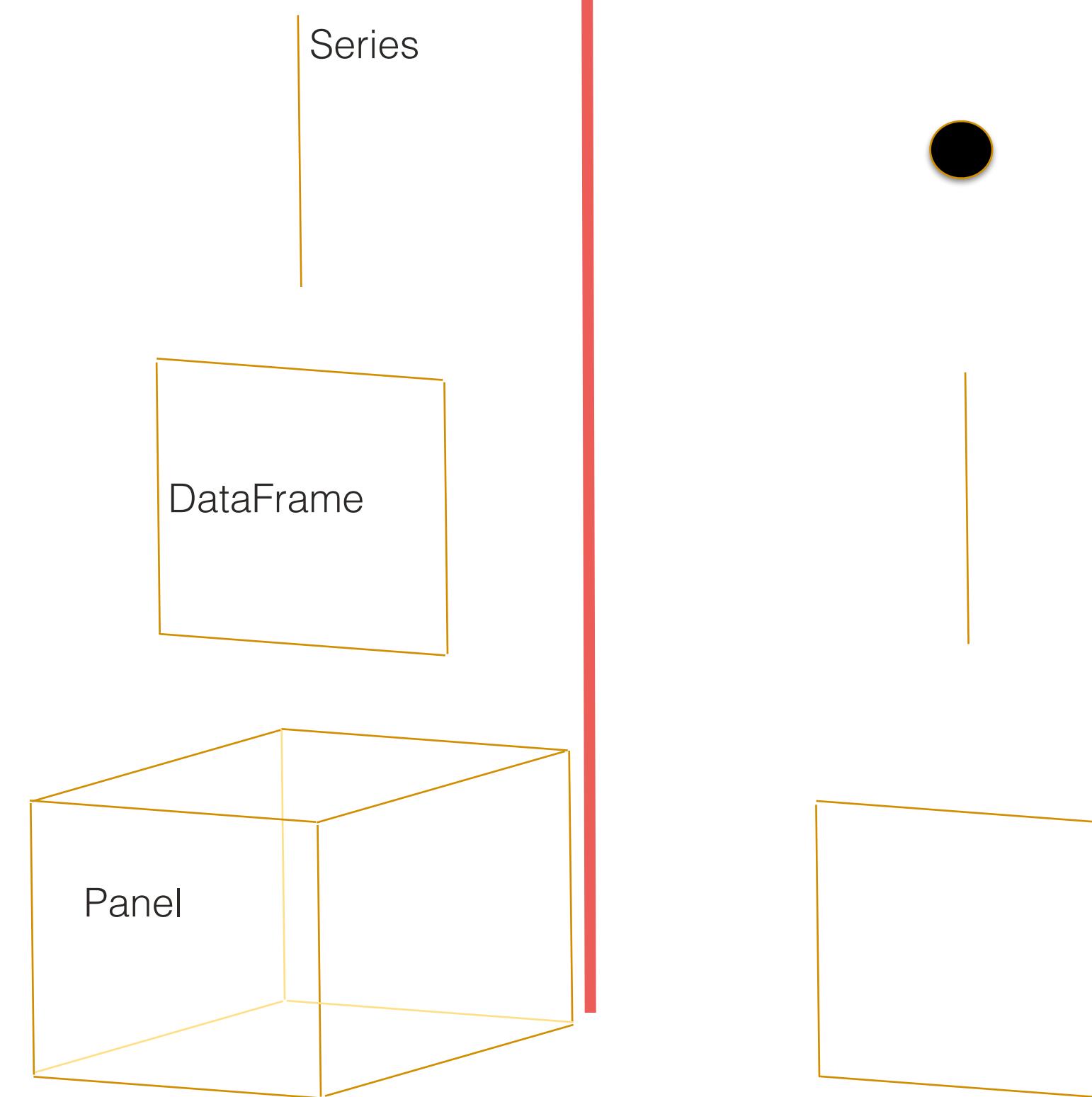
Filter

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

# `data.apply( <function> )`

IF  
Call Apply On...

THEN  
Function  
Receives



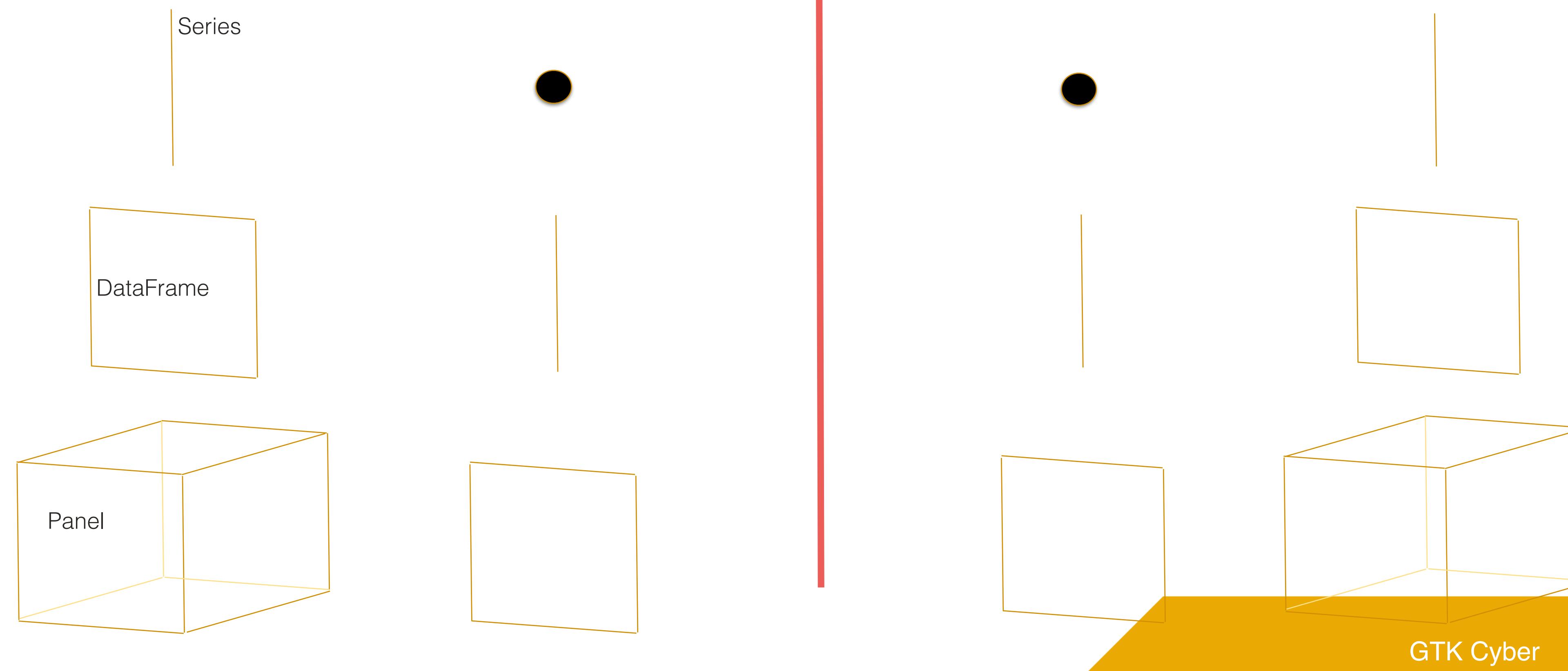
# `data.apply( <function> )`

IF  
Call Apply On...

THEN  
Function  
Receives

IF  
Function  
Returns

THEN  
Apply  
Returns...



# Questions?

1	2	3
4	5	6
7	8	9

## **data.T**

1	4	7
2	5	8
3	6	9

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

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

```
DataFrame.drop(labels,
    axis=0,
    level=None,
    inplace=False,
errors='raise')¶
```

# Merging Data Sets

# Union

Series 1

Index	Value
0	6
1	4
2	2
3	3

Series 1

Index	Value
0	6
1	4
2	2
3	3

Series 2

Index	Value
0	7
1	5
2	3
3	4

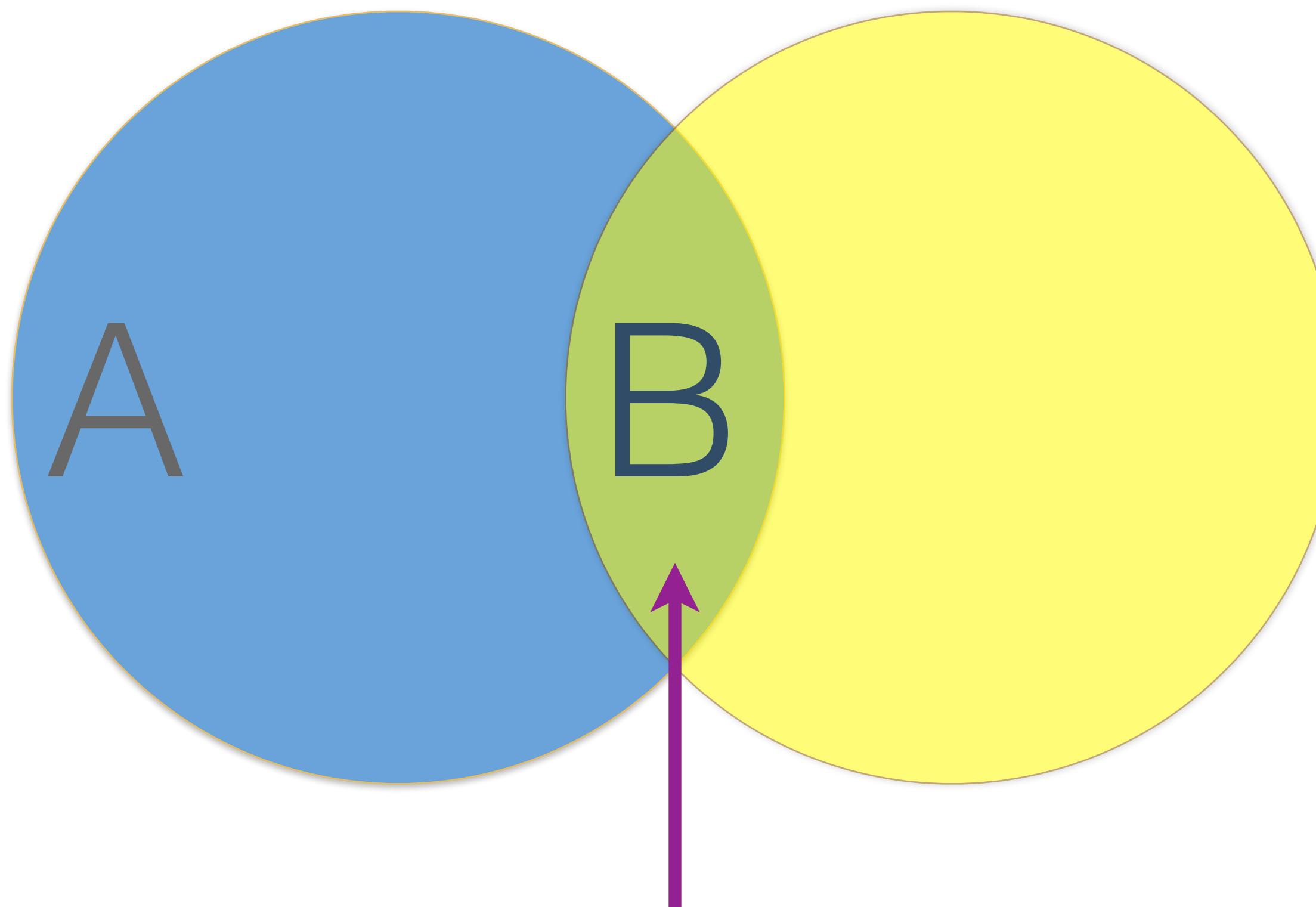
combinedSeries

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 )
```

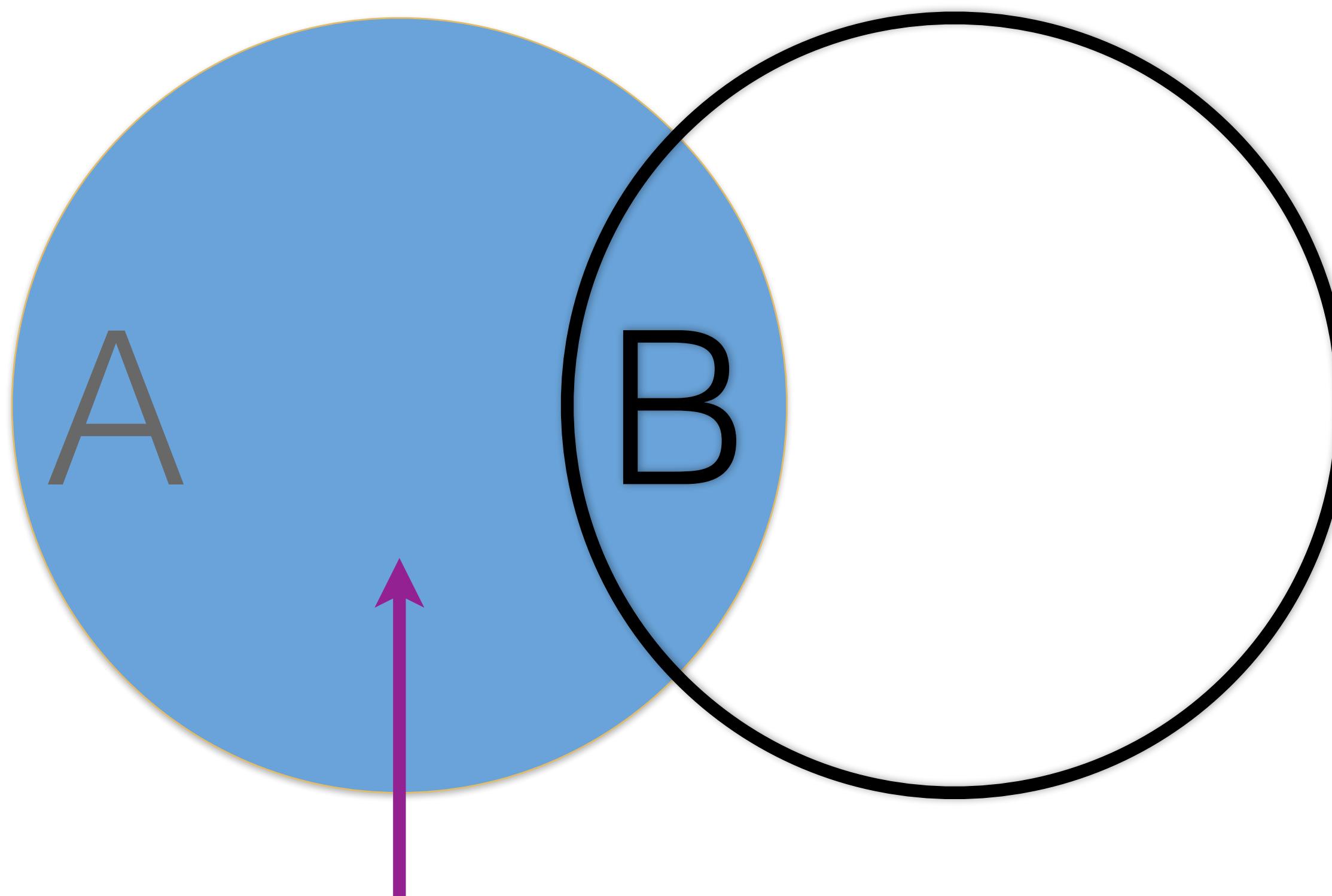
# **Joins**

# Inner Join (Intersection)



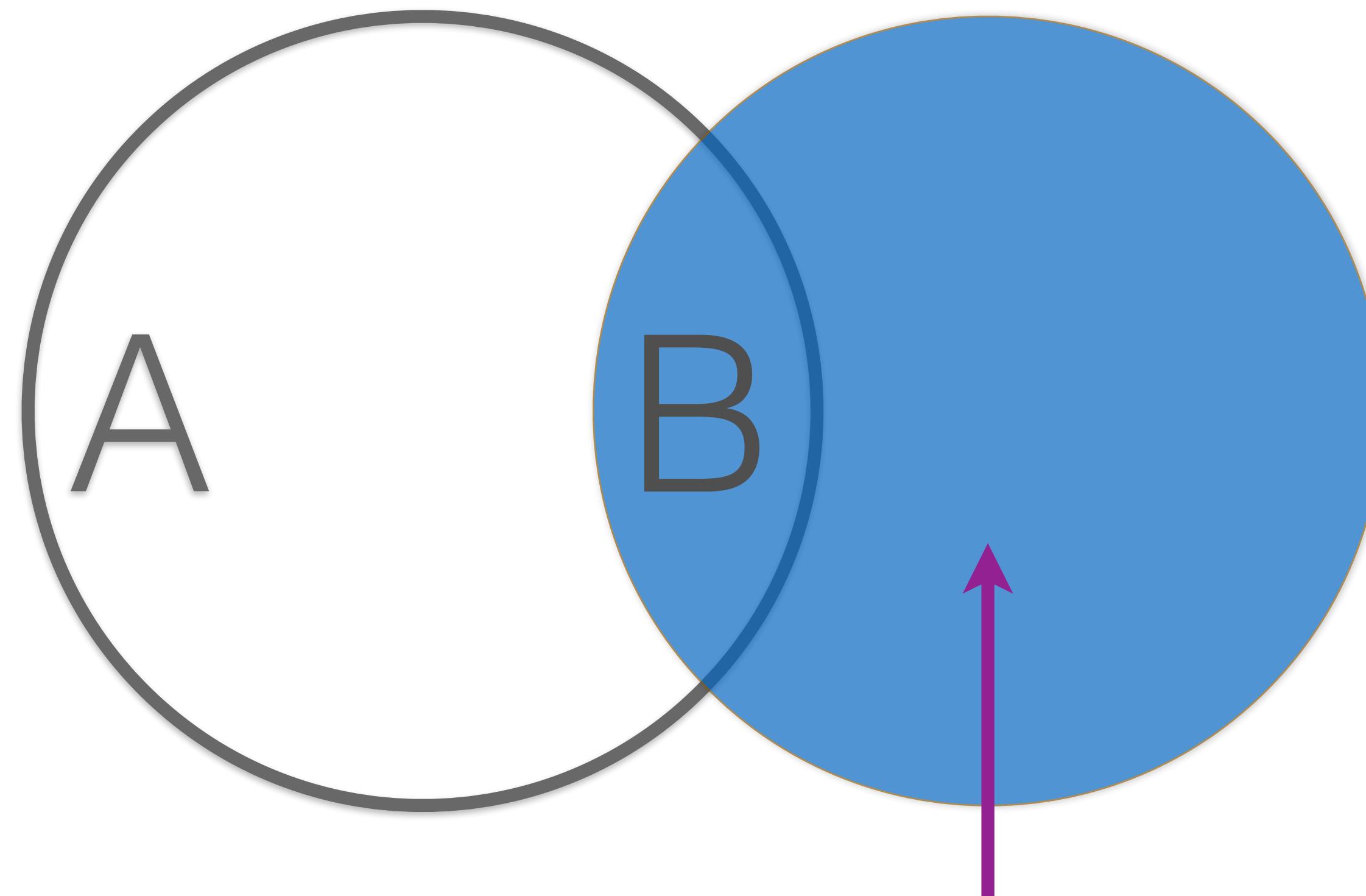
Inner Join

# Left Join



Left Join

# Right Join



Right Join

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

```
pd.merge( leftData,  
        rightData,  
        how="" )
```

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**

# 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

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

	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	

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

# **In-Class Exercise**

Please complete Worksheet 1: Exploring Your Data

# **Module 3: Exploring Data Through Visualization**

# **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.**

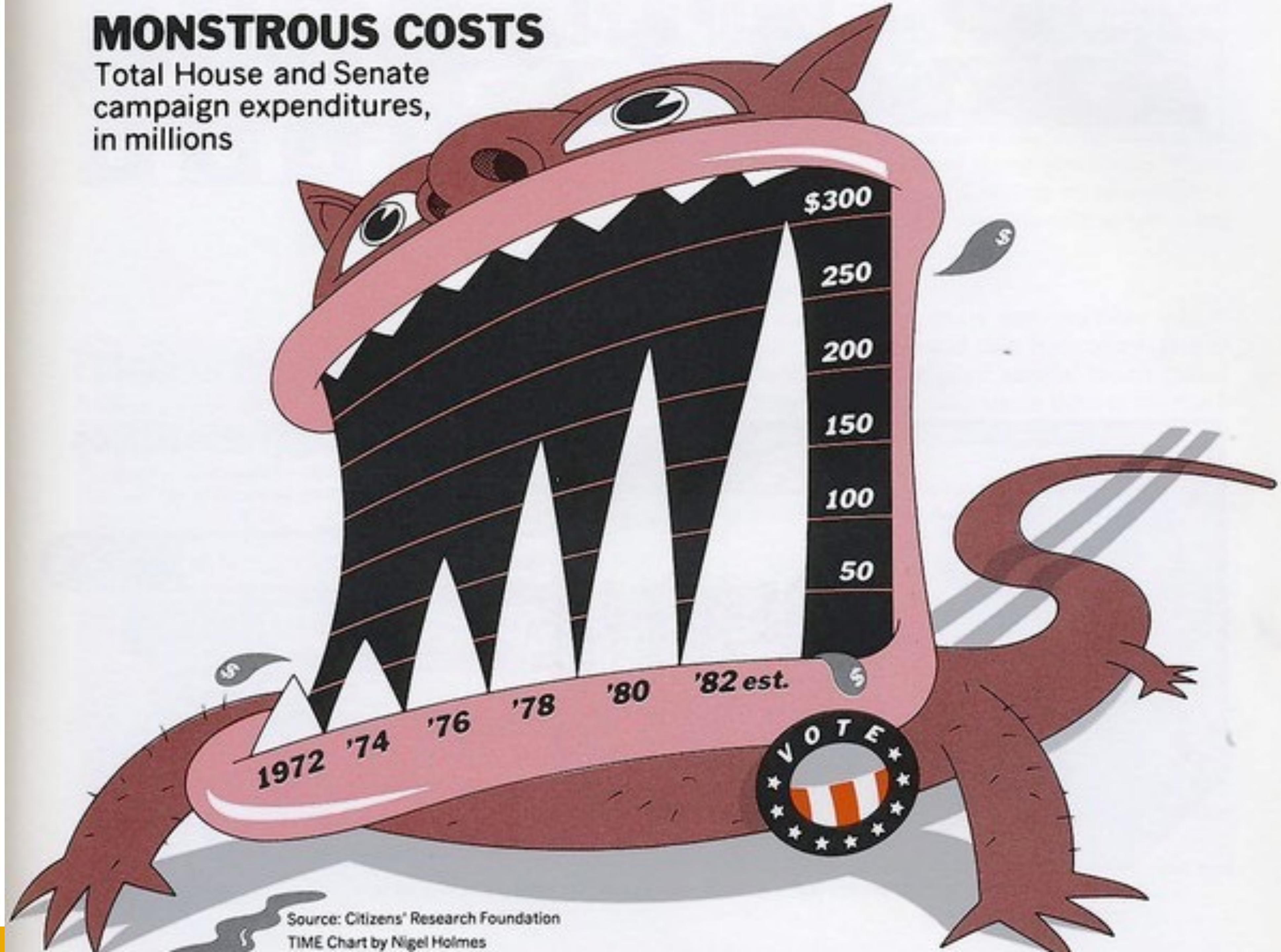
“The greatest value of a picture is when it forces us to notice what we never expected to see.”

*—John Tukey, 1977*

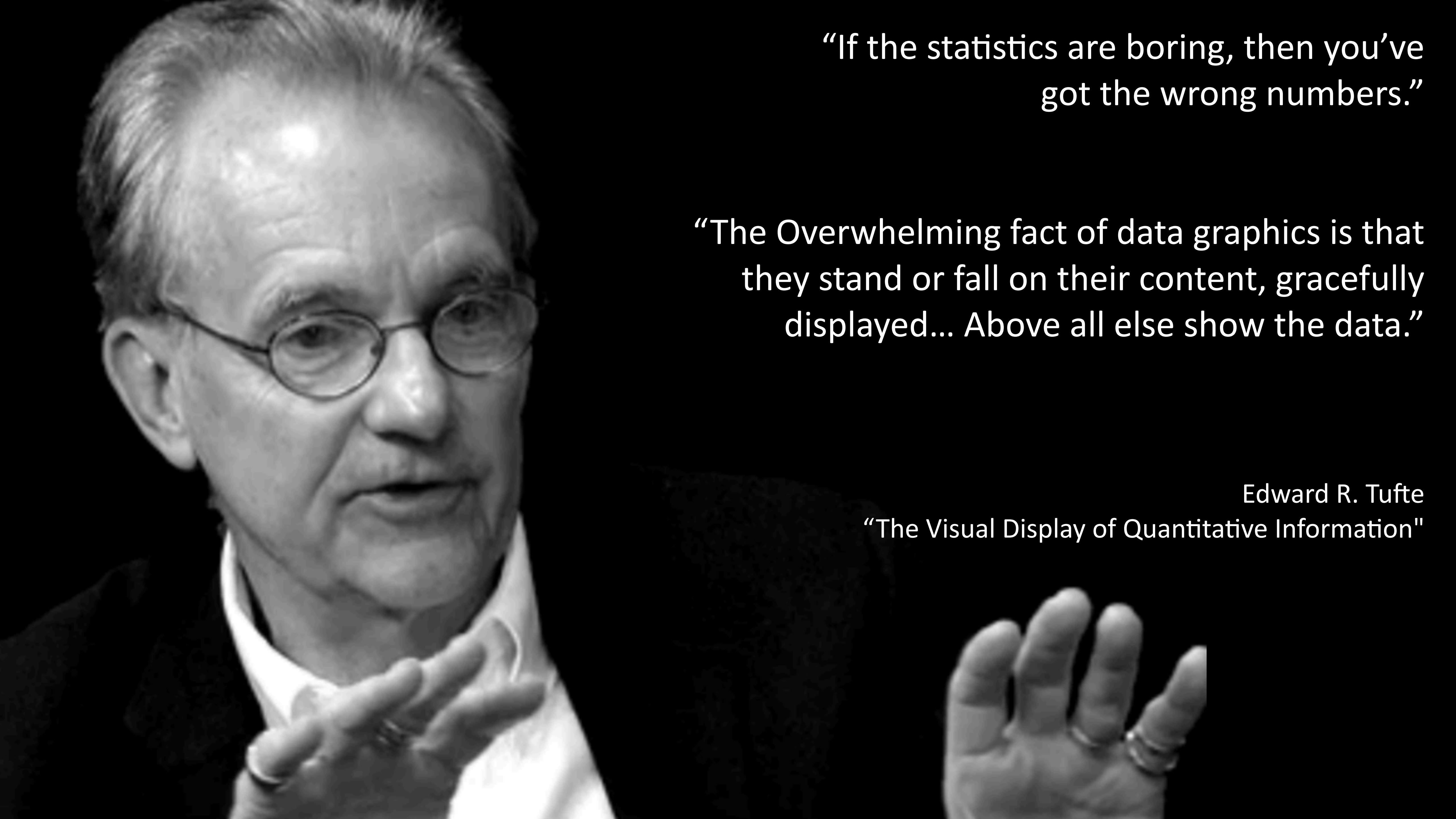
**The power of visualization comes from illustrating relationships, contrasts and comparisons between many different dimensions of data.**

# MONSTROUS COSTS

Total House and Senate campaign expenditures,  
in millions



Source: Citizens' Research Foundation  
TIME Chart by Nigel Holmes



“If the statistics are boring, then you’ve got the wrong numbers.”

“The Overwhelming fact of data graphics is that they stand or fall on their content, gracefully displayed... Above all else show the data.”

Edward R. Tufte

“The Visual Display of Quantitative Information”

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

**Show Comparisons, Contrasts  
and Differences**

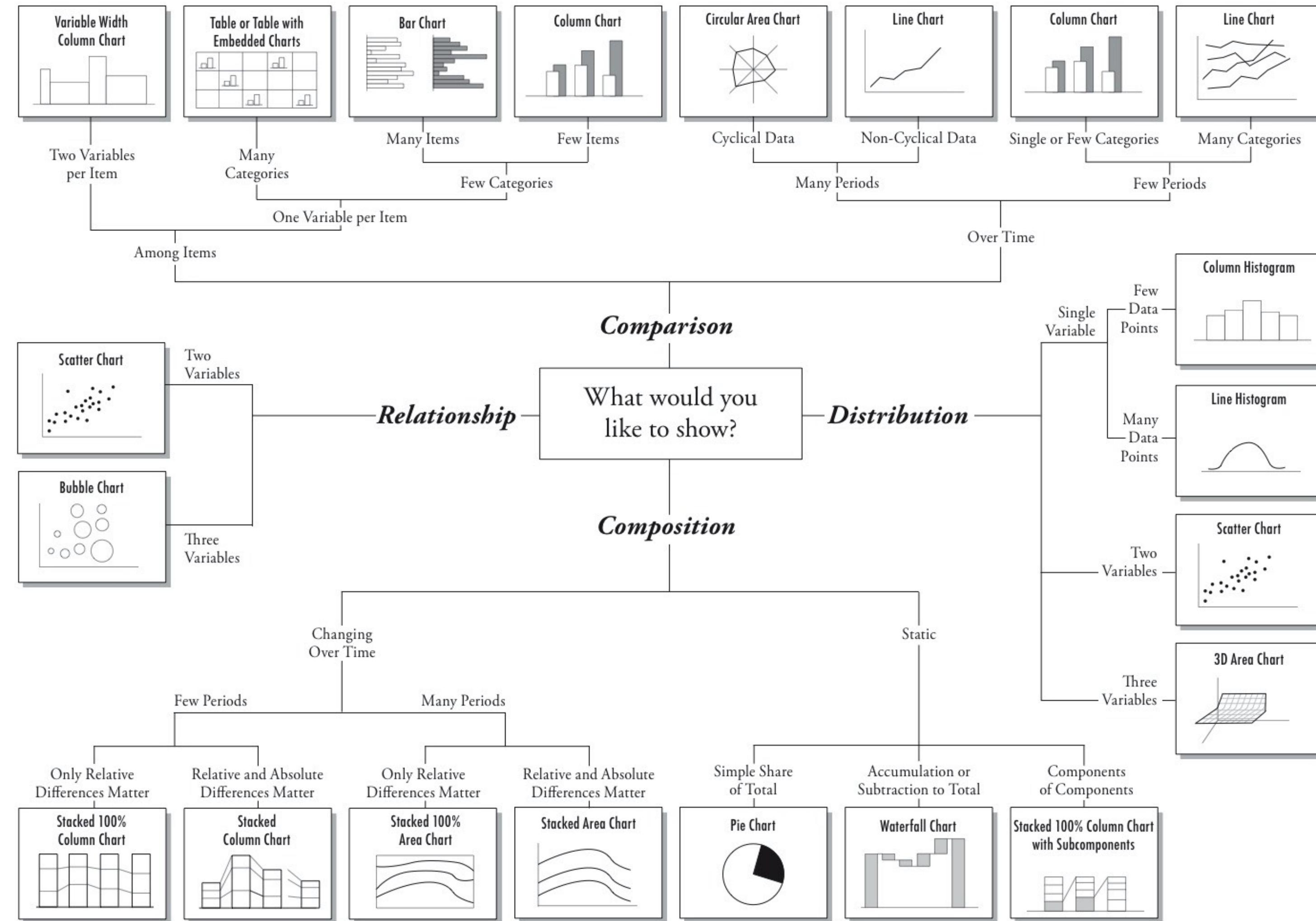
# Show Multivariate Data

**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



© 2006 A. Abela — a.v.abela@gmail.com

# Visualization Goals

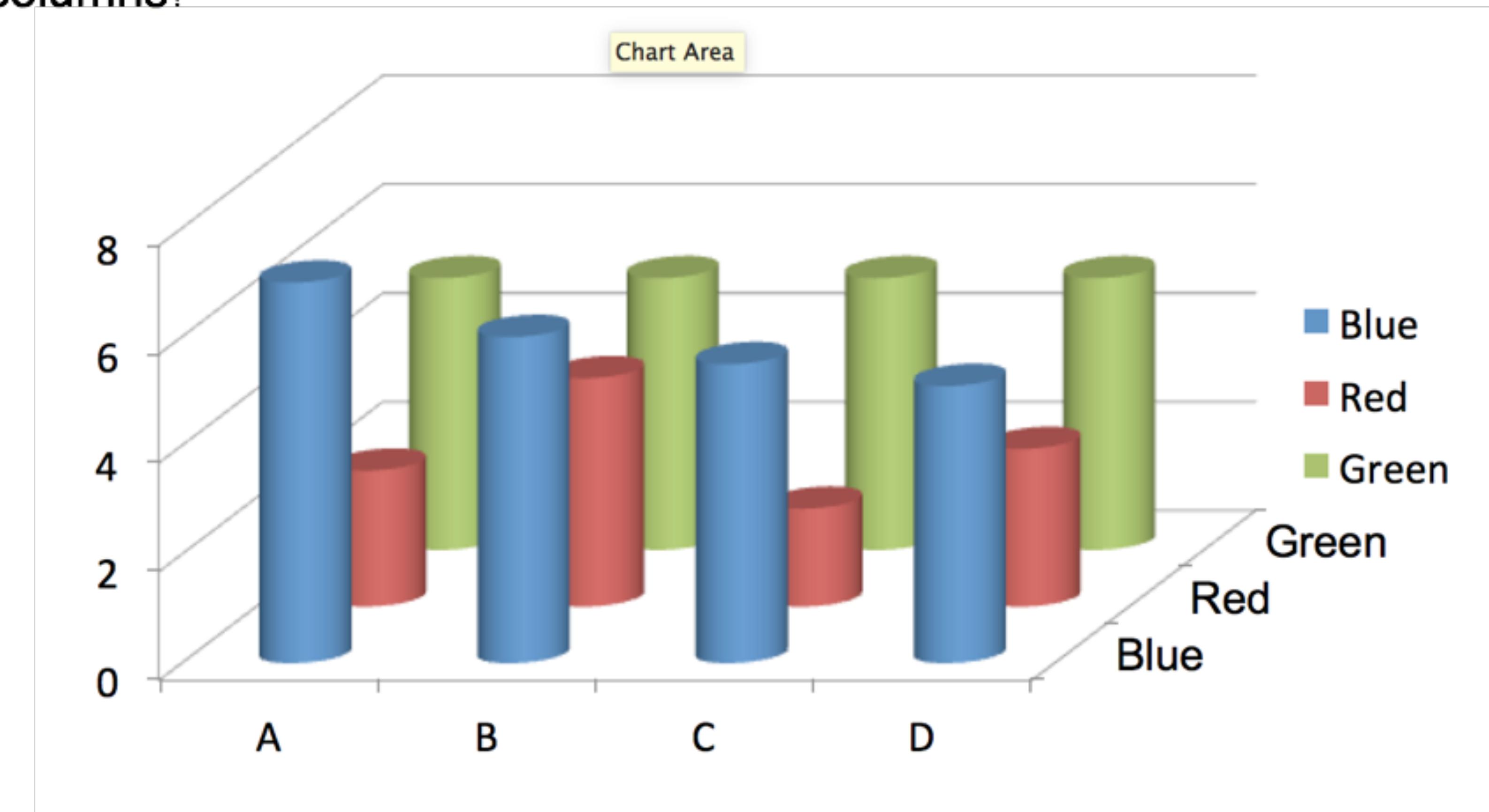
- Analyze
- Explore
- Assess
- Determine
- Decide
- Communicate
- Explain
- Present
- Prove
- Persuade

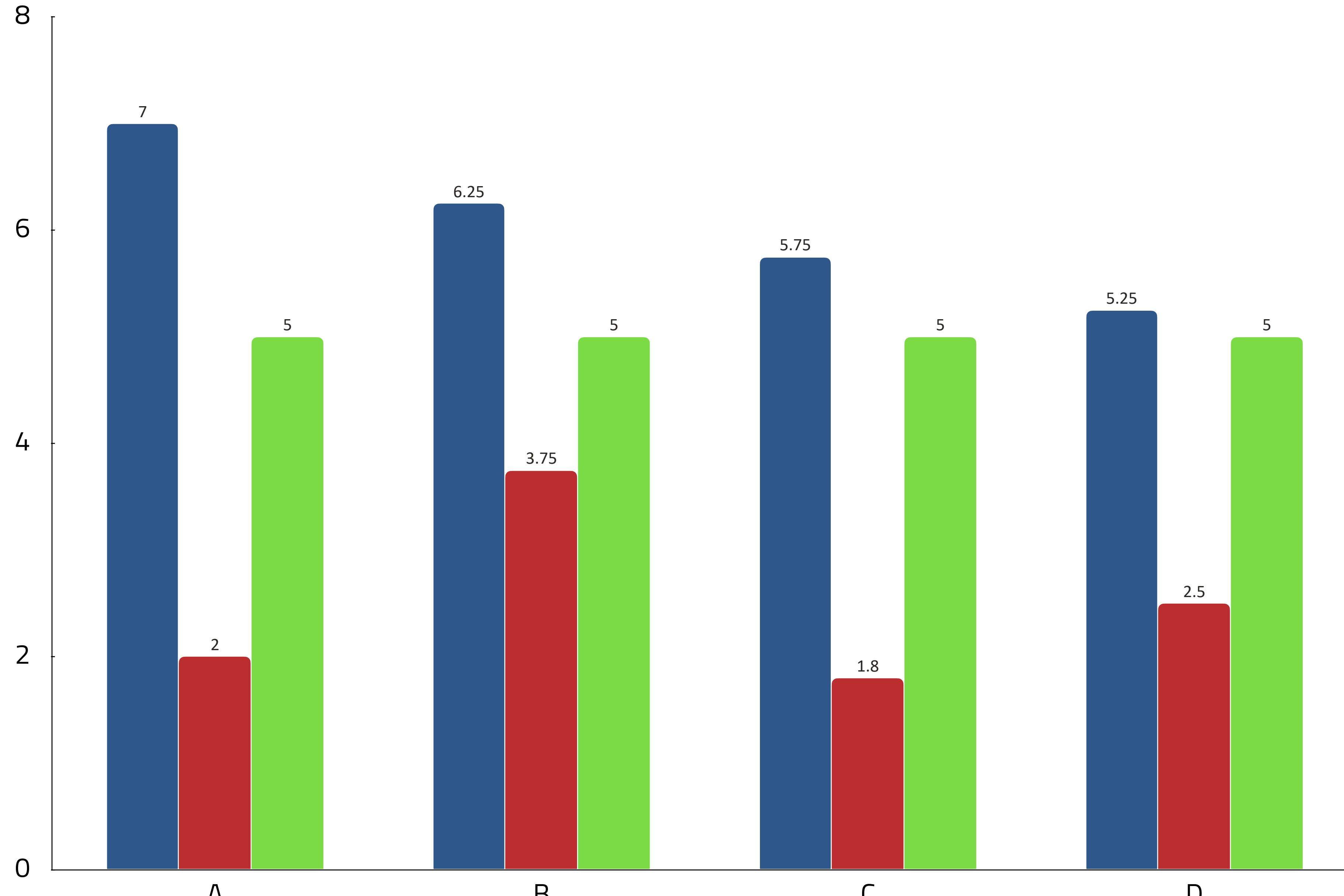
# Elements of Good Visualizations

1. Graphical Integrity
2. Simple
3. Proper Display
4. Proper Color
5. Tells a story

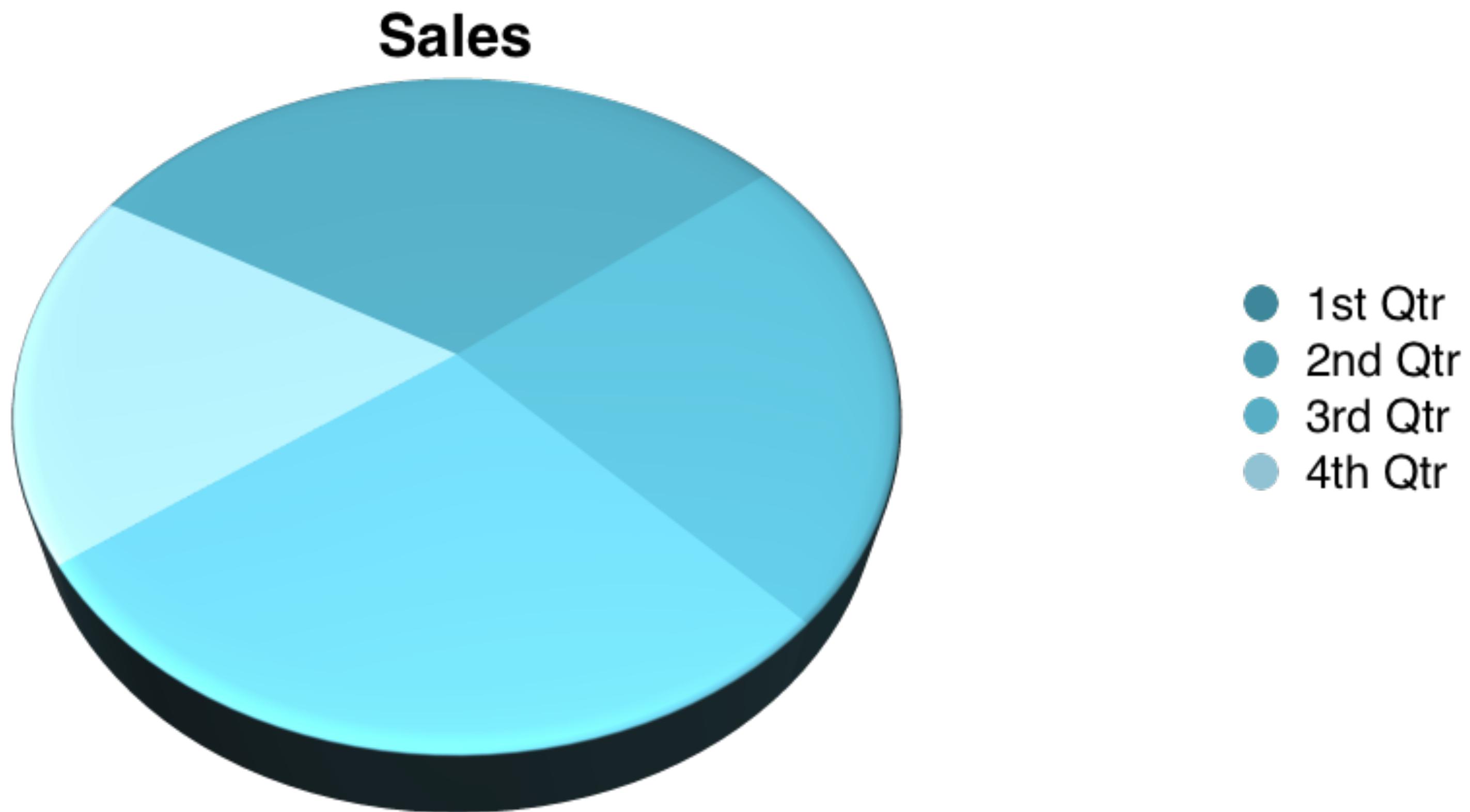
# Proper Display

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?

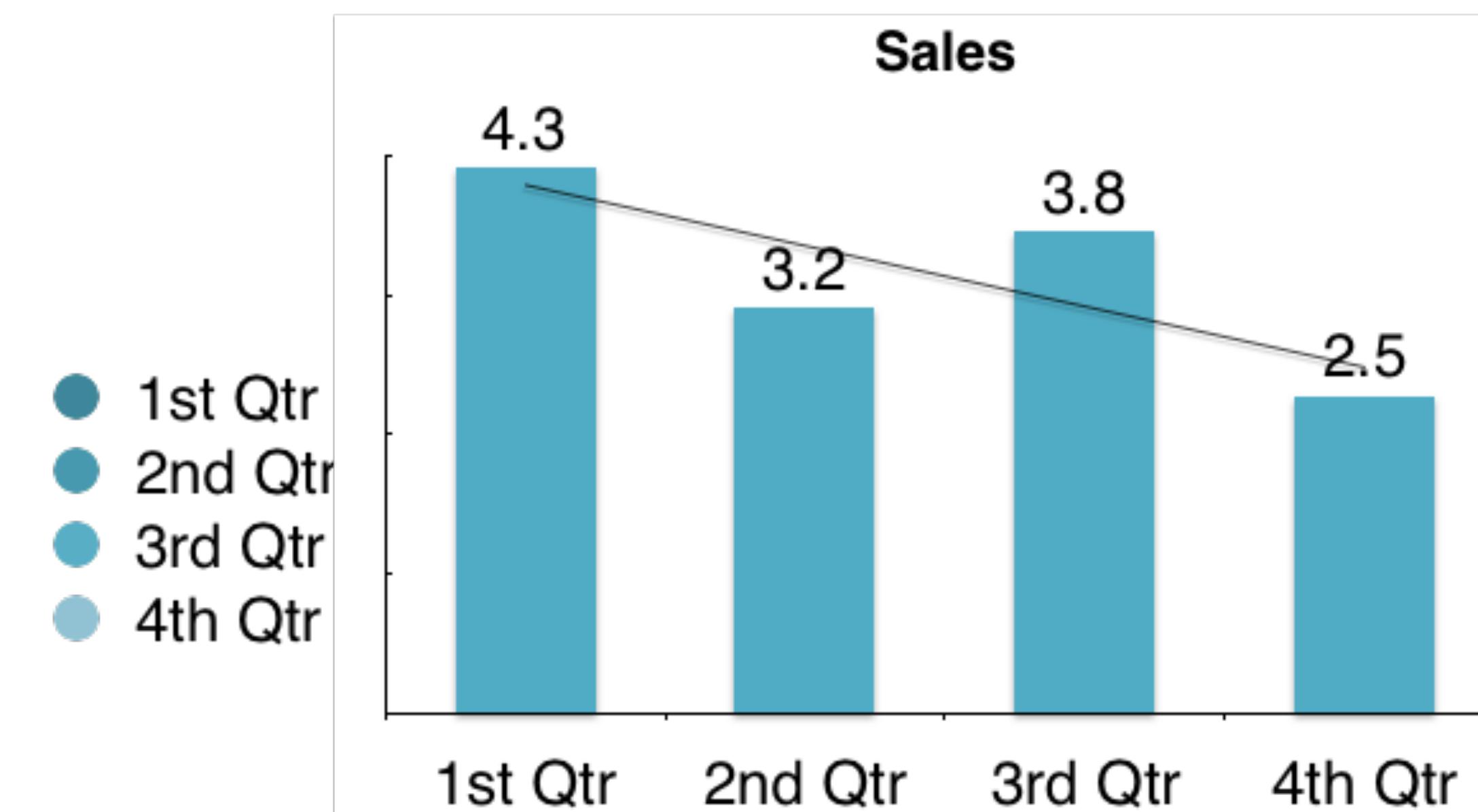




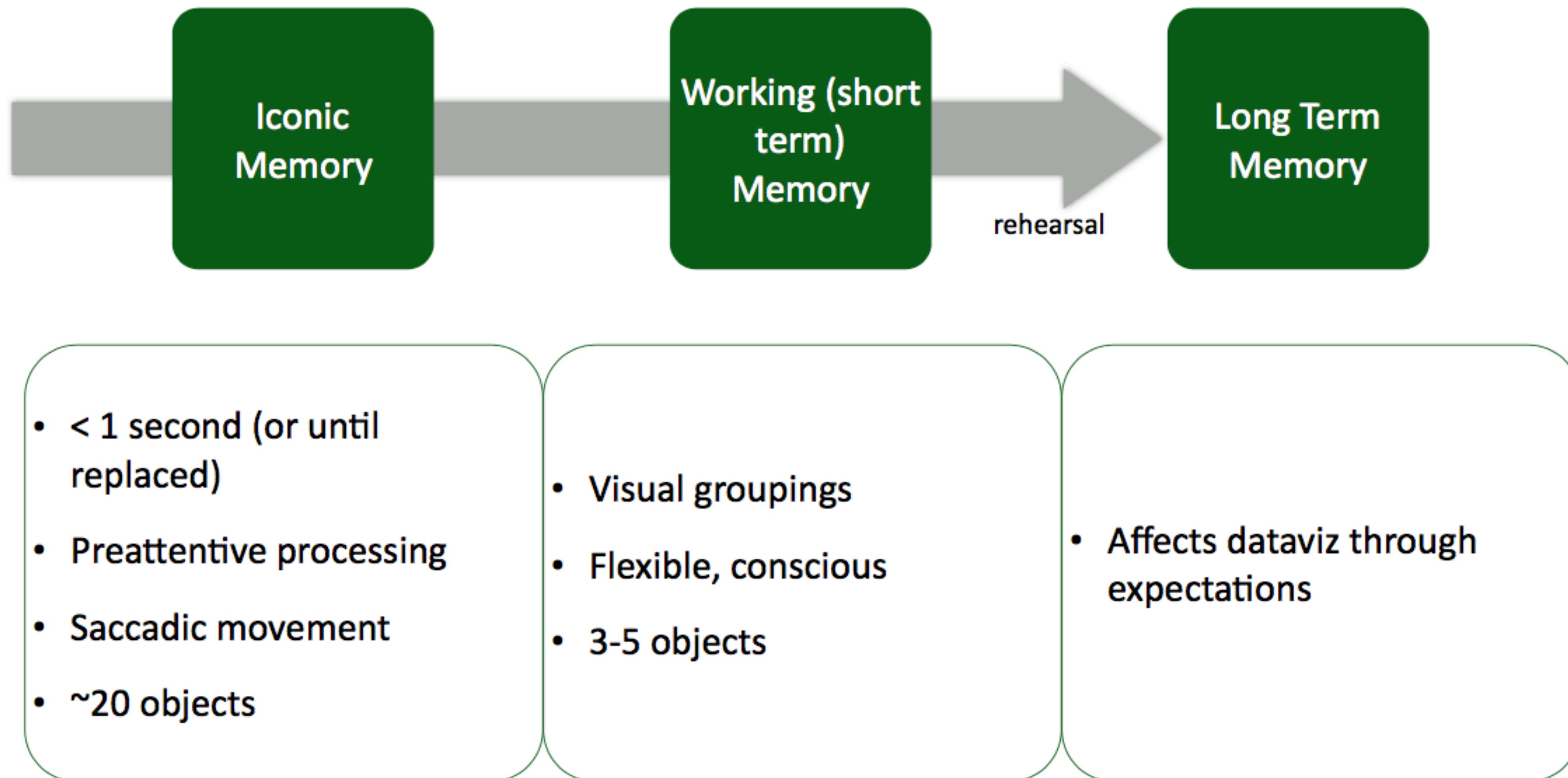
# Proper Display



# Proper Display



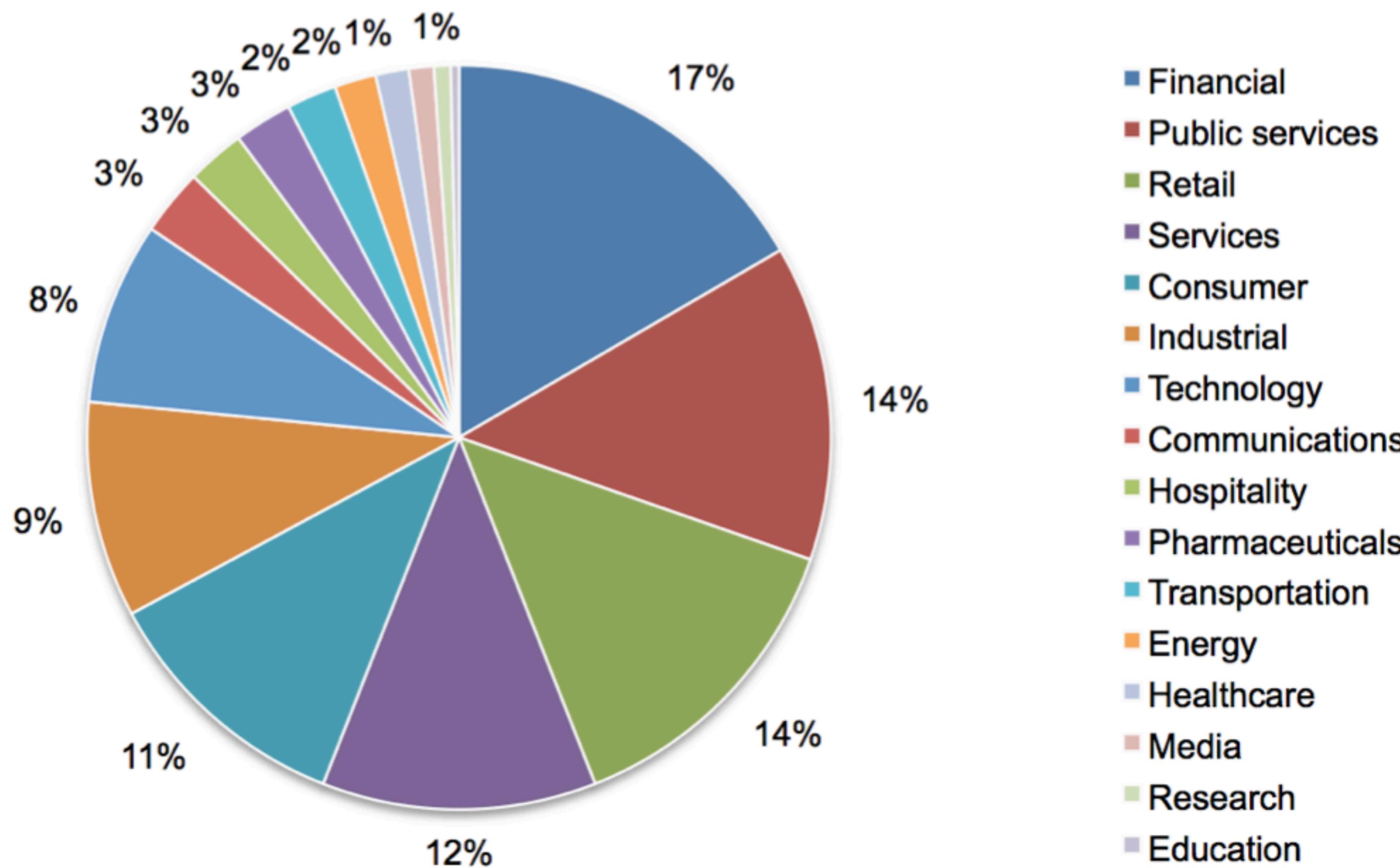
# Visual Processing System



# Overworking Visual Memory

**Figure 20. Distribution of the benchmark sample by industry segment**

Consolidated (n = 277 organizations)

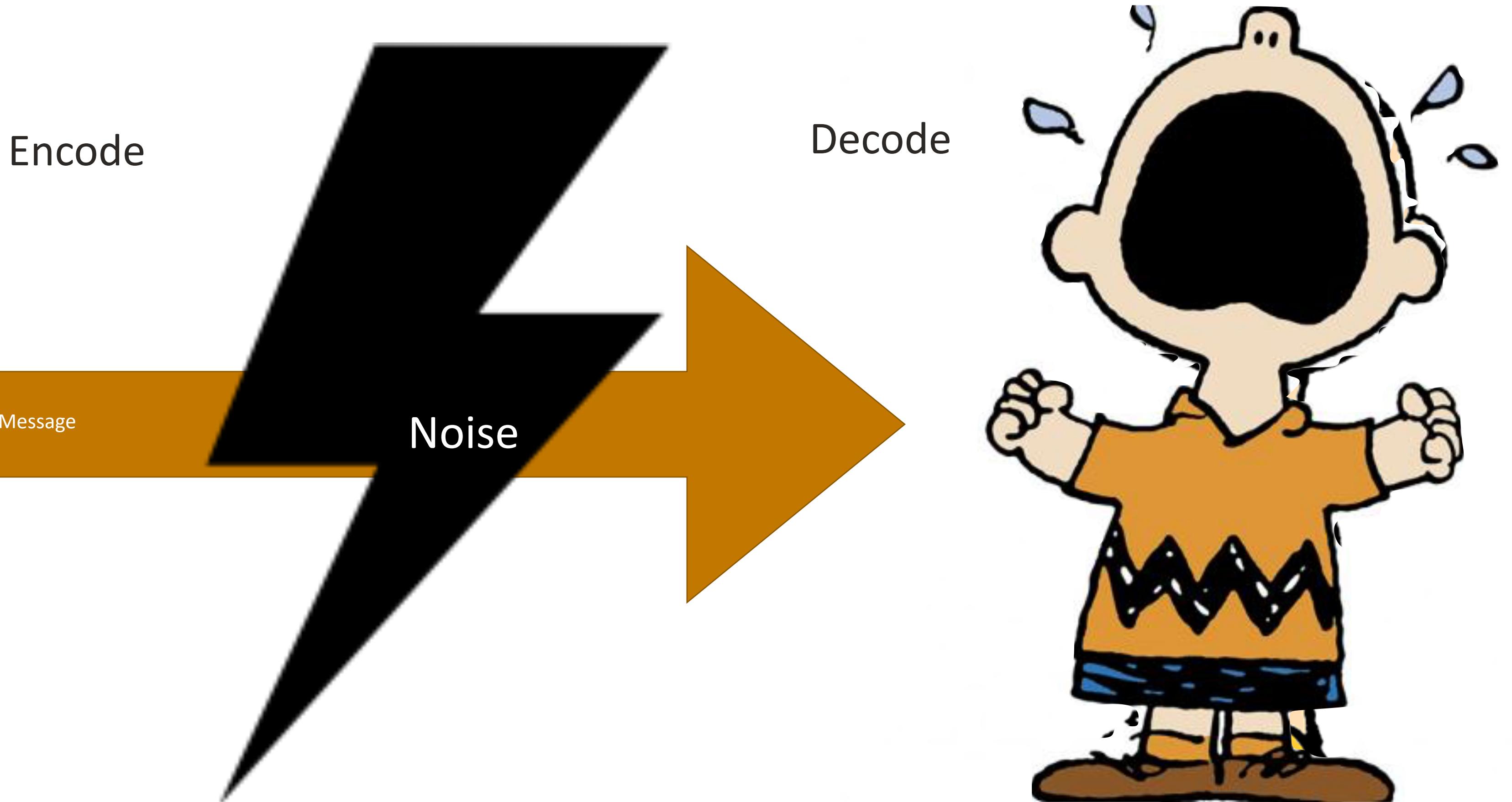


**Brainpower used  
for decoding**

**Brainpower remaining  
for understanding**

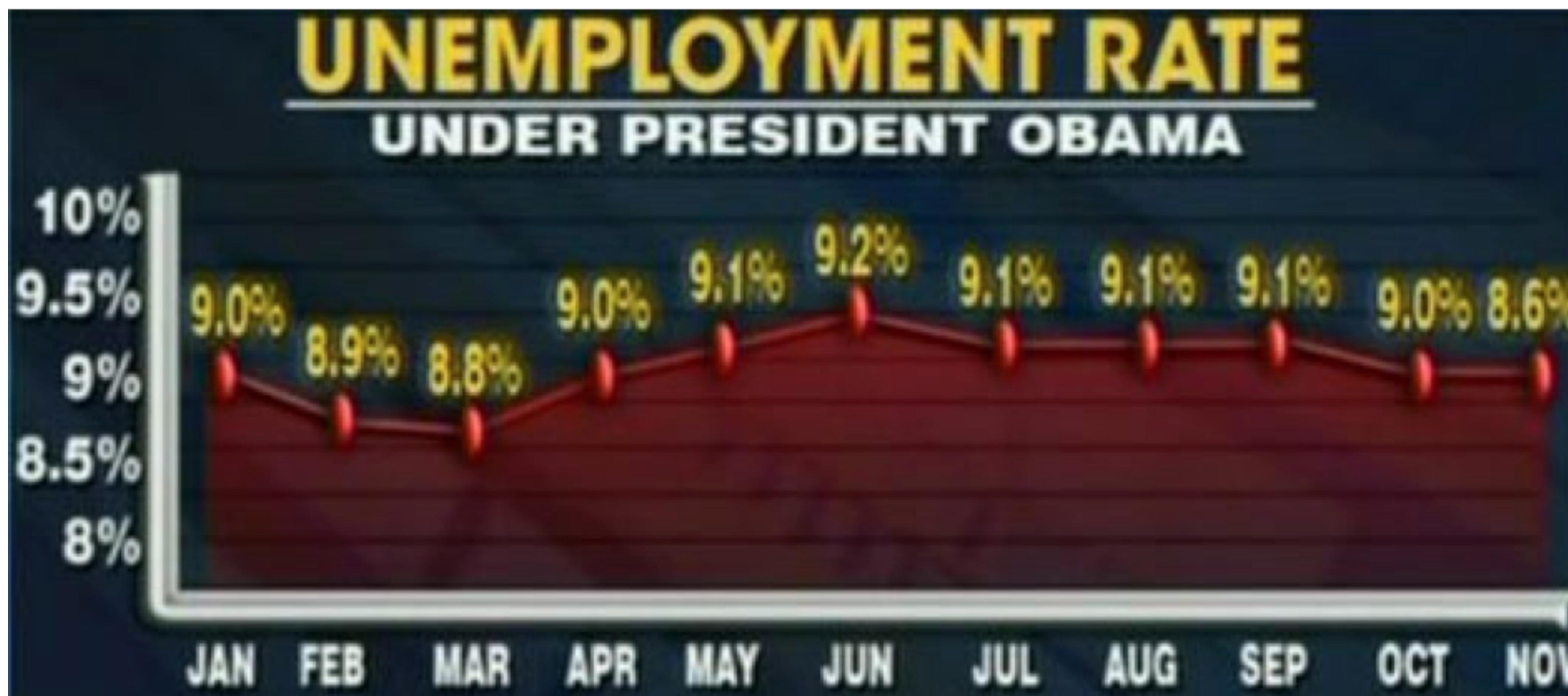


**Total brainpower available**

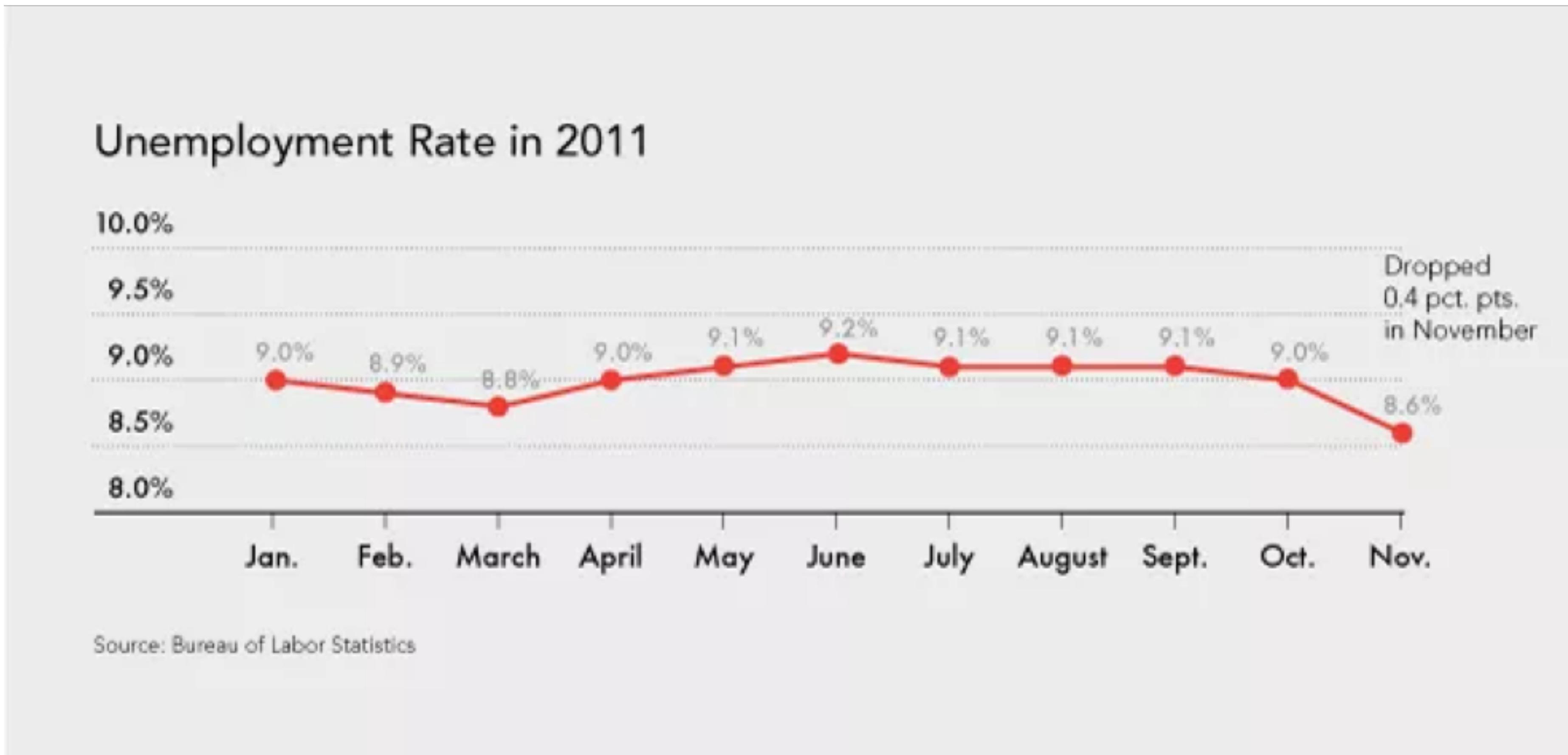


# The Bad...

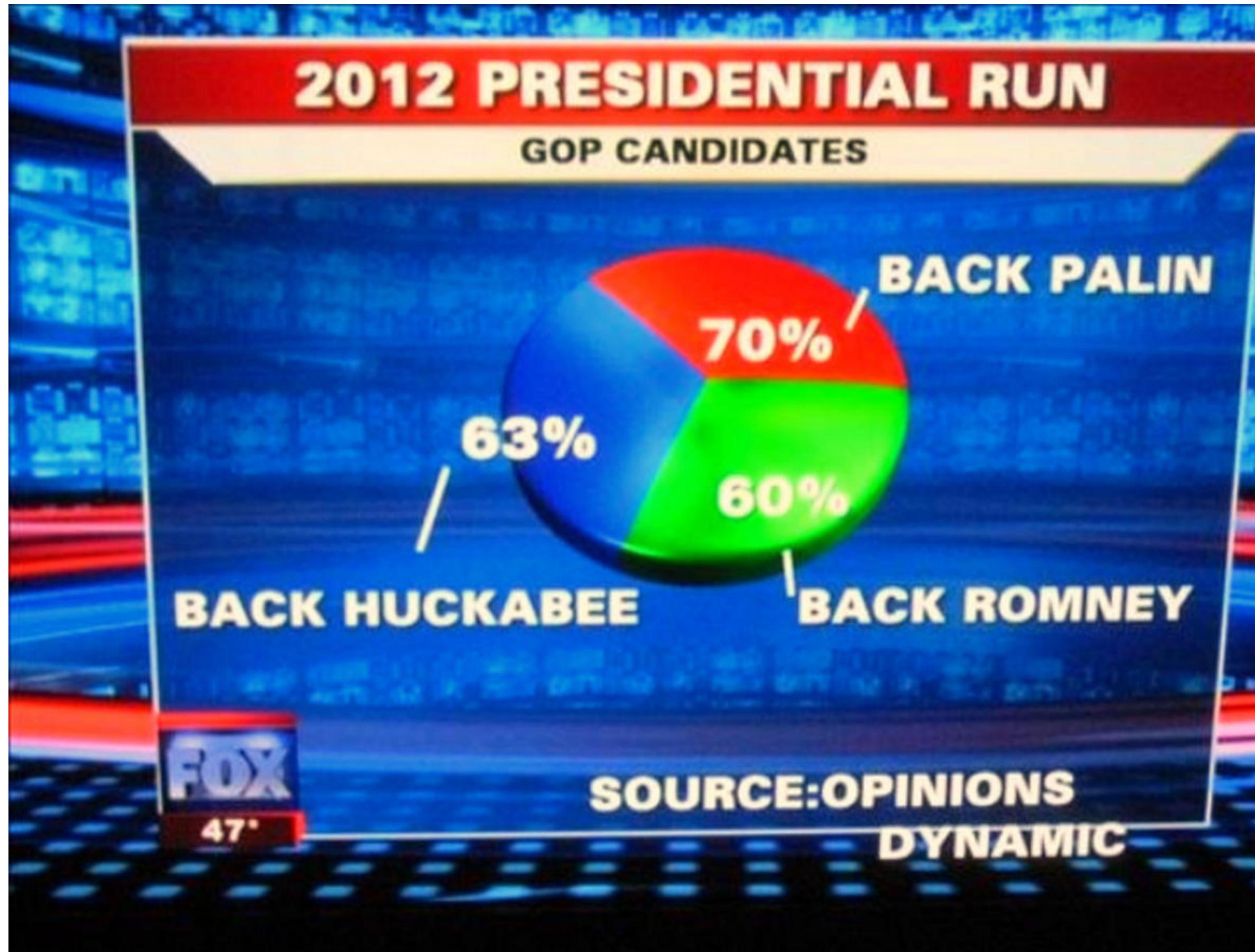
# Graphical Integrity



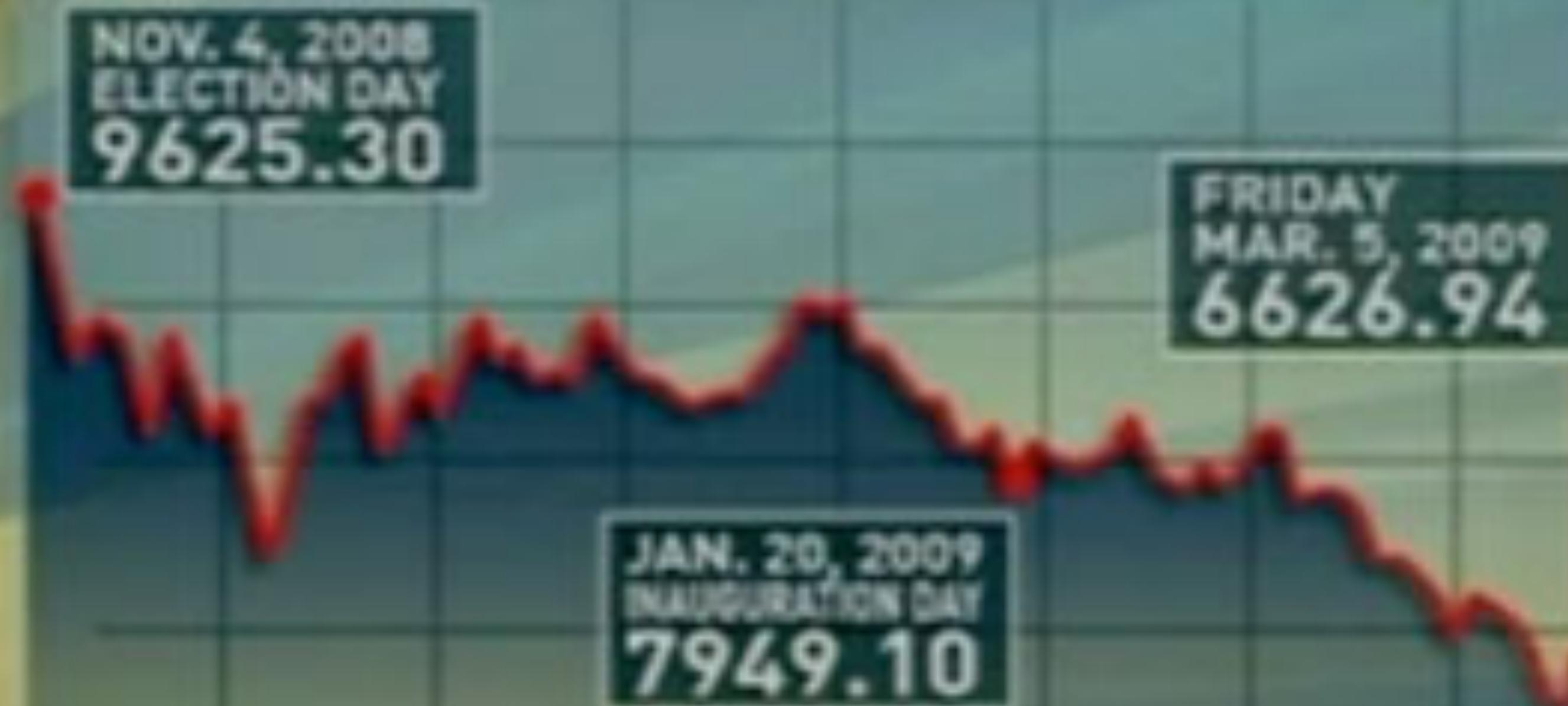
# Graphical Integrity



# Graphical Integrity?



## STOCK MARKET SLIDE

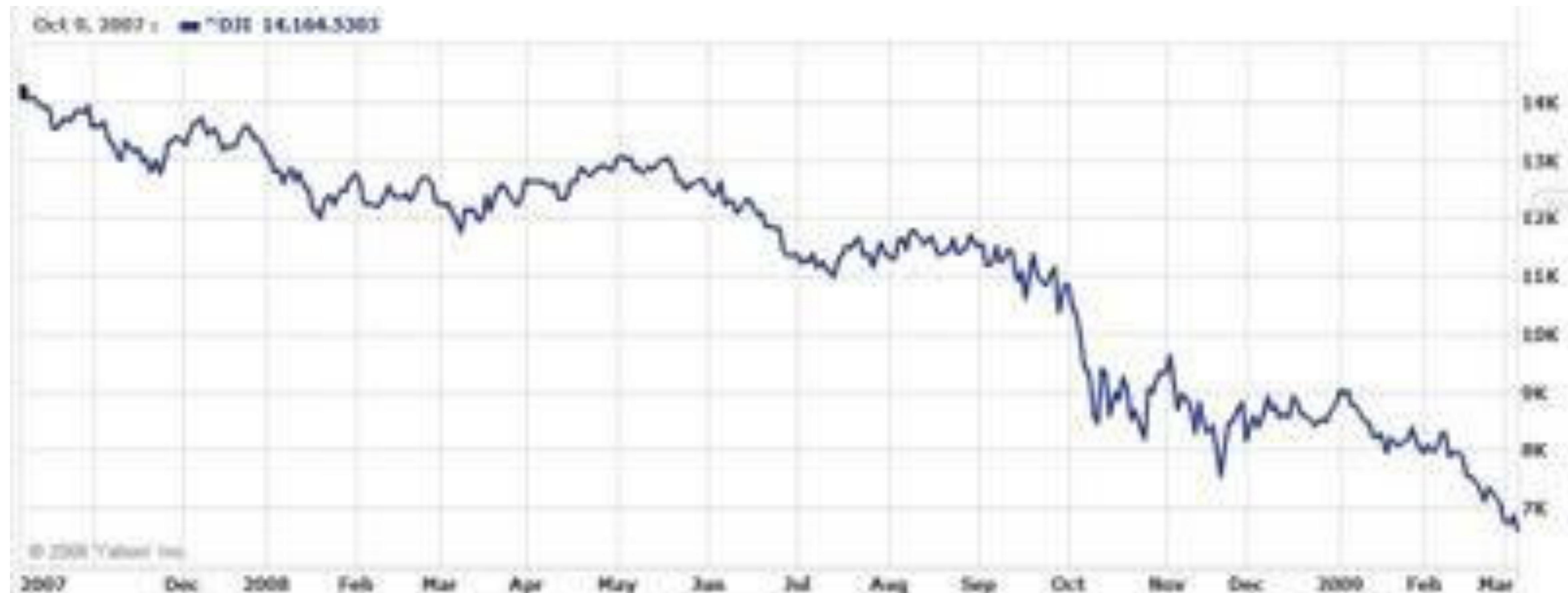


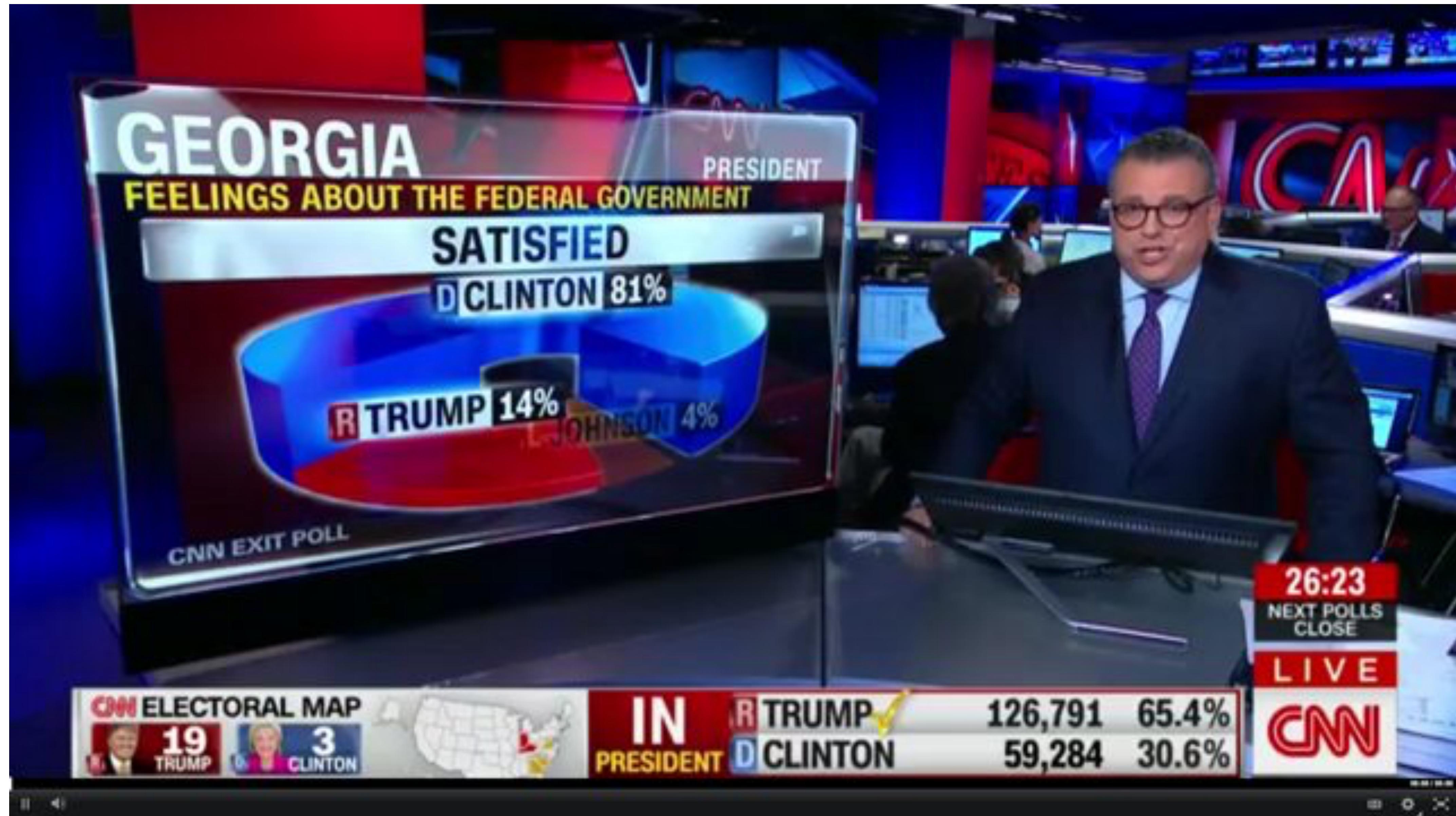
SOURCE: BUSINESSWEEK

6:31 CT

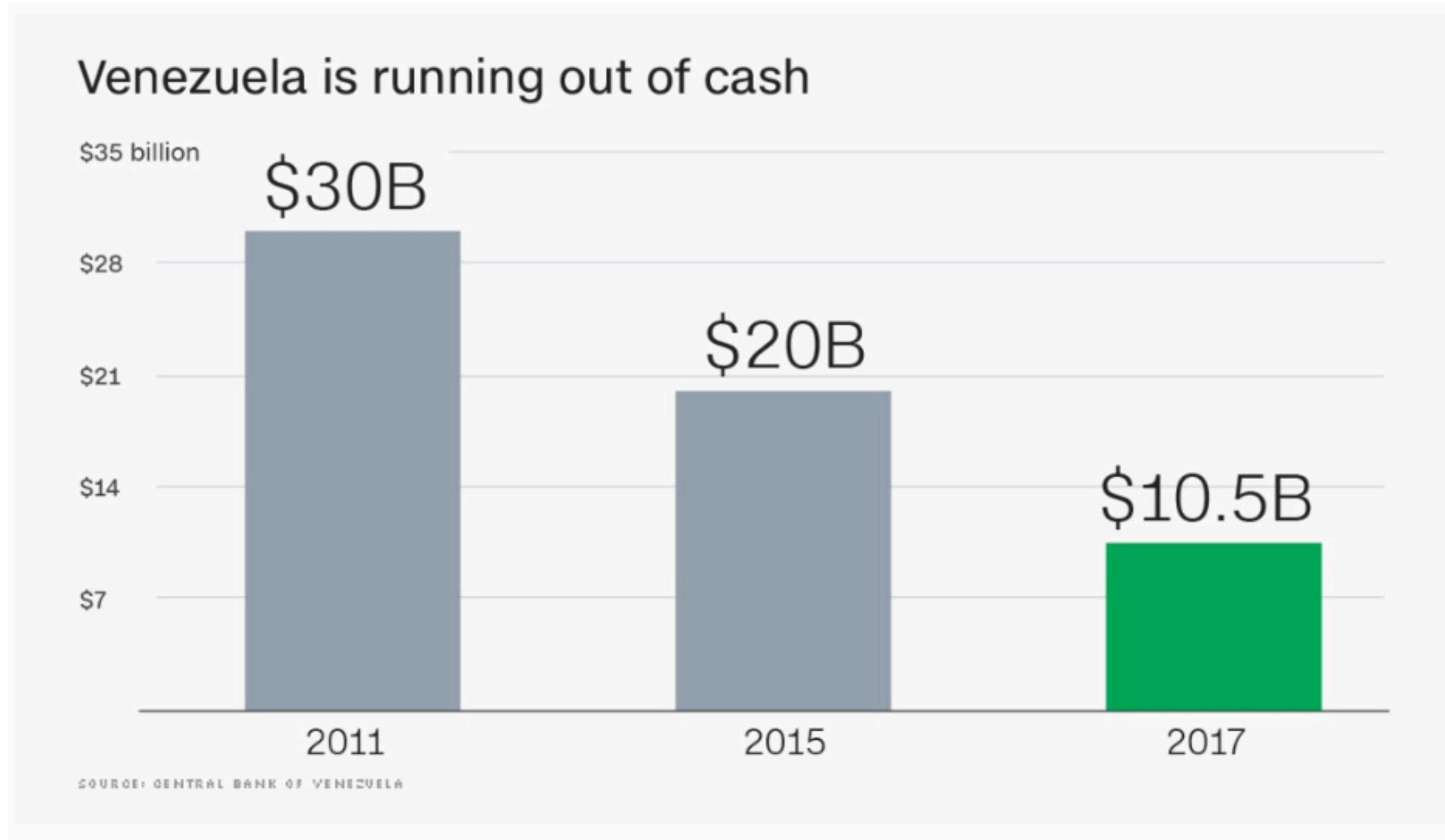
MSNBC

"A PRESIDENTT. HAS MINDD ABBAS-S IT WILL TAKE FFFFCCT .0

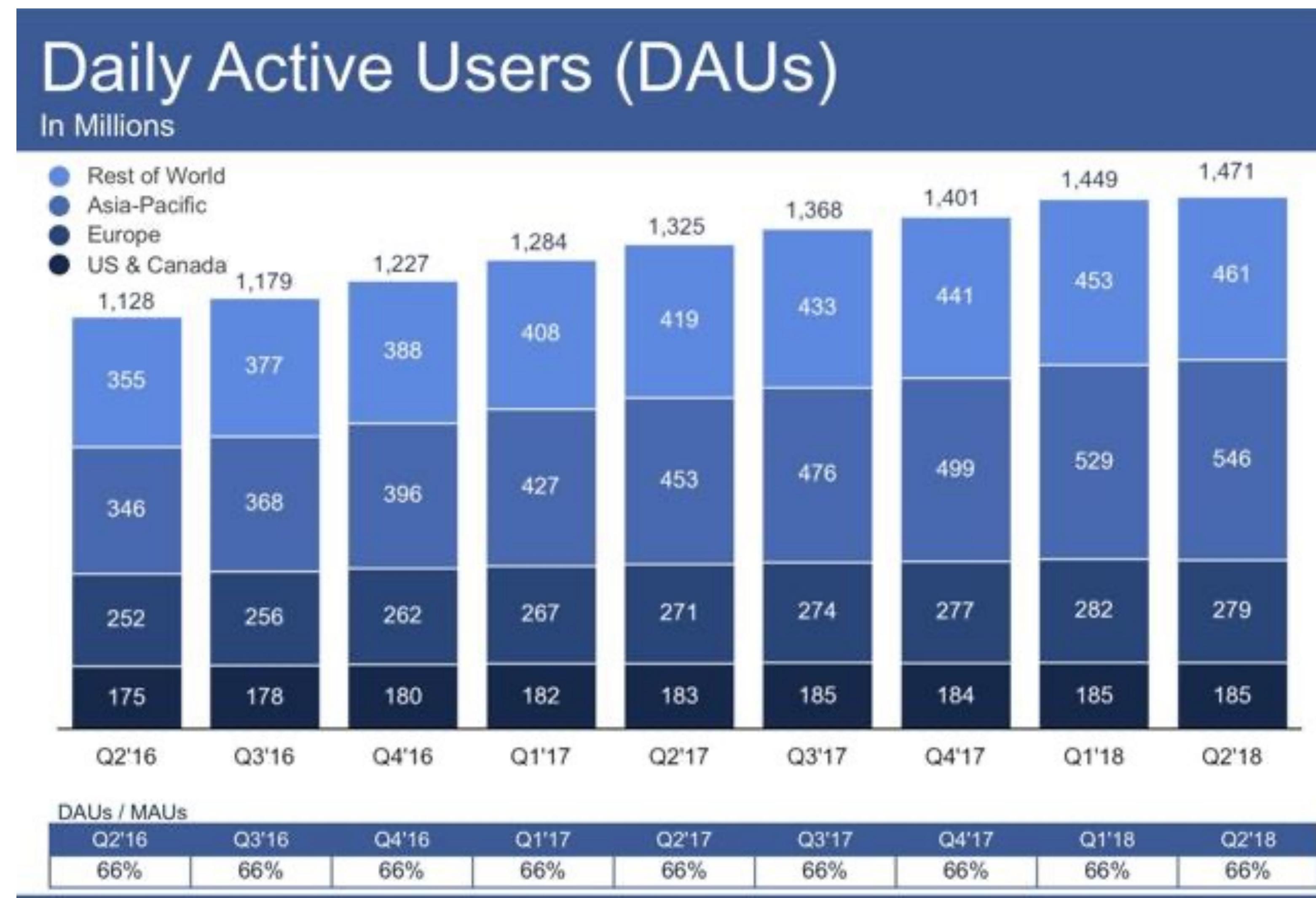




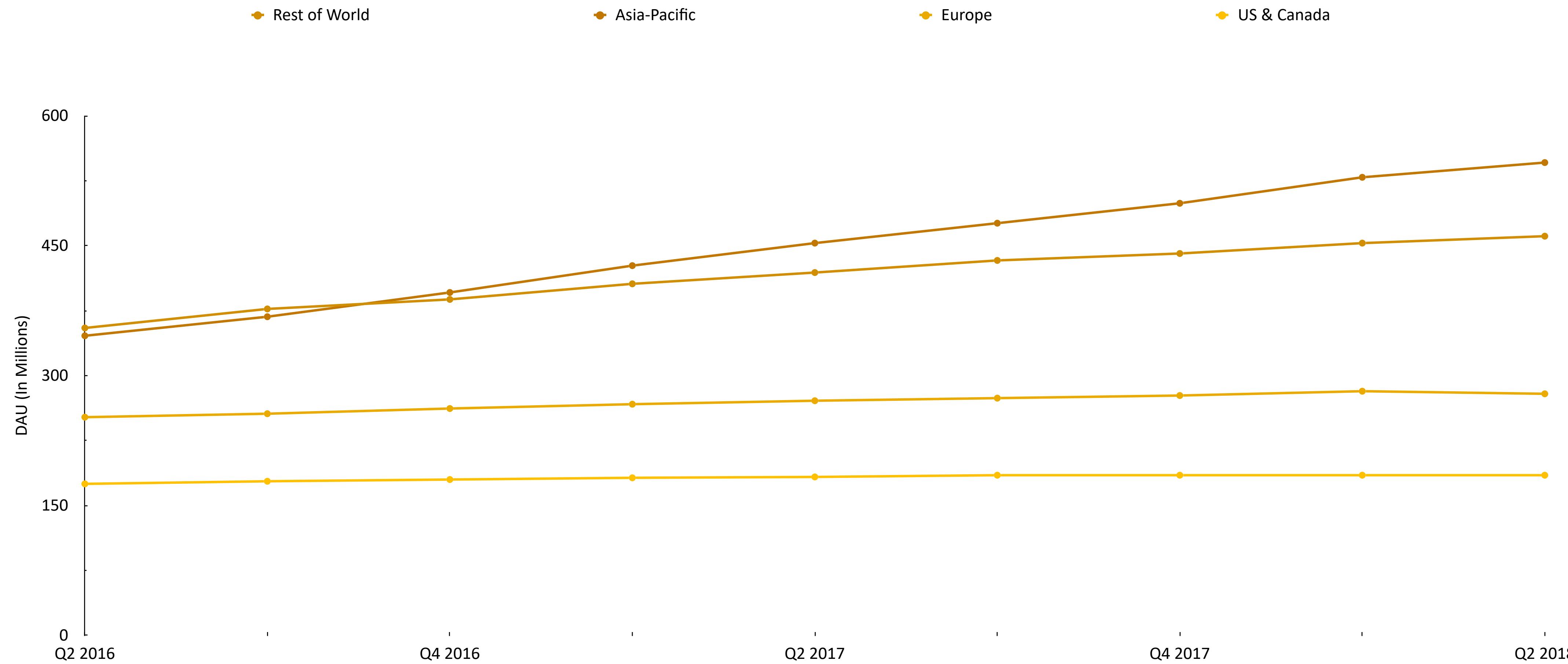
# Graphical Integrity



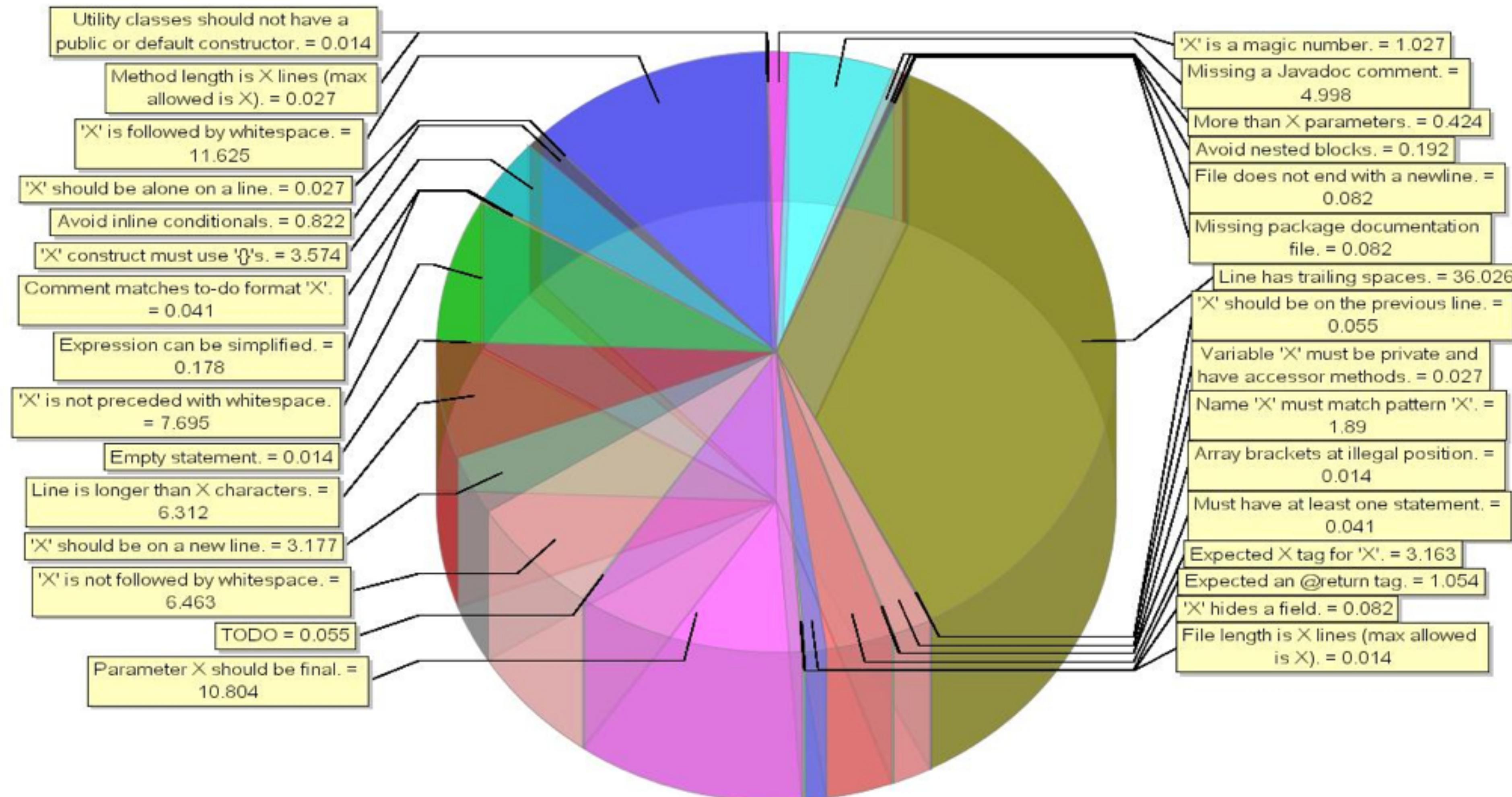
# Proper Display



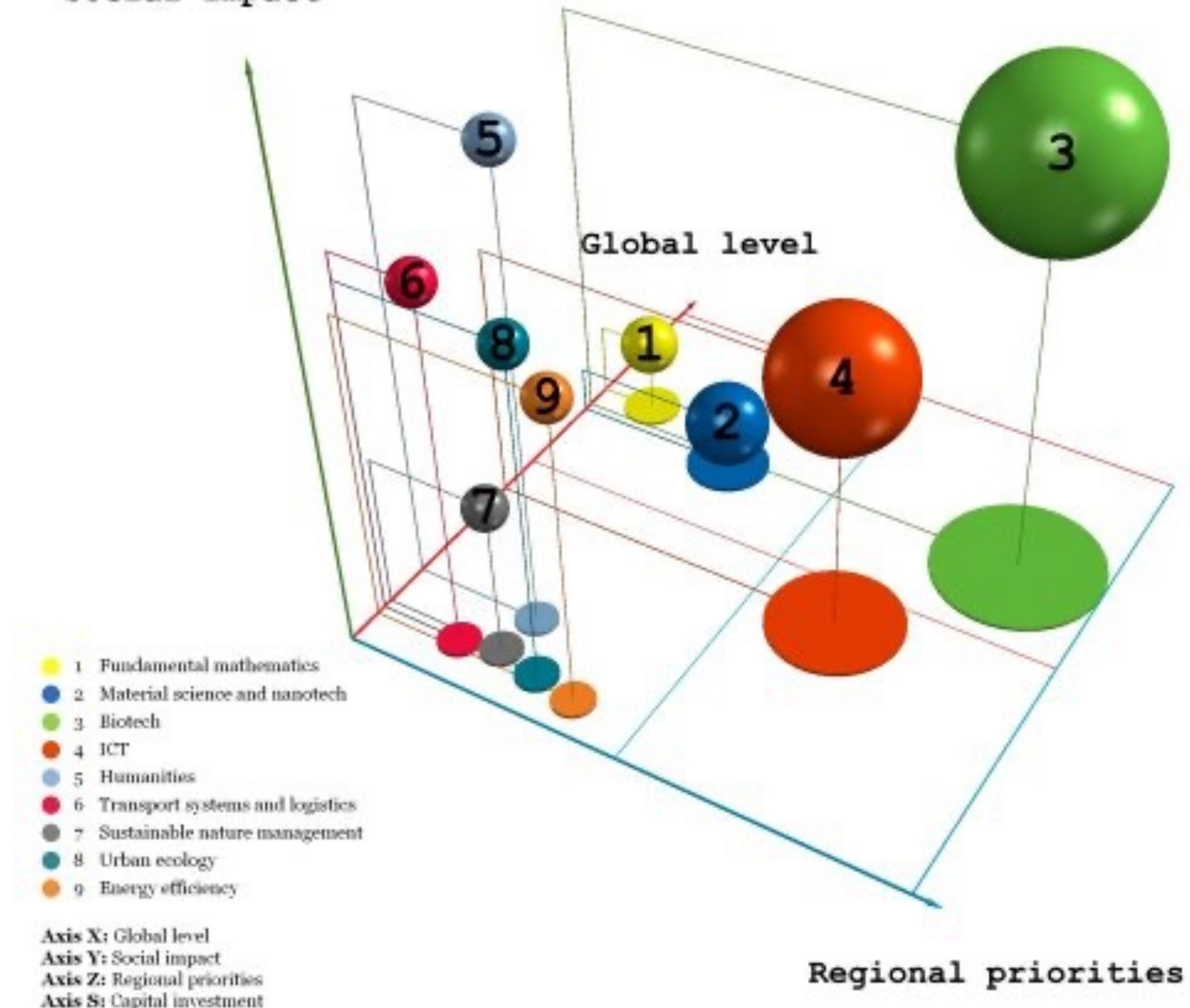
# Proper Display



# Simple

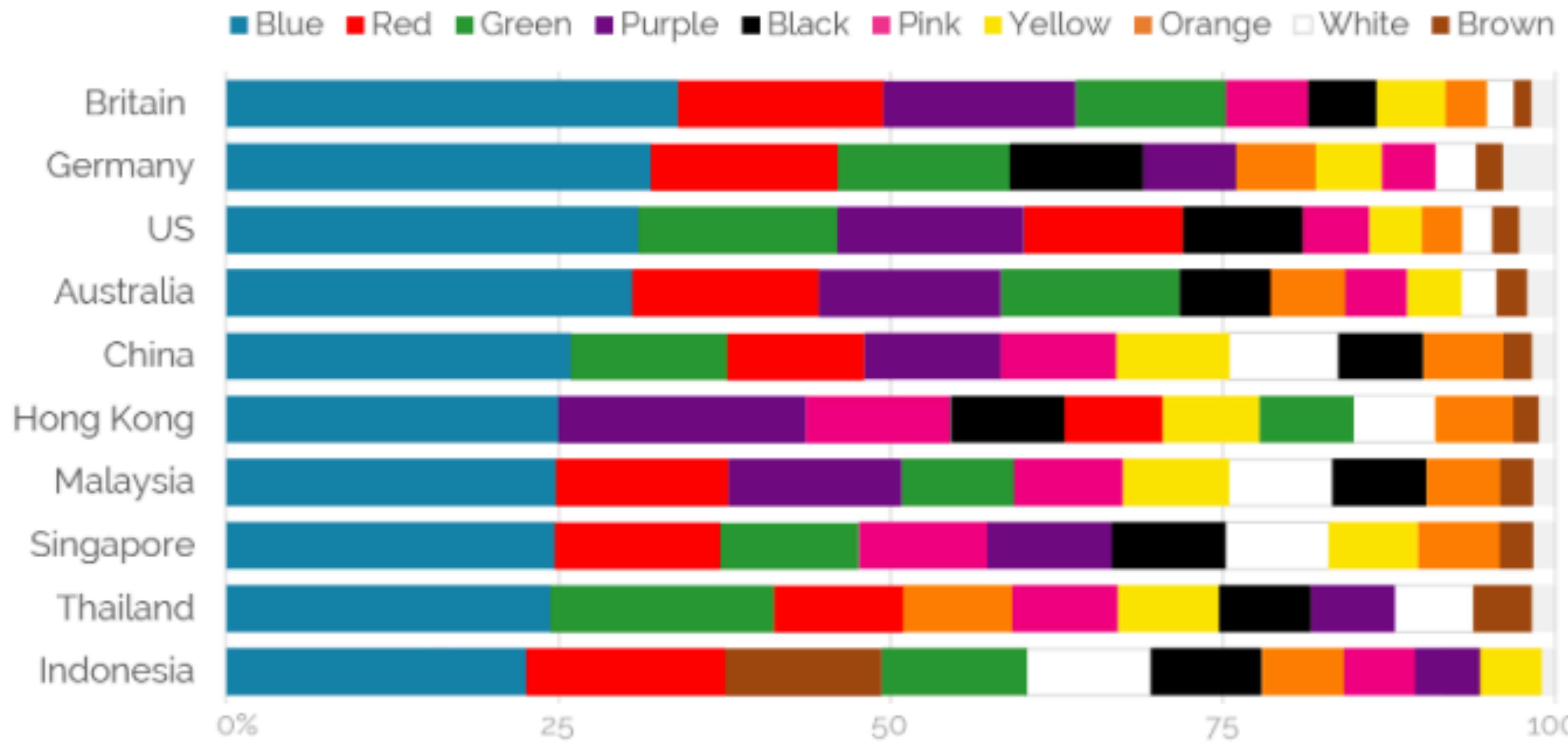


## Social impact

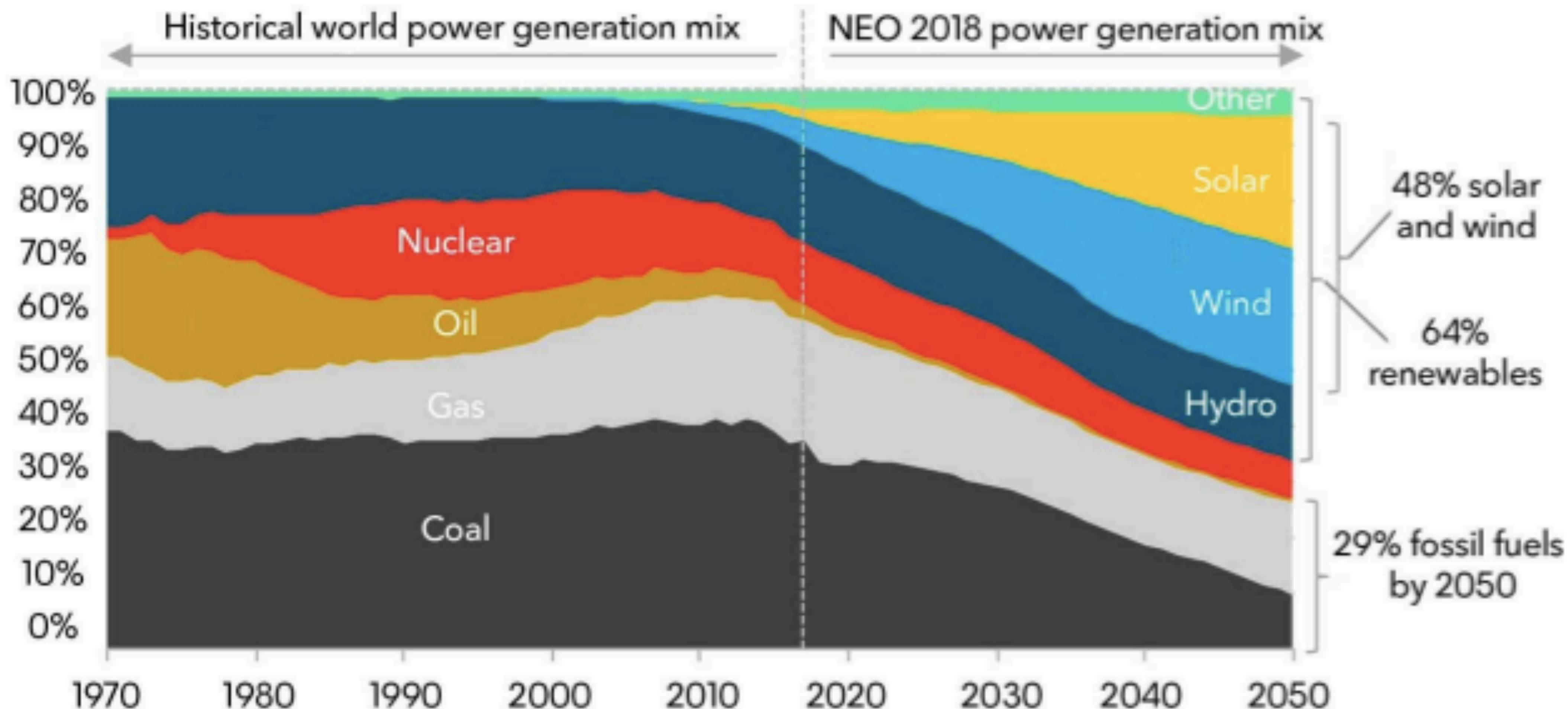


## Blue planet

Which one of the colors listed below do you like the most?

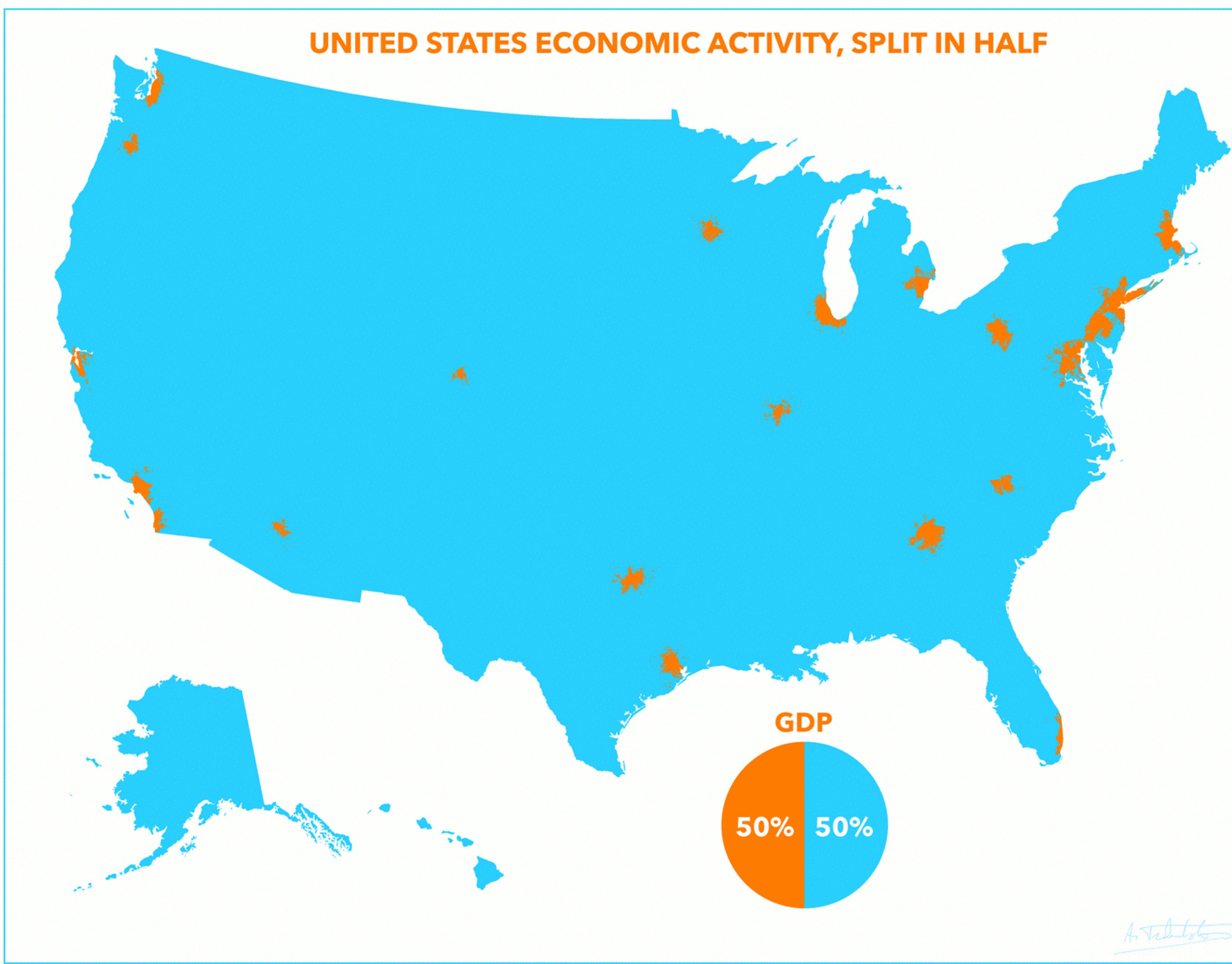


## Power generation mix

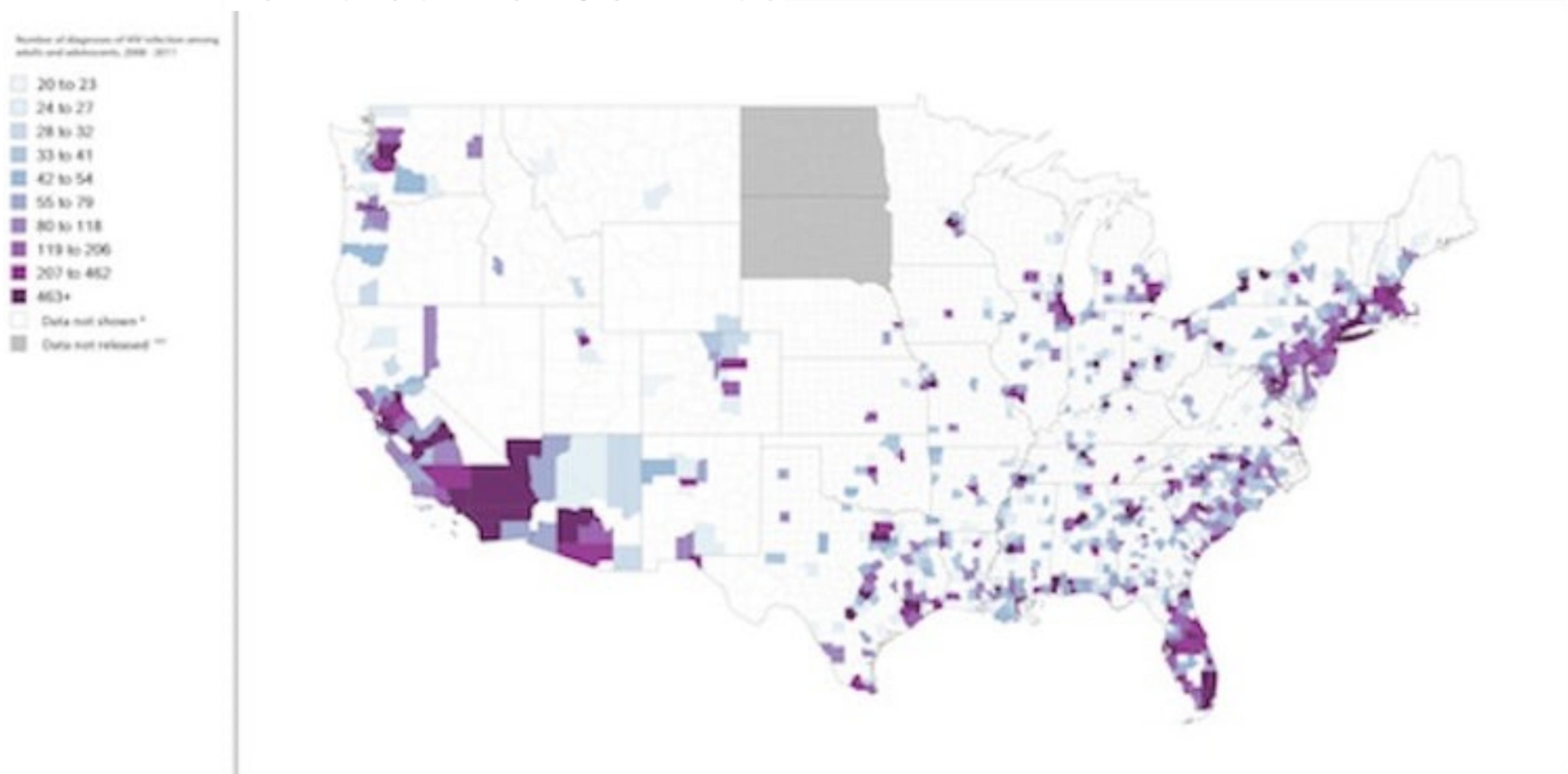


Source: Bloomberg NEF

## UNITED STATES ECONOMIC ACTIVITY, SPLIT IN HALF

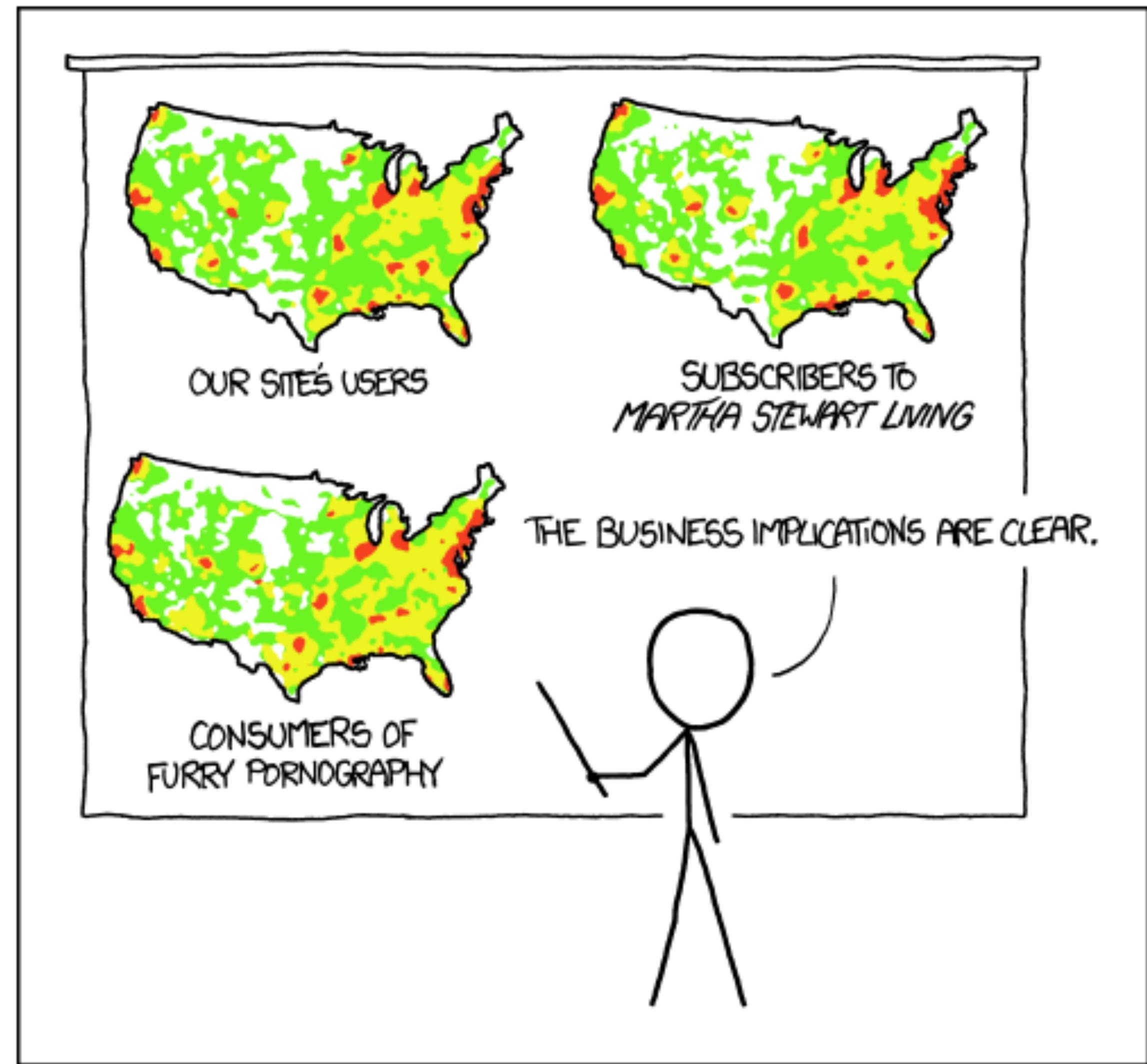


# Startling New Map: 92 Percent of New HIV Cases are in 25 Percent of Counties



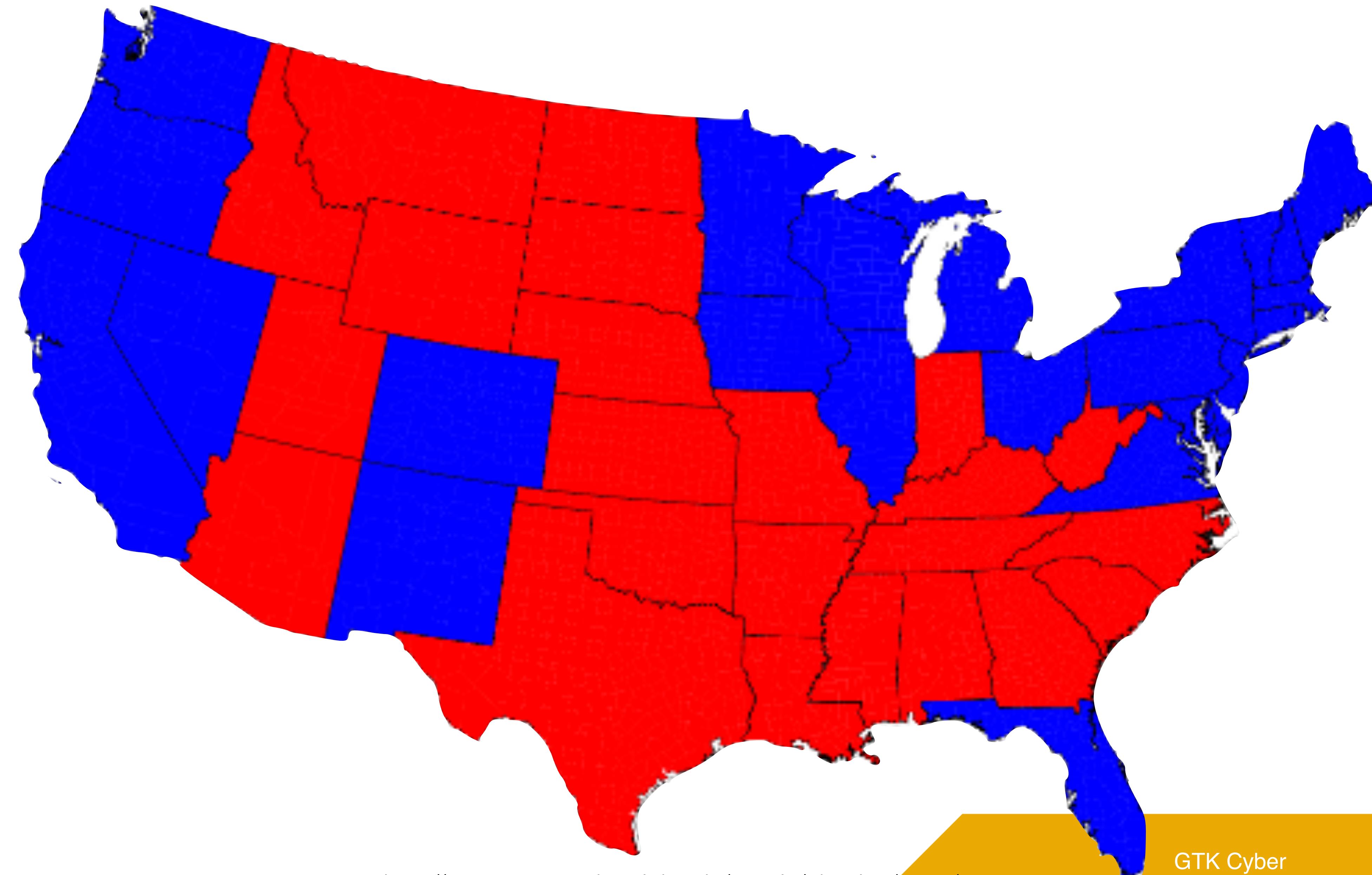
\* Data are not shown to protect privacy.

\*\* State health department, per its HIV data re-release agreement with CDC, requested not to release data to AIDSvu.

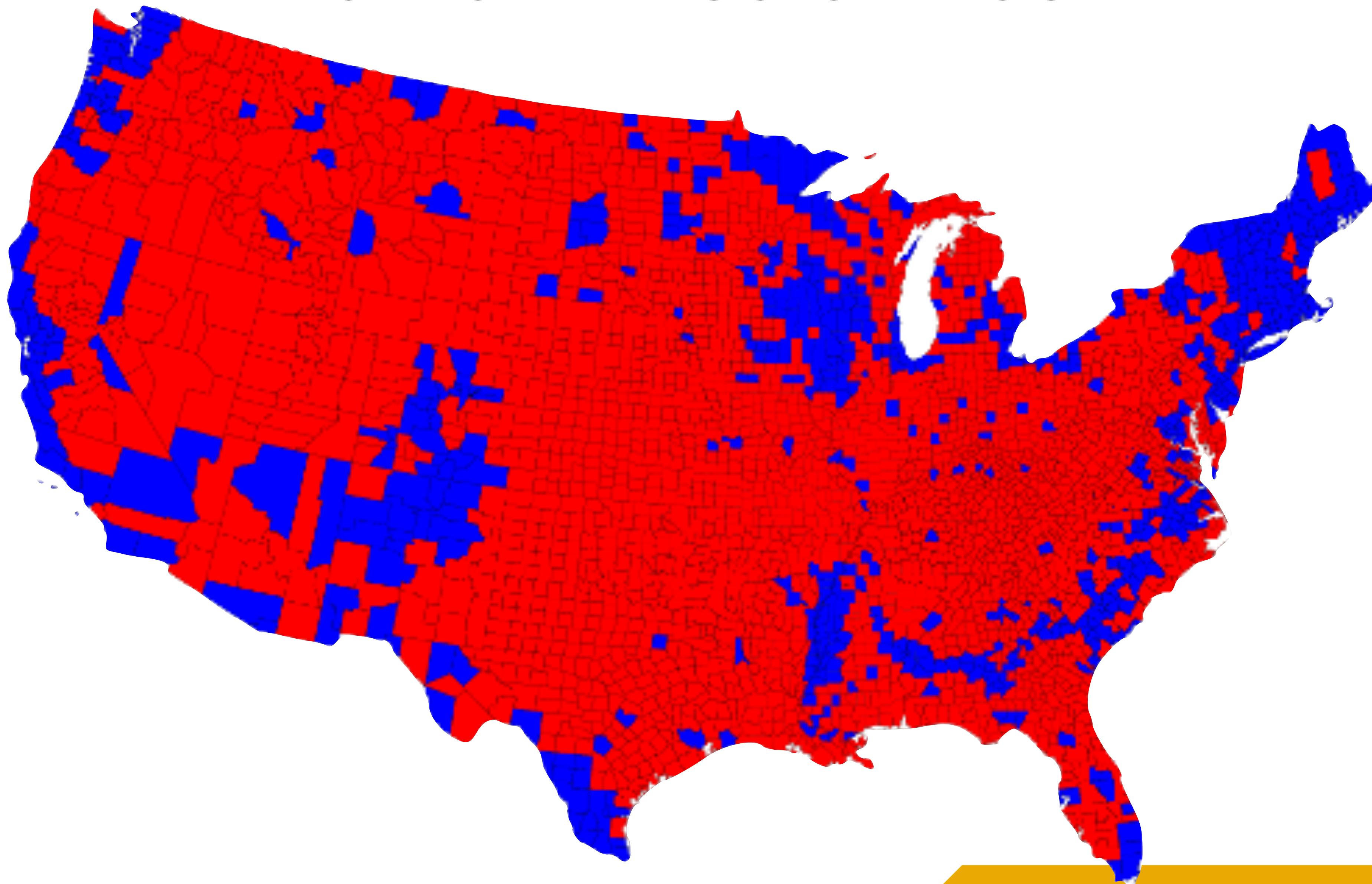


PET PEEVE #208:  
GEOGRAPHIC PROFILE MAPS WHICH ARE  
BASICALLY JUST POPULATION MAPS

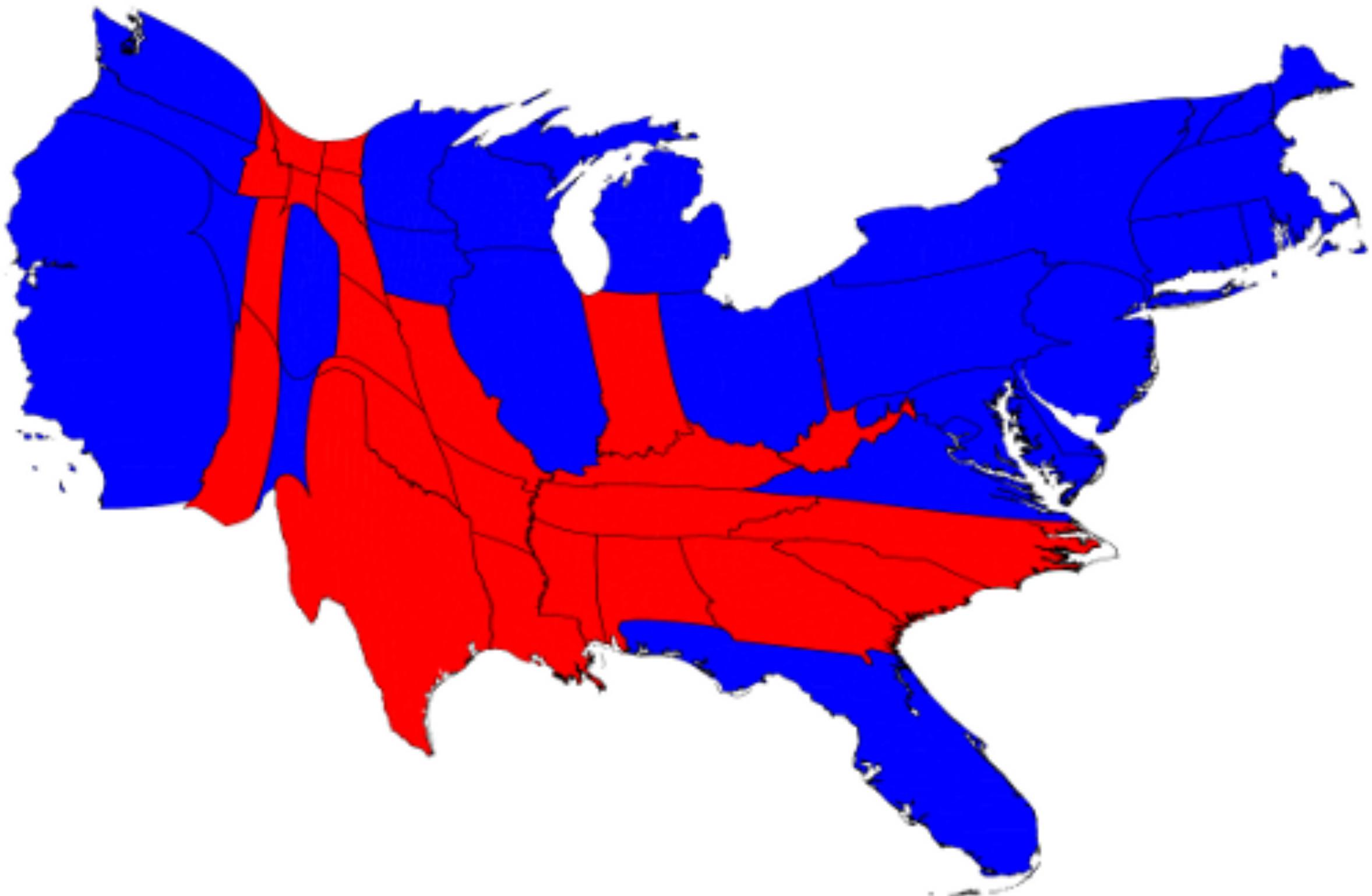
# Who won? Red or Blue?



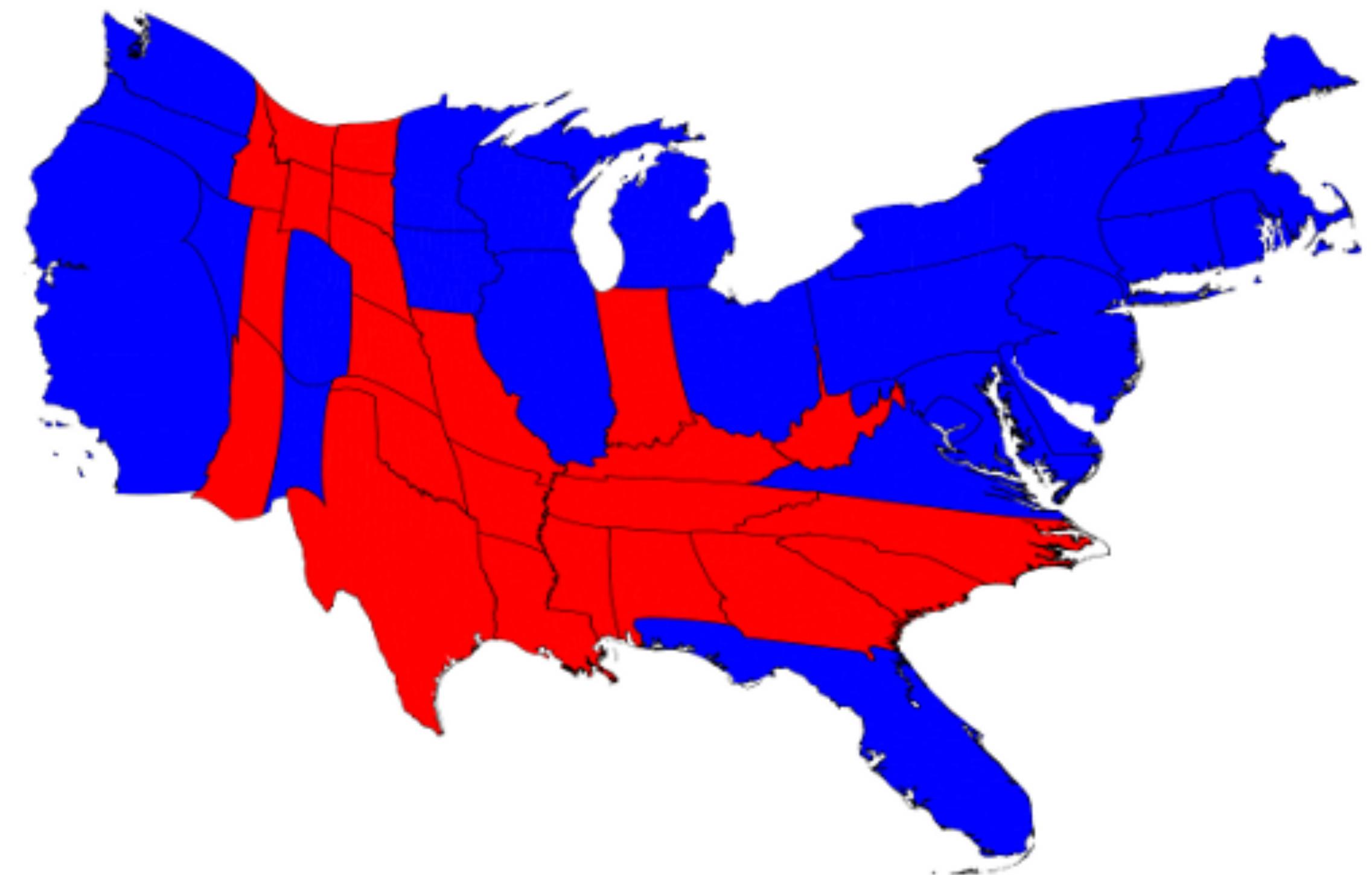
# Who won? Red or Blue?



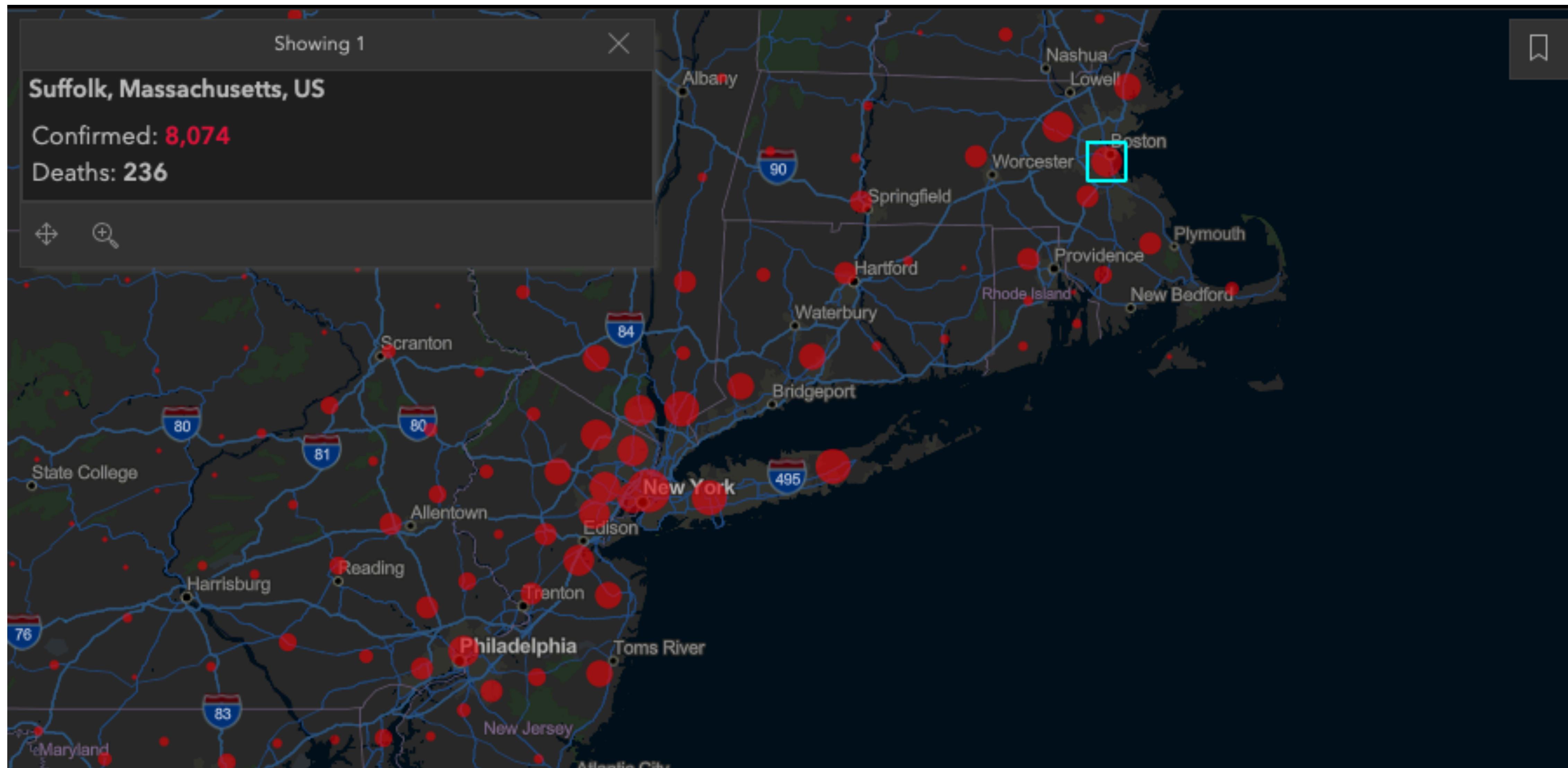
# Who won? Red or Blue?

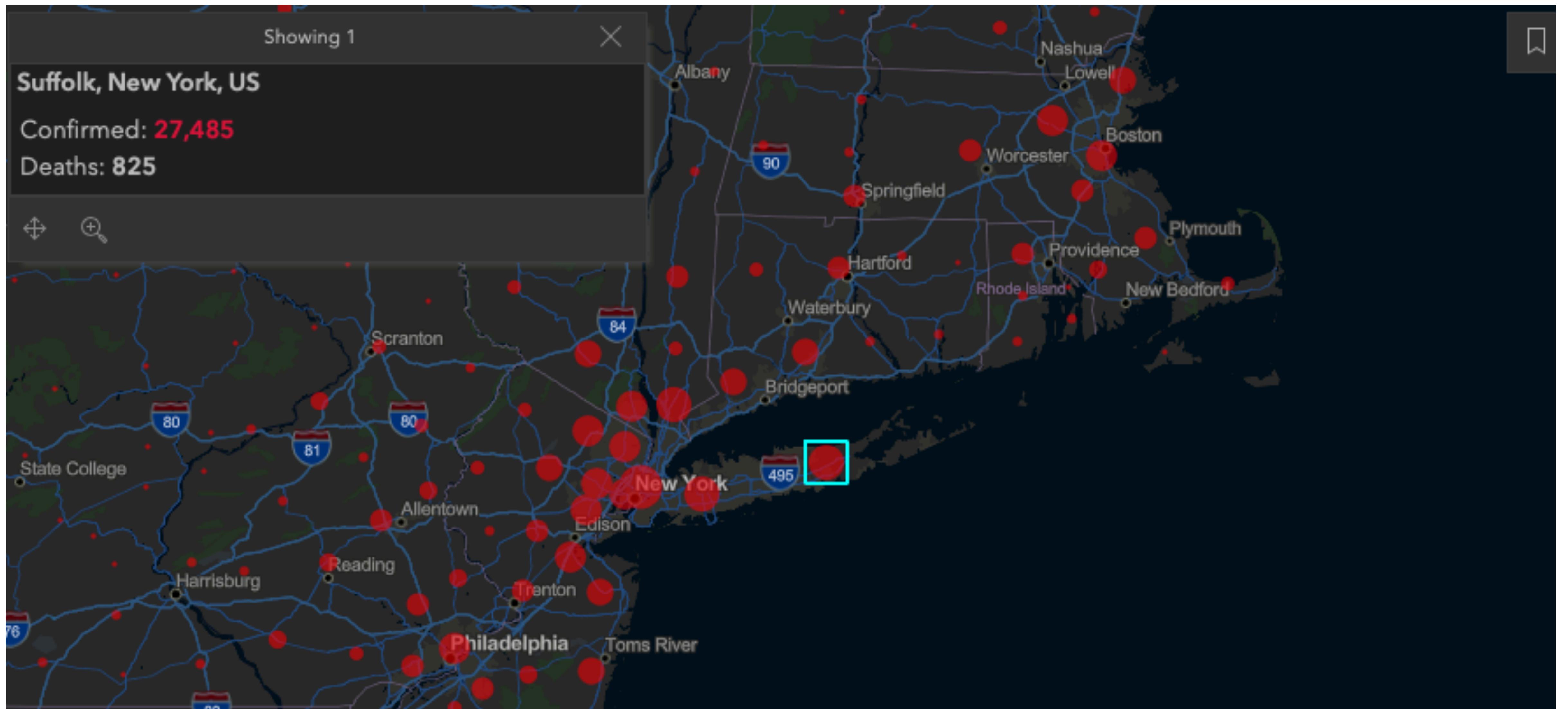


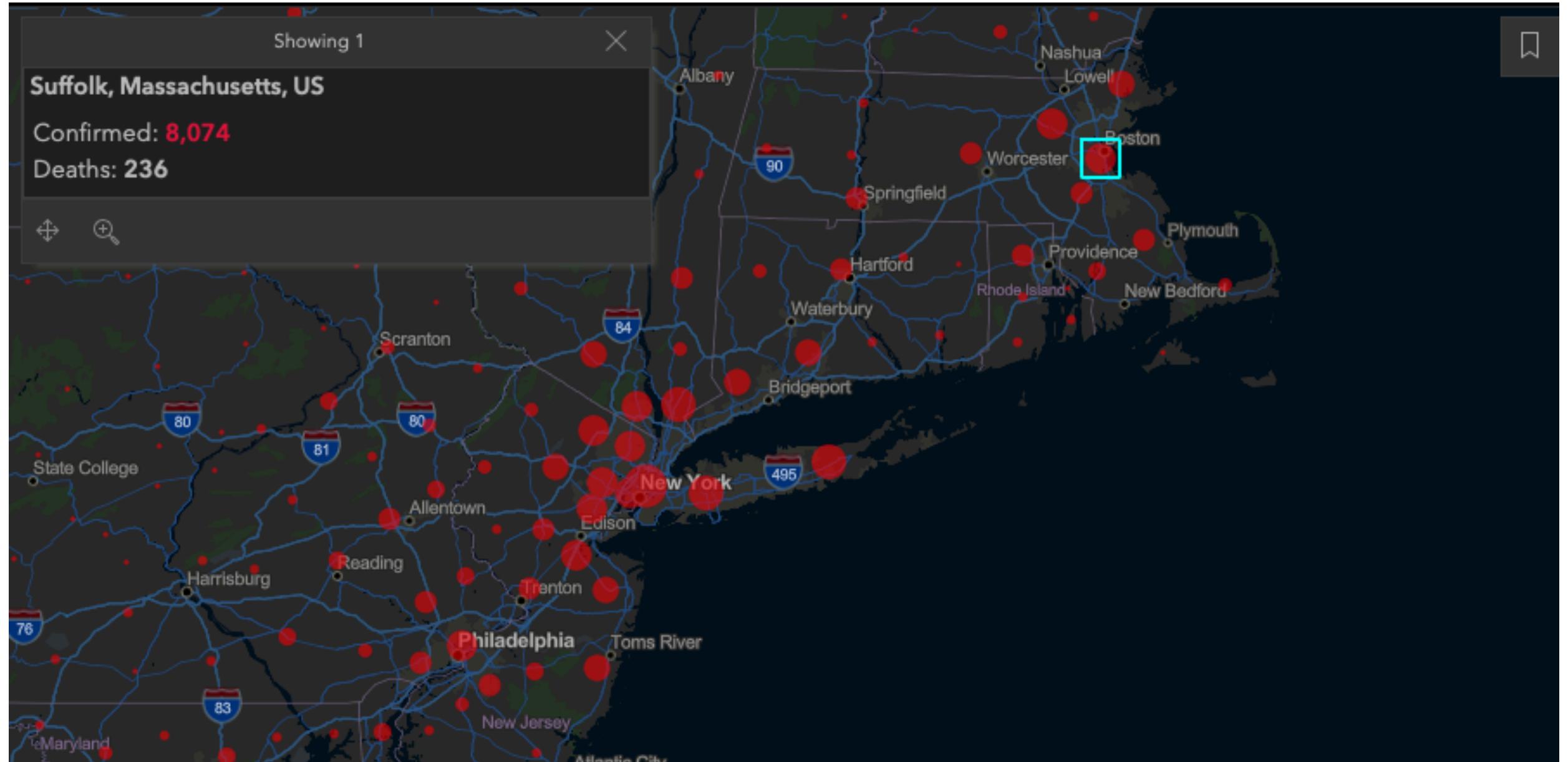
Scaled by Population



Scaled by Electoral Votes

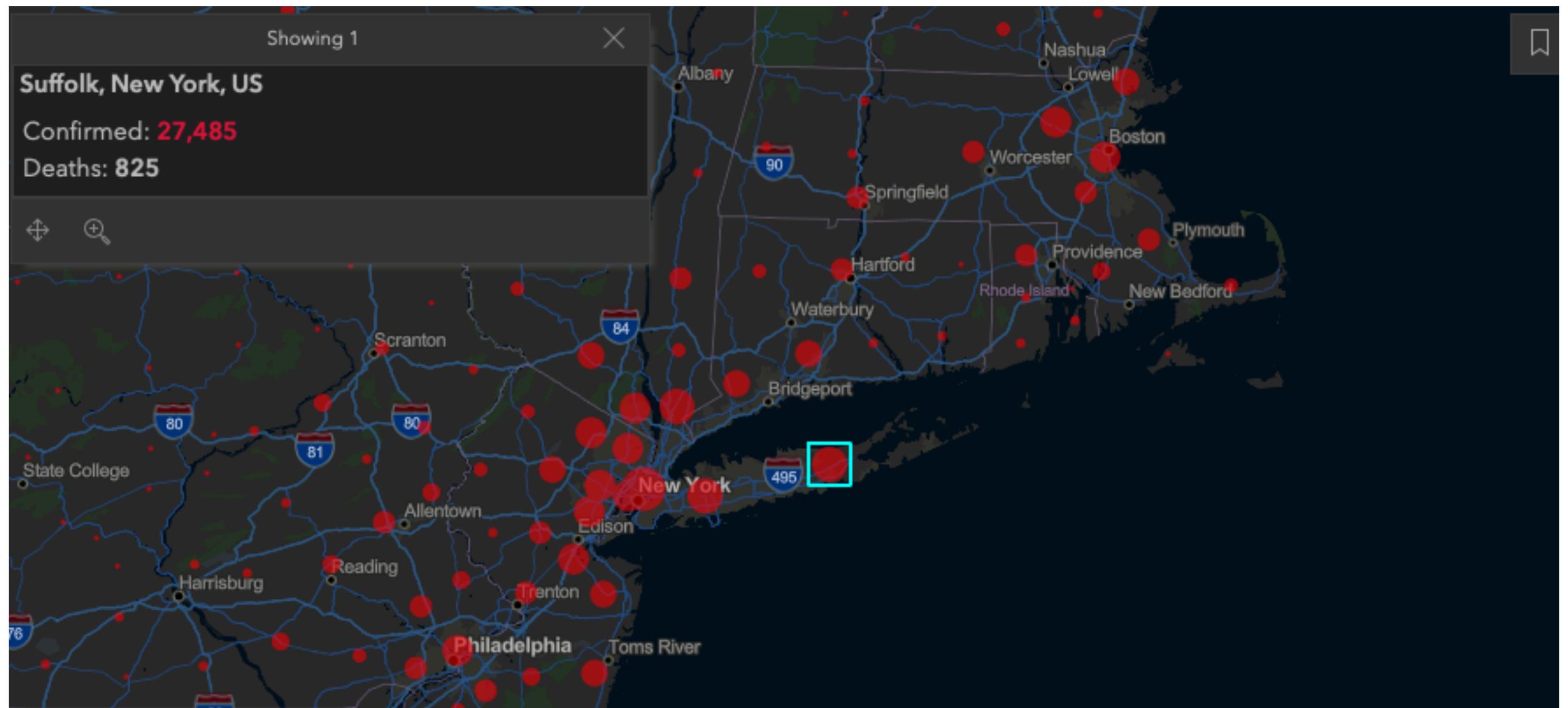






# Suffolk vs Suffolk

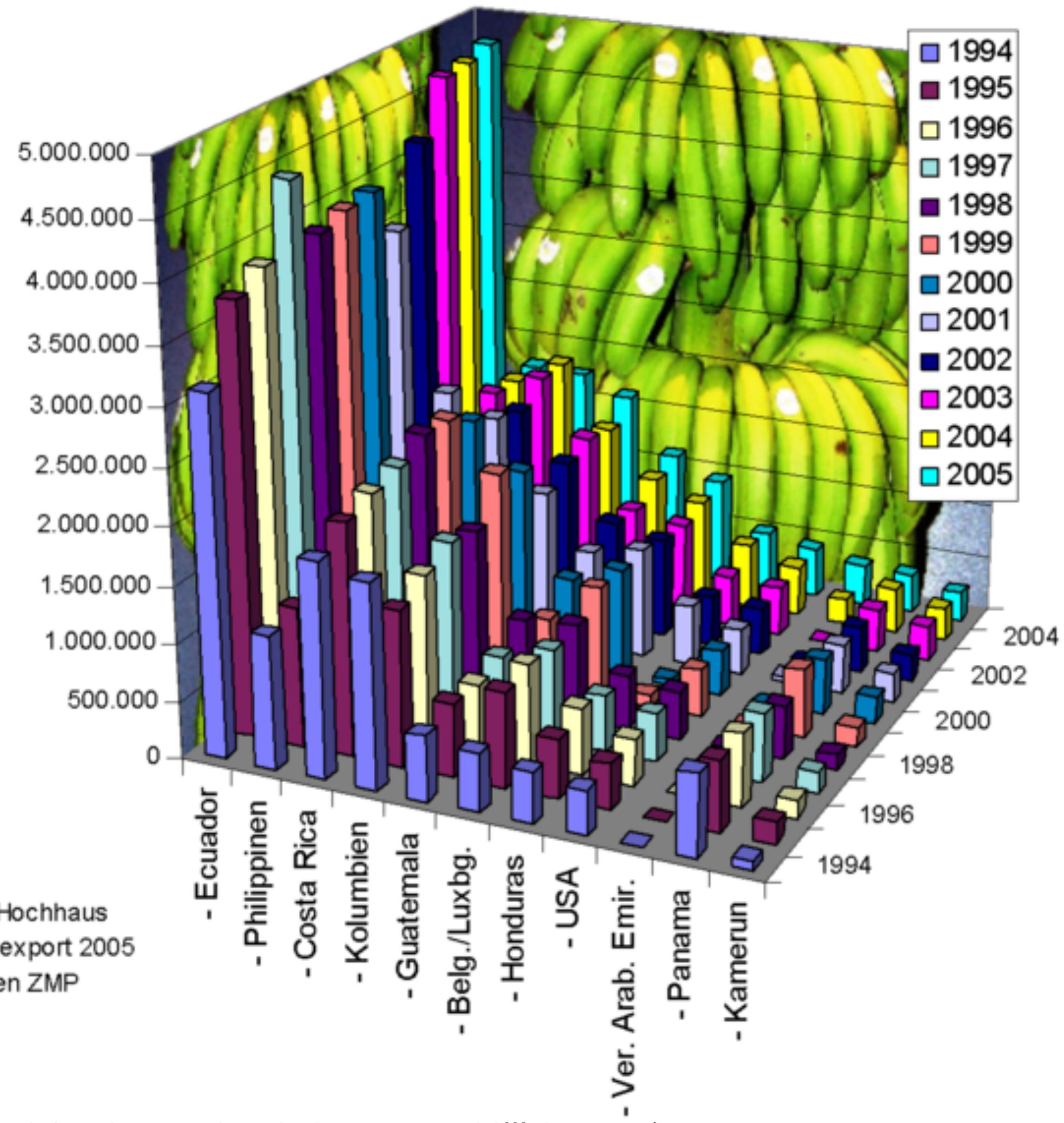
Suffolk, Massachusetts  
8,074 confirmed cases



Suffolk New York  
27,485 confirmed cases

# The Ugly

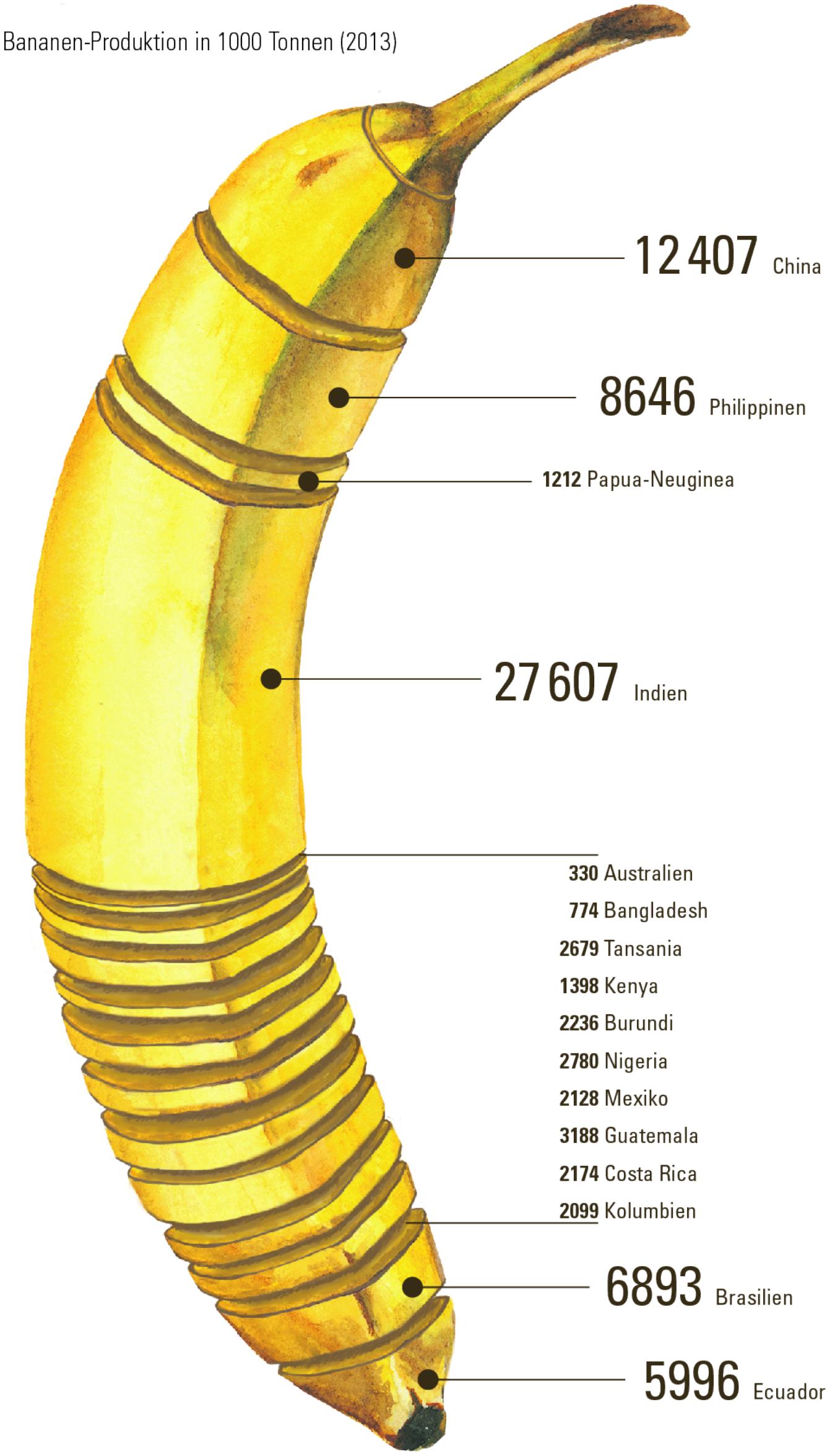
## Export von Bananen in Tonnen von 1994-2005



Dr. Hochhaus  
Banlexport 2005  
Daten ZMP

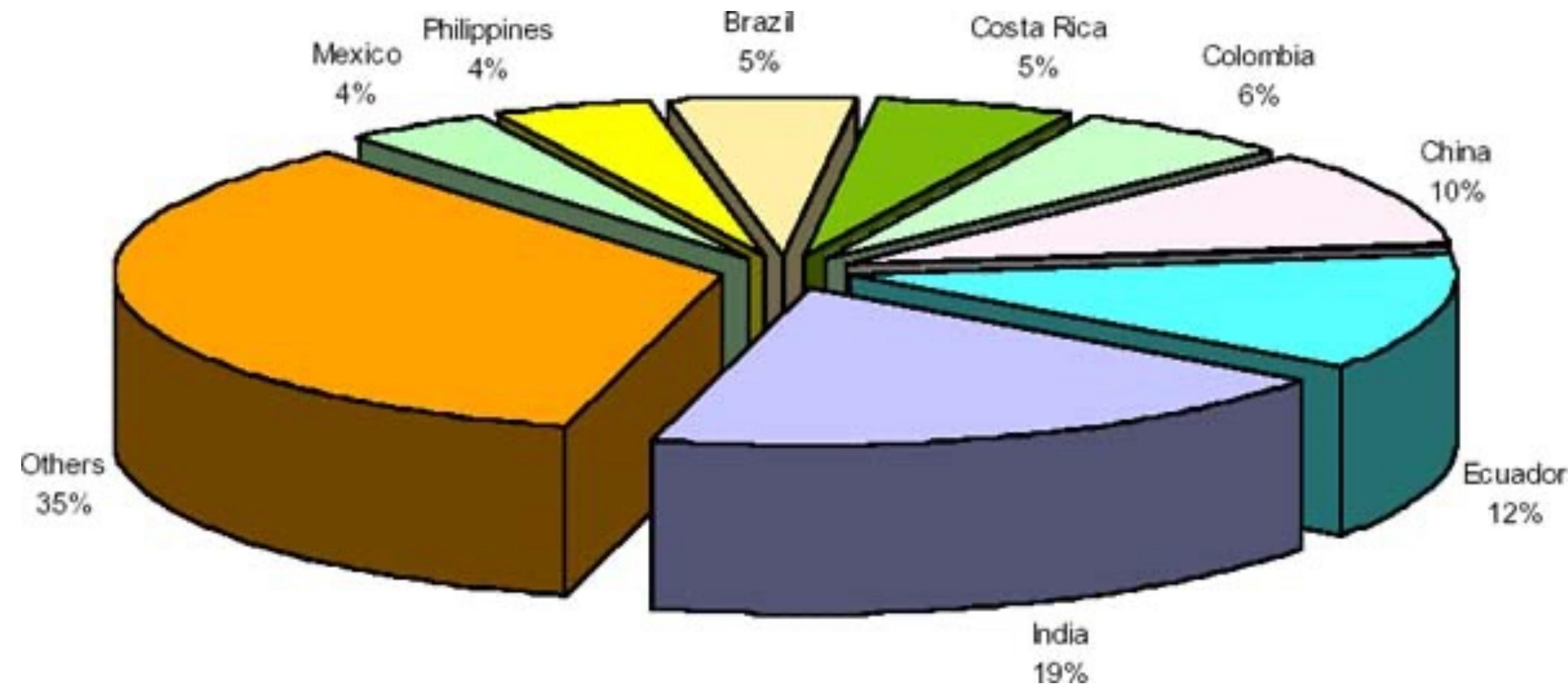
Eine Südfrucht

Bananen-Produktion in 1000 Tonnen (2013)

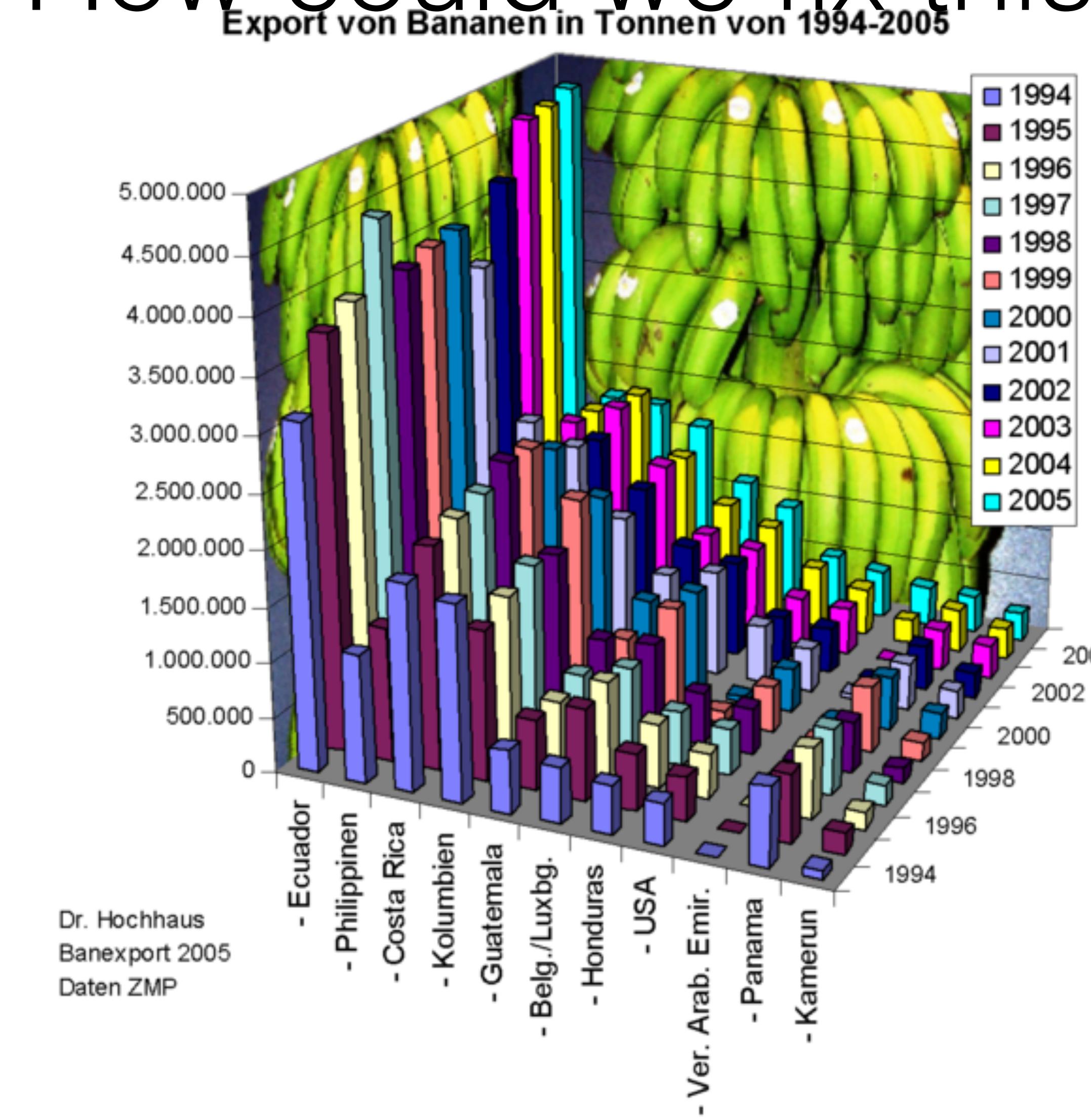


QUELLE: FAOSTAT

NZZ-Infografik/lea.



# How could we fix this?



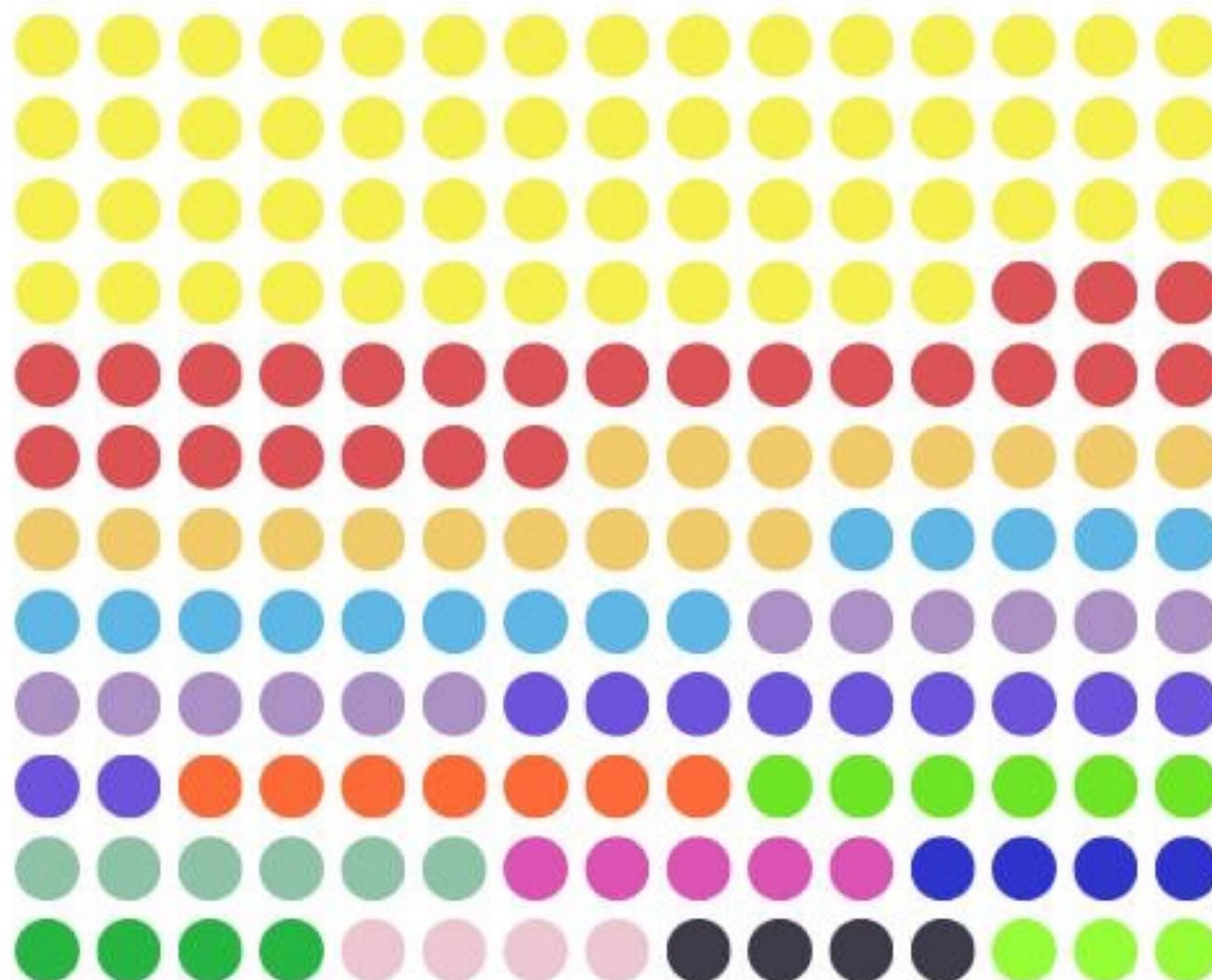
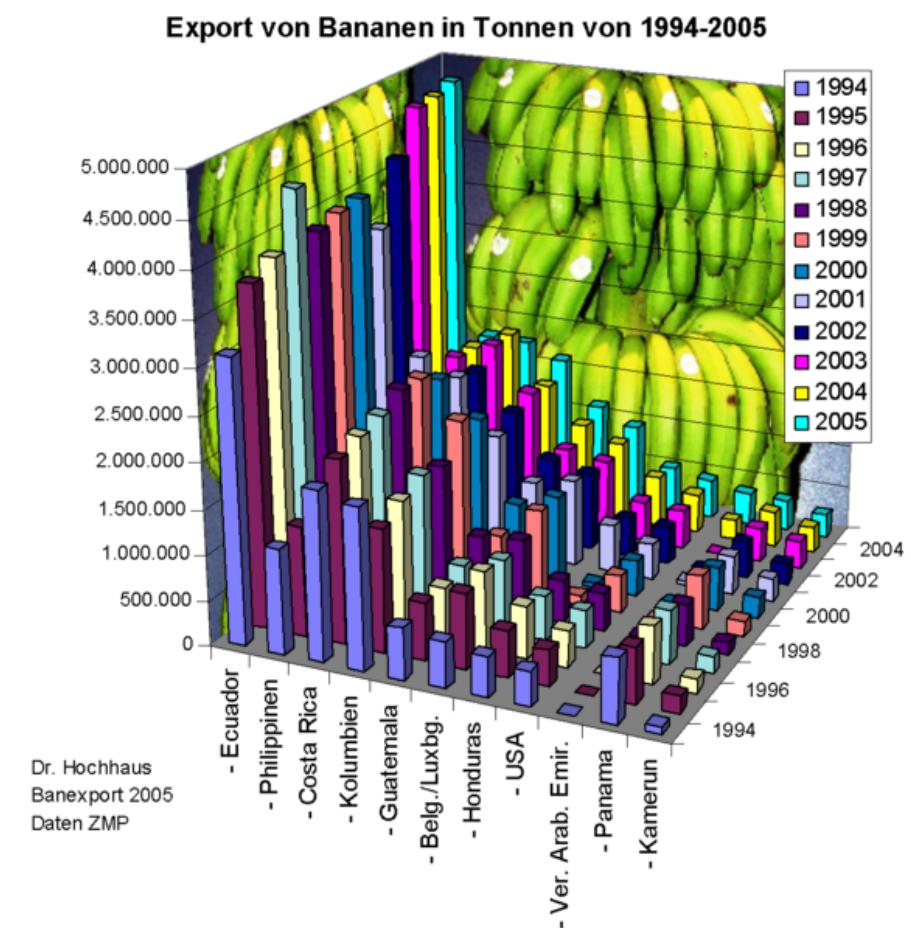
# How could we fix this?



## Production of bananas in the world



data of 2013



● India ● China ● Philippines ● Brazil ● Ecuador  
● Indonesia ● Guatemala ● Angola ● Tanzania ● Burundi  
● Costa Rica ● Mexico ● Colombia ● Viet Nam ● Thailande

World rank	Countries	Production of bananas in thousand of ton
1	India	27575
2	China	12075
3	Philippines	8646
4	Brazil	6893
5	Ecuador	5996
6	Indonesia	5359
7	Guatemala	3188
8	Angola	3095
9	Tanzania	2679
10	Burundi	2236
11	Costa Rica	2175
12	Mexico	2128
13	Colombia	2099
14	Viêt Nam	1893
15	Thailande	1585

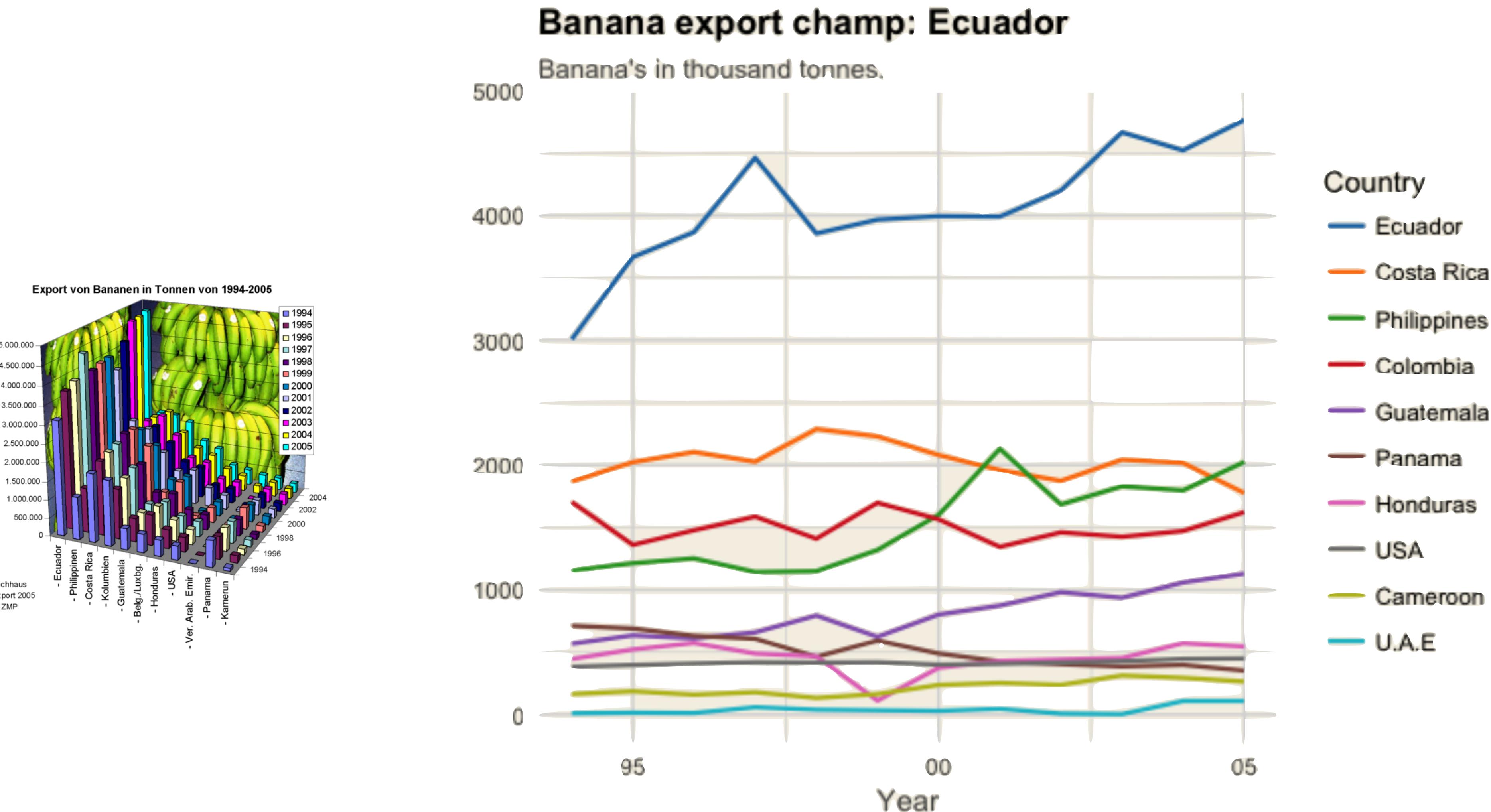
Post by PacoJoke

[https://www.reddit.com/r/dataisbeautiful/comments/6dv56j/production\\_of\\_bananas\\_in\\_the\\_world\\_oc/](https://www.reddit.com/r/dataisbeautiful/comments/6dv56j/production_of_bananas_in_the_world_oc/)

<https://hochhaus-schiffsbetrieb.jimdo.com/200-jahre-internationale-bananenschiffahrt-273/>

GTK Cyber

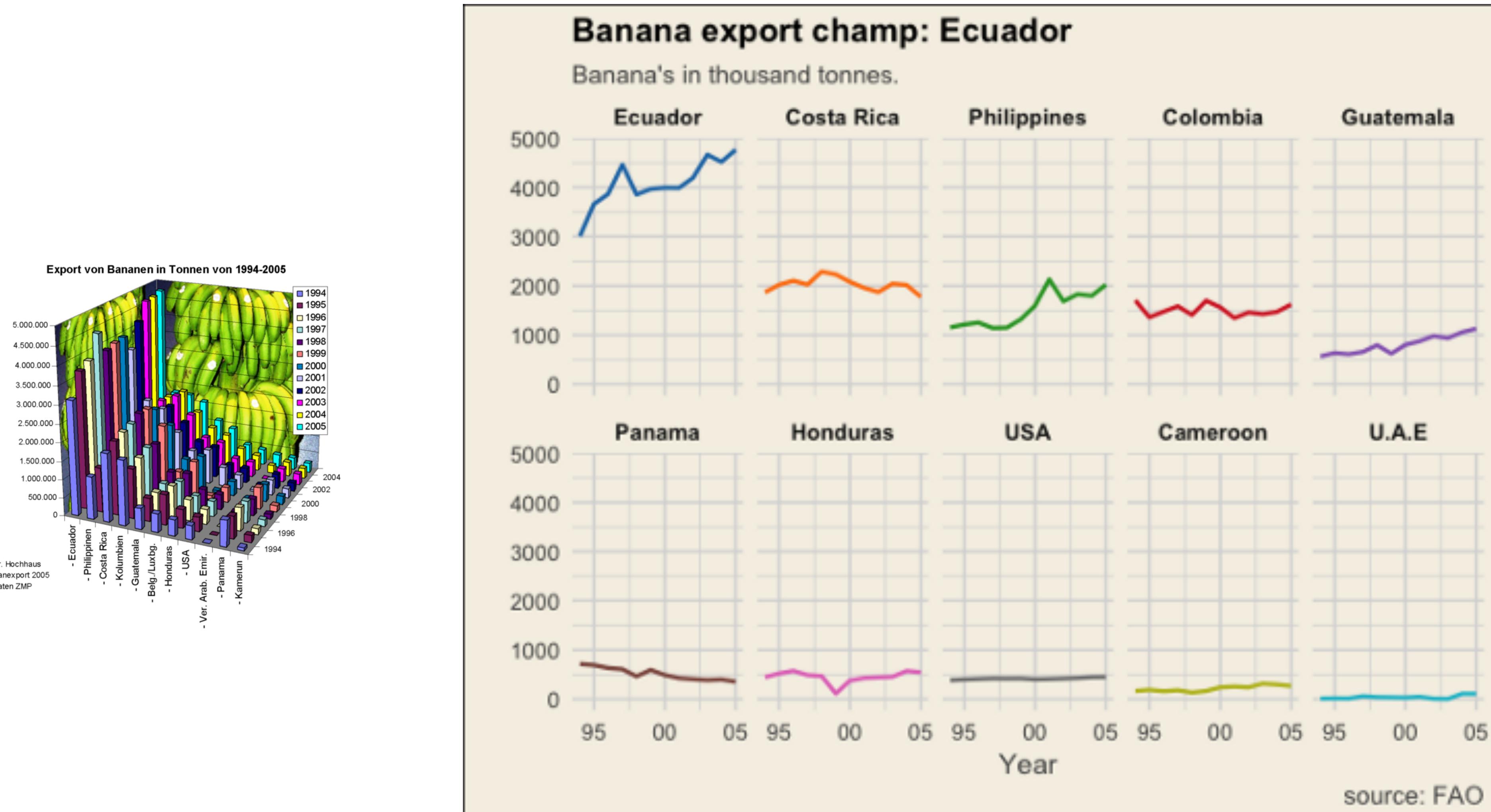
# How could we fix this?



<https://medium.com/tdebeus/redesign-of-a-truly-bananas-chart-1617f930808d>

<https://hochhaus-schiffsbetrieb.jimdo.com/200-jahre-internationale-bananenschifffahrt-273/>

# How could we fix this? Small Multiples

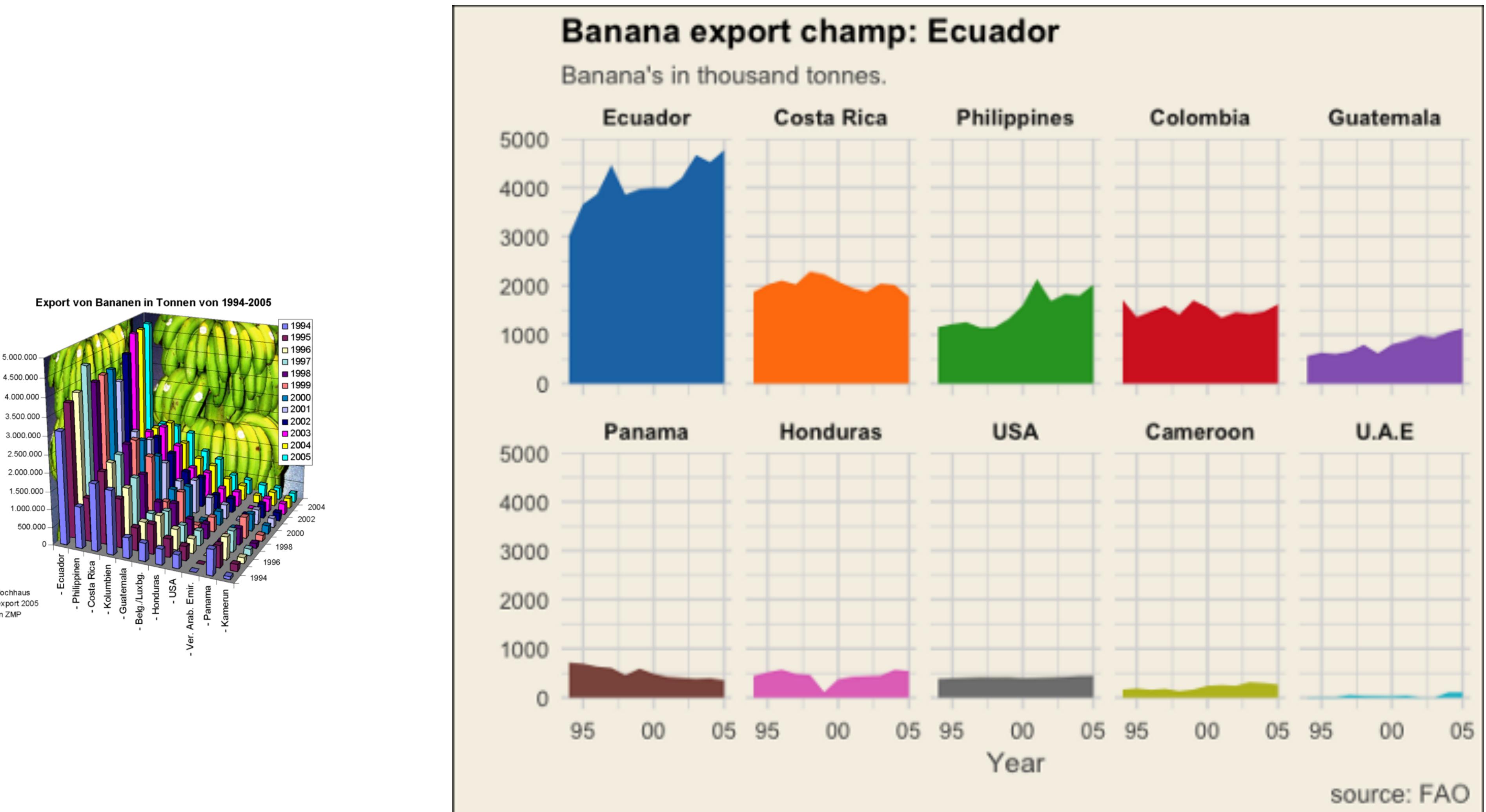


<https://medium.com/tdebeus/redesign-of-a-truly-bananas-chart-1617f930808d>

<https://hochhaus-schiffsbetrieb.jimdo.com/200-jahre-internationale-bananenschifffahrt-273/>

GTK Cyber

# How could we fix this? Small Multiples



<https://medium.com/tdebeus/redesign-of-a-truly-bananas-chart-1617f930808d>

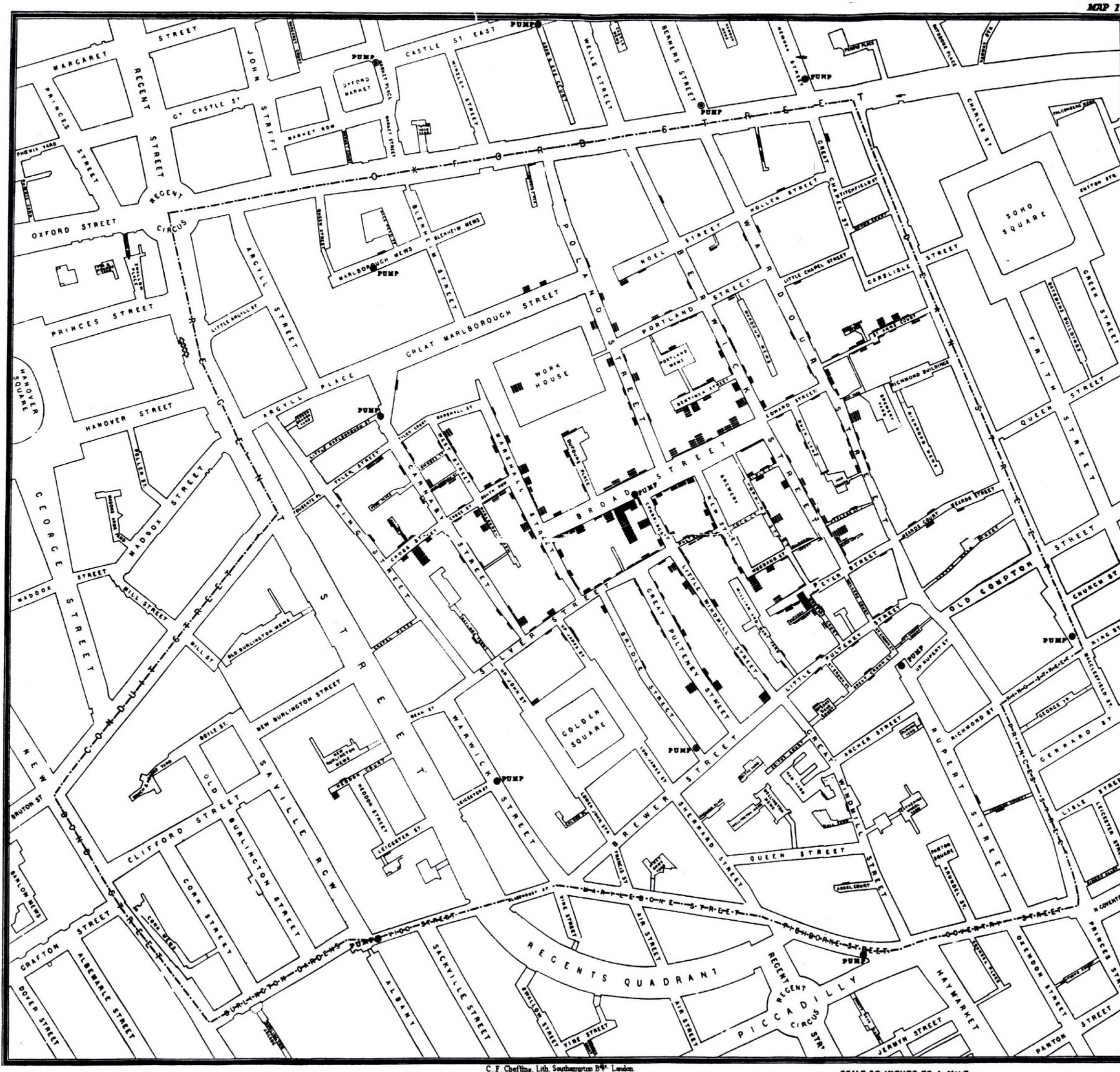
<https://hochhaus-schiffsbetrieb.jimdo.com/200-jahre-internationale-bananenschifffahrt-273/>

GTK Cyber

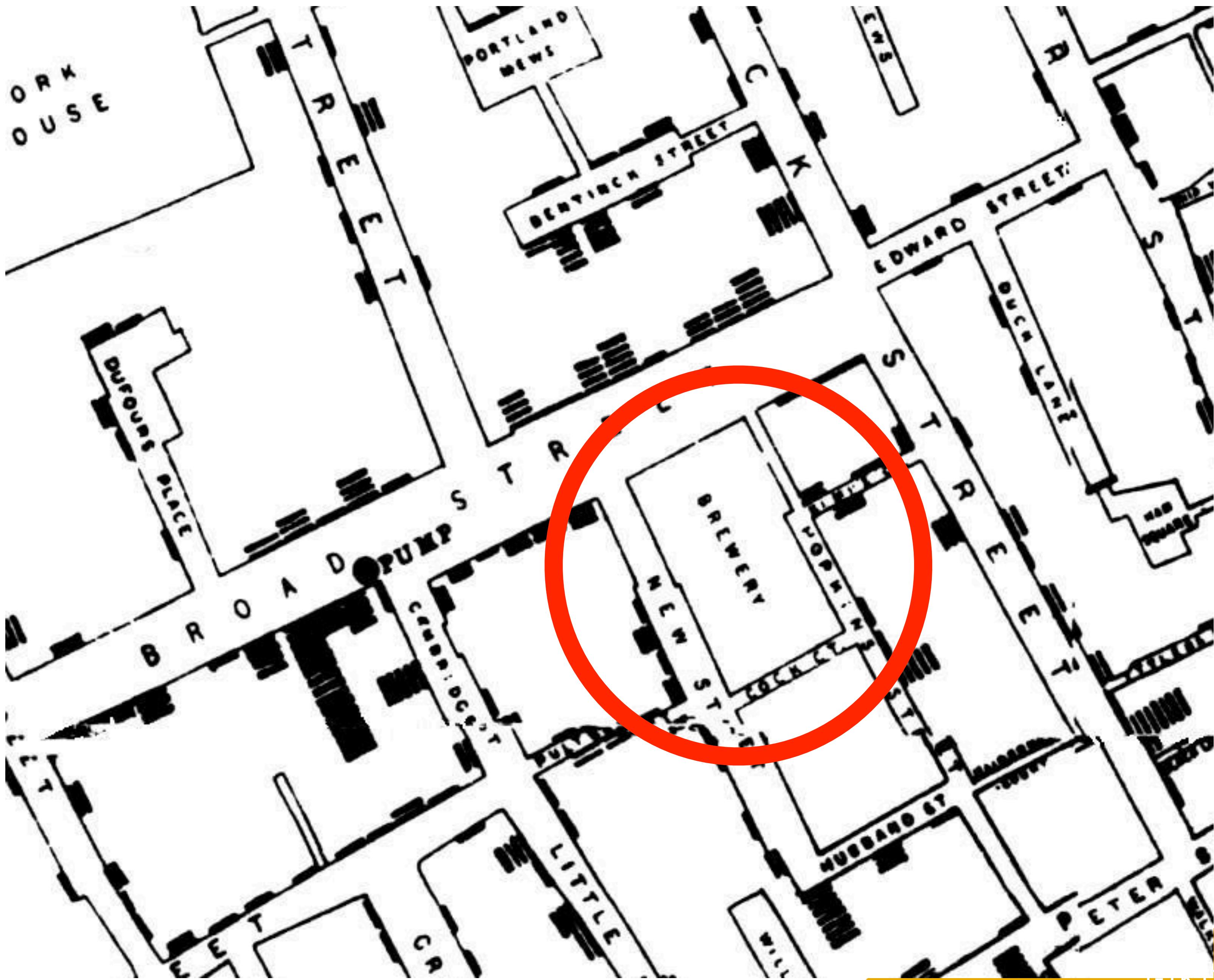
# The Good

# Proper Display



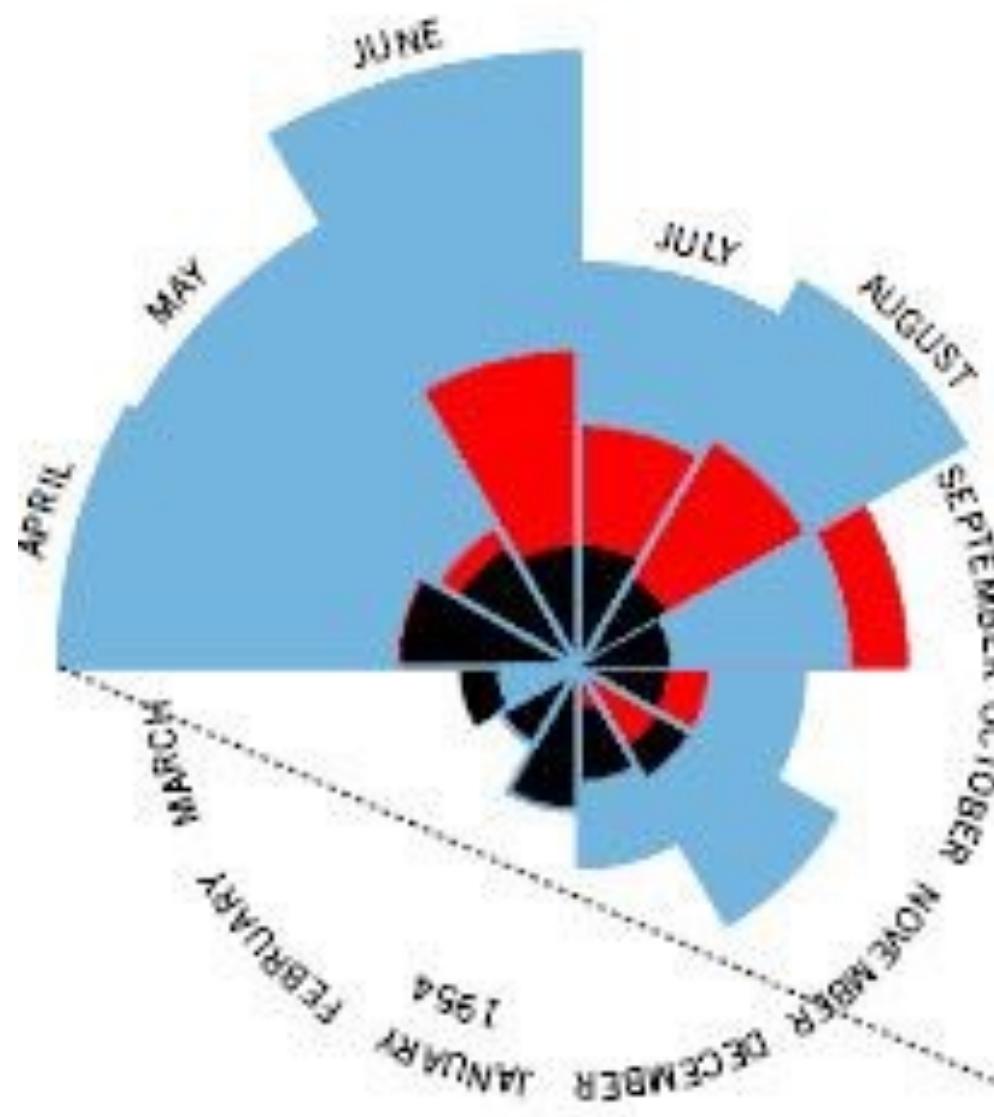


- Created by John Snow in 1854
- This visualization proved that cholera was water-borne
- Snow identified the source of the outbreak as a water pump on Broad Street
- Spurred city to create a true sewage system

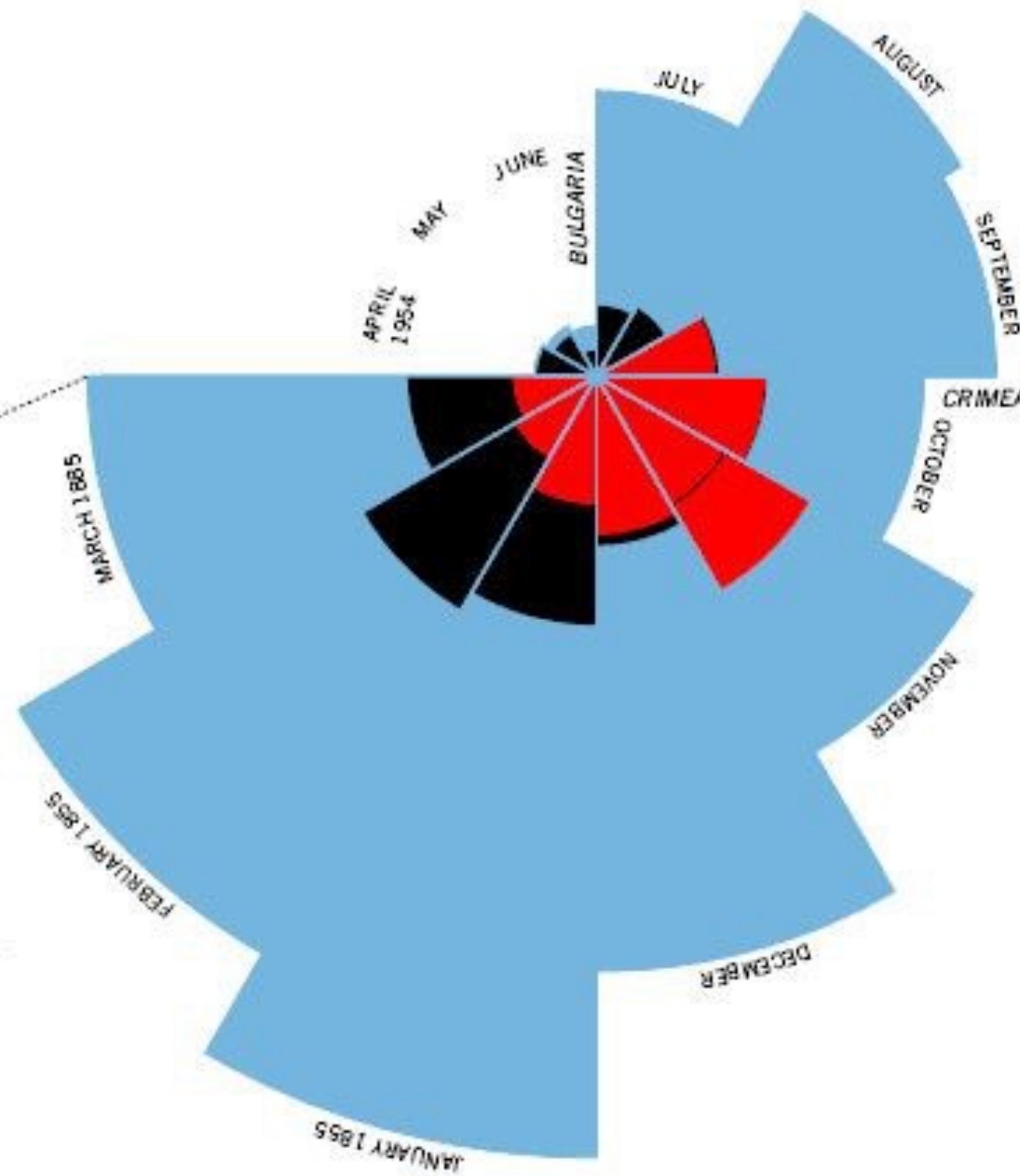


# DIAGRAM OF THE CAUSES OF MORTALITY IN THE ARMY IN THE EAST

2.  
APRIL 1855 to MARCH 1856



1.  
APRIL 1854 to MARCH 1855



*The Areas of the blue, red, & black wedges are each measured from the centre as the common vertex*

*The blue wedges measured from the centre of the circle represent area for area the deaths from Preventible or Mitigable Zymotic Diseases, the red wedges measured from the centre the deaths from wounds, & the black wedges measured from the centre the deaths from all other causes*

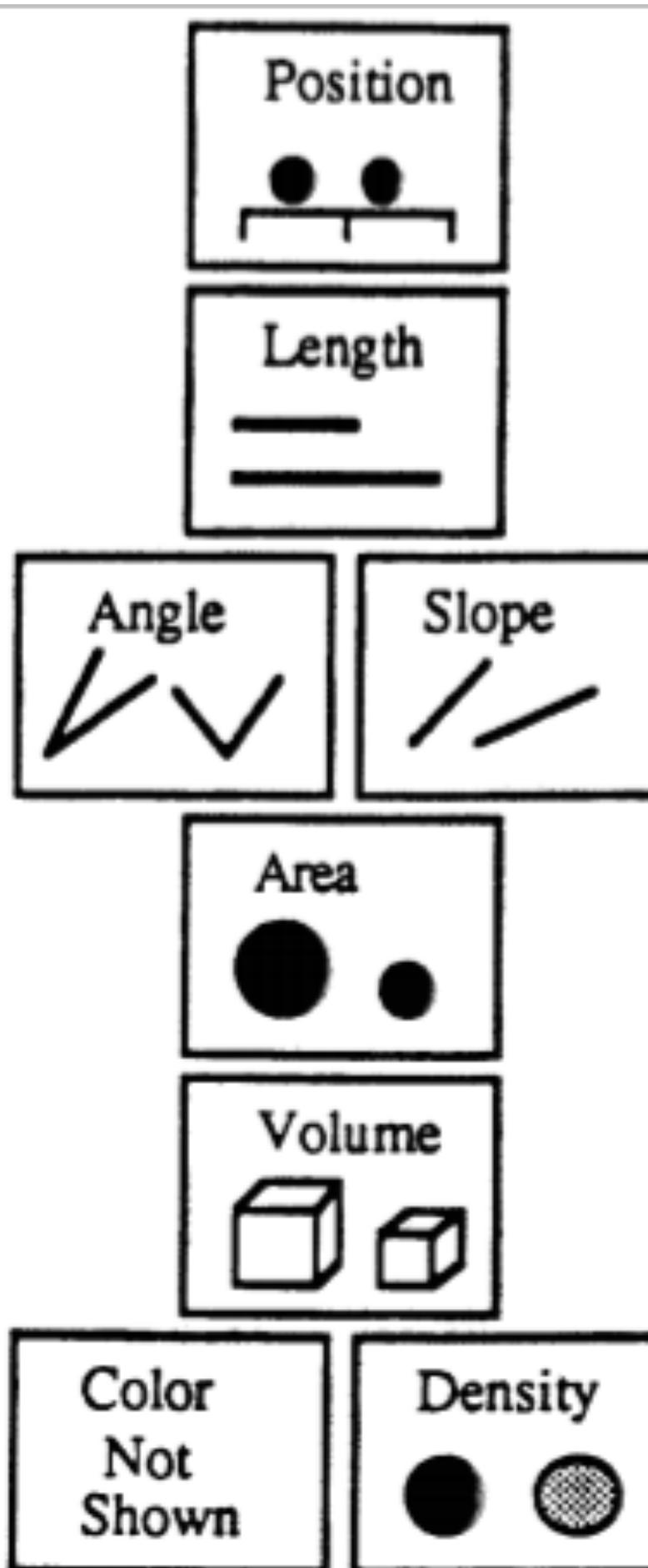
*The black line across the red triangle in Nov '1854 marks the boundary of the deaths from all other causes during the month*

*In October 1854, & April 1855, the black area coincides with the red, in January & February 1856, the blue coincides with the black*

*The entire areas may be compared by following the blue, the red & the black lines enclosing them.*

# **Visual Encoding**

More accurate

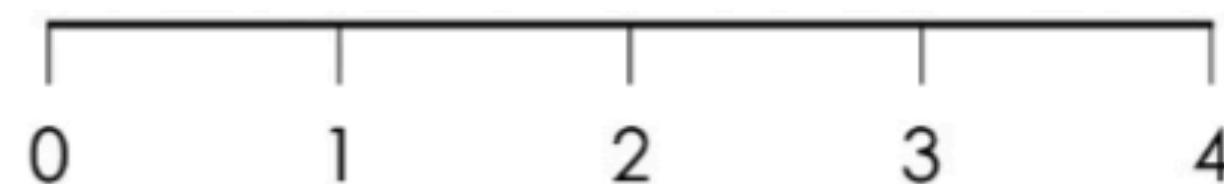


Less accurate



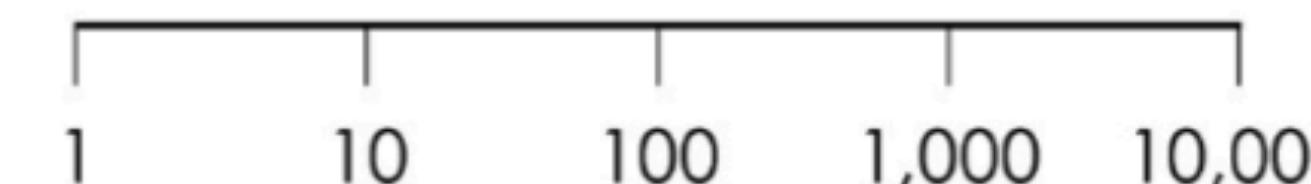
## Linear

Values are evenly spaced



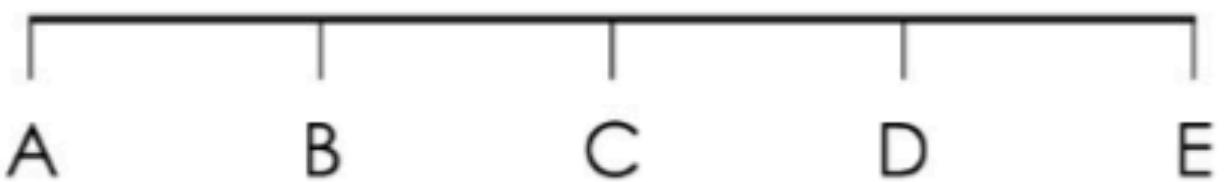
## Logarithmic

Focus on percent change



## Categorical

Discrete placement in bins



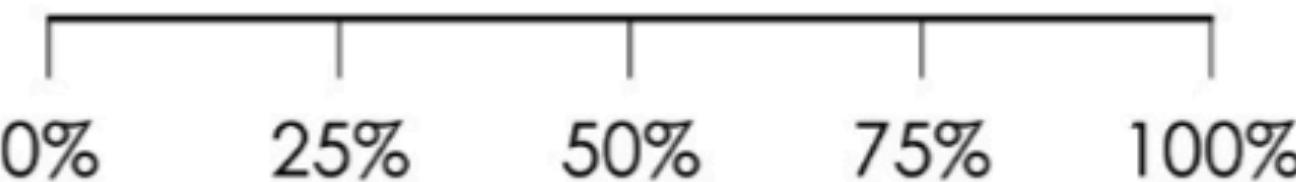
## Ordinal

Categories where order matters



## Percent

Representing parts of a whole

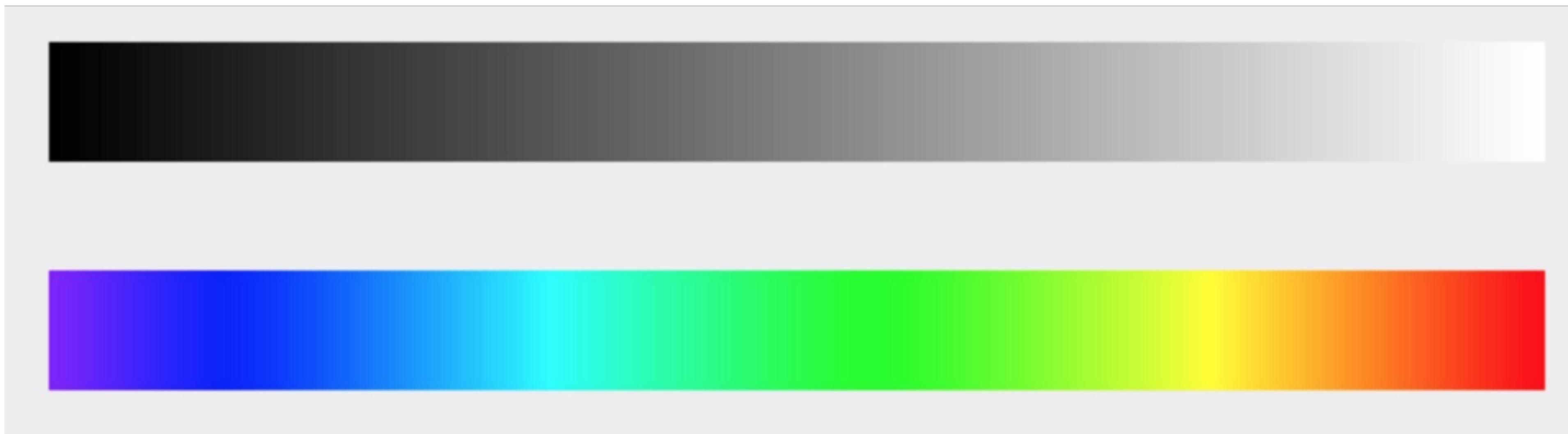


## Time

Units of months, days, or hours

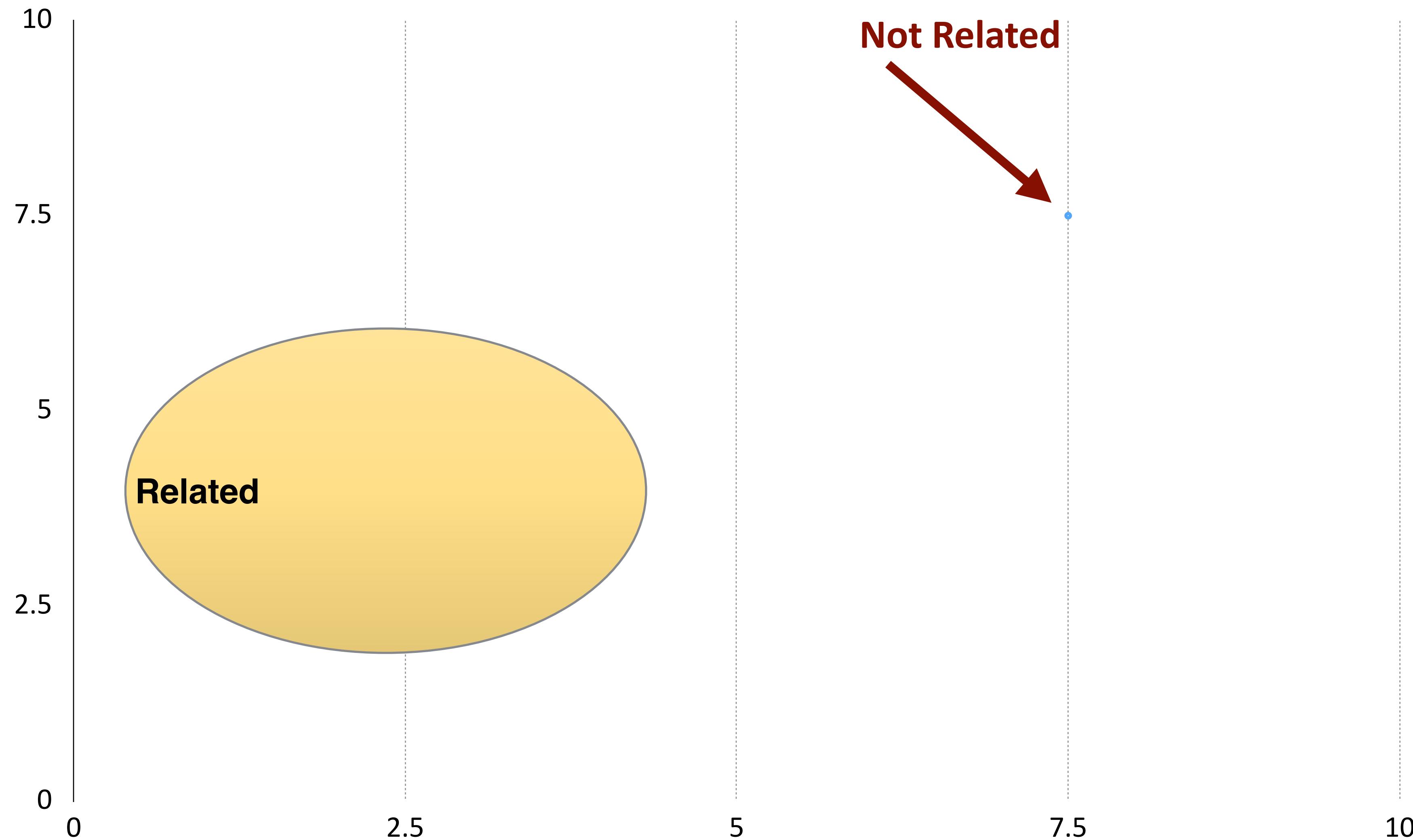


# Color



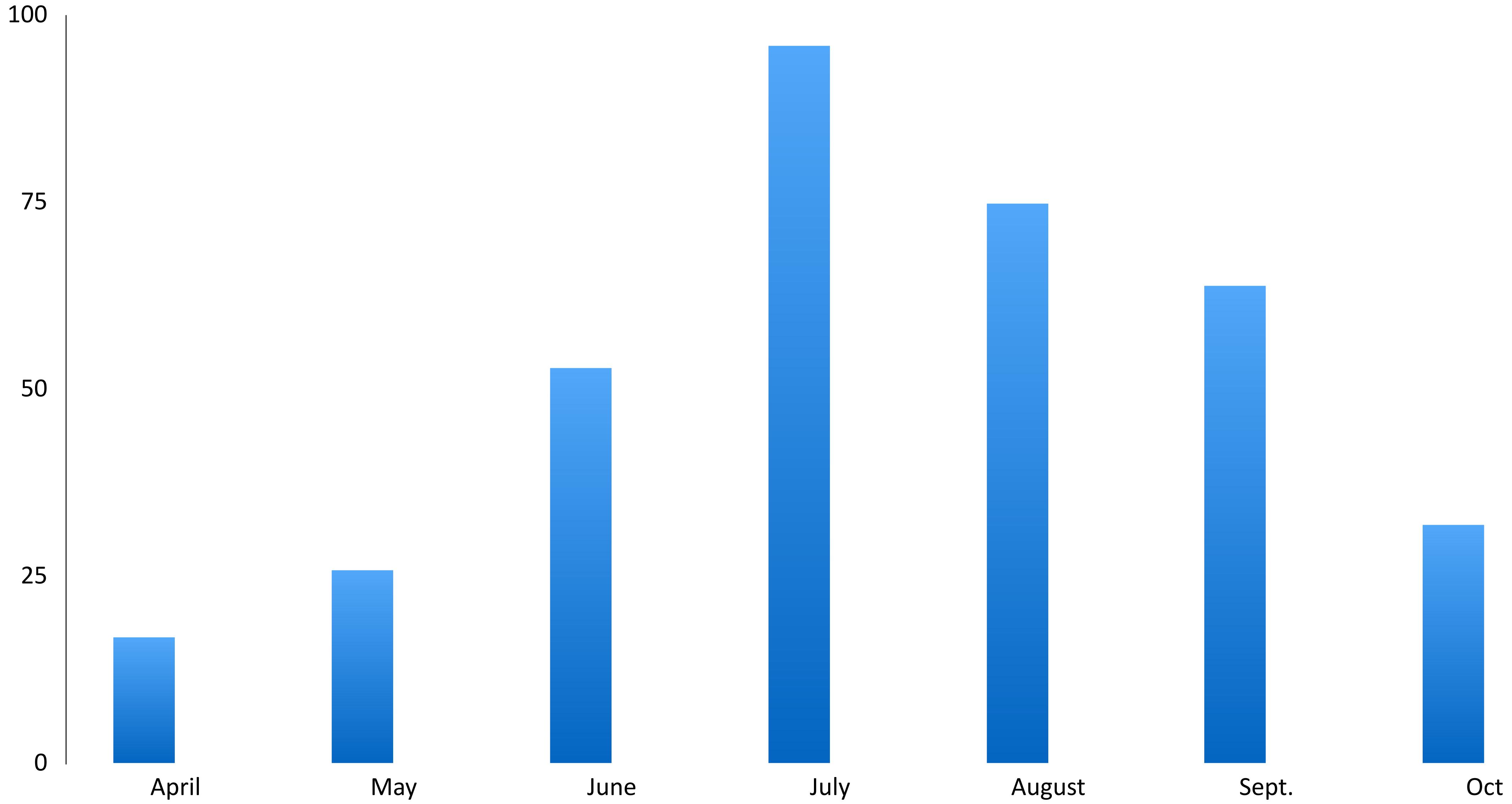
# **Visual Encodings: Translating Data into Visual Form**

# **Visual Encodings: Position**



GTK Cyber

# **Visual Encodings: Length**



GTK Cyber





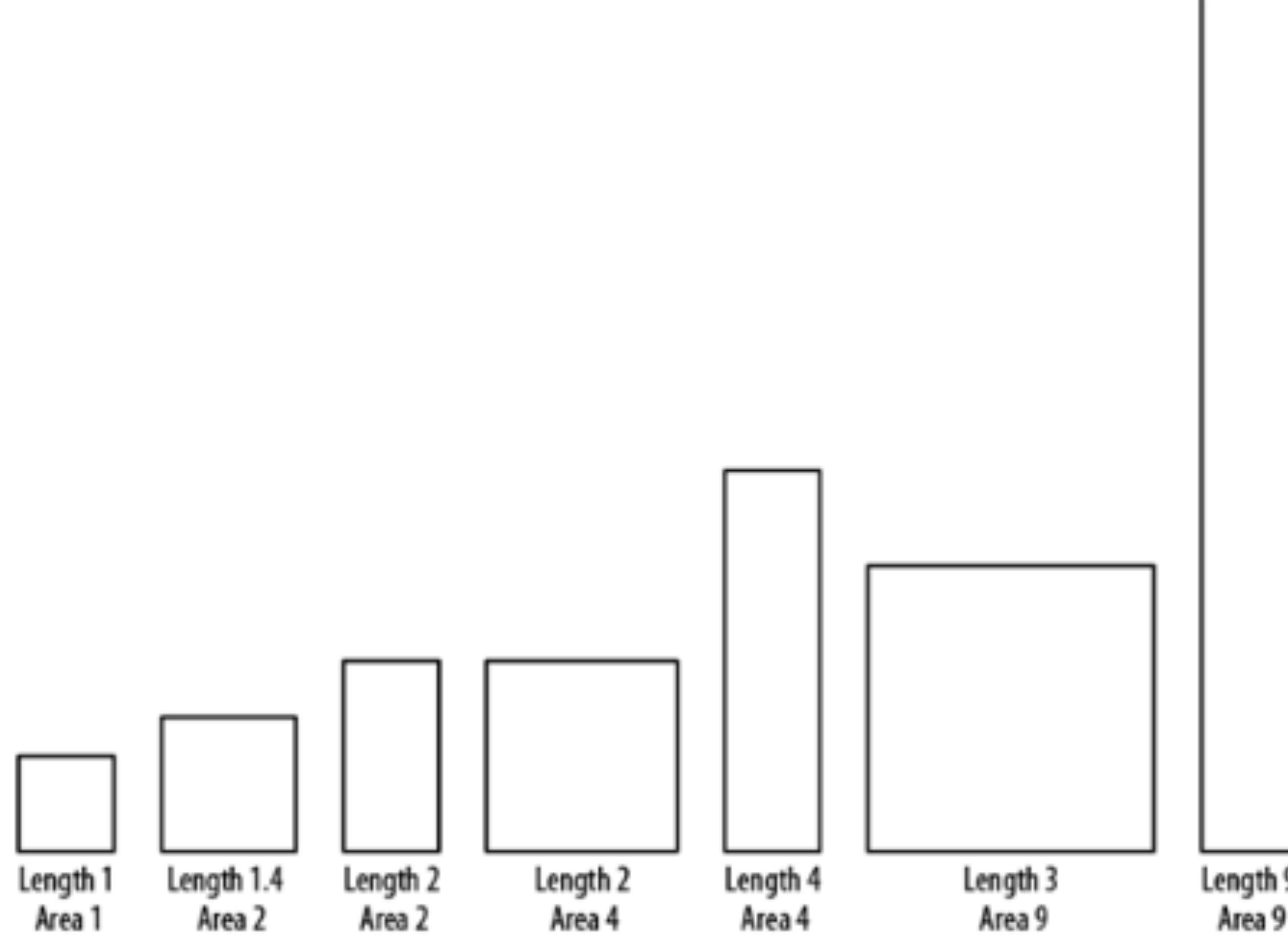


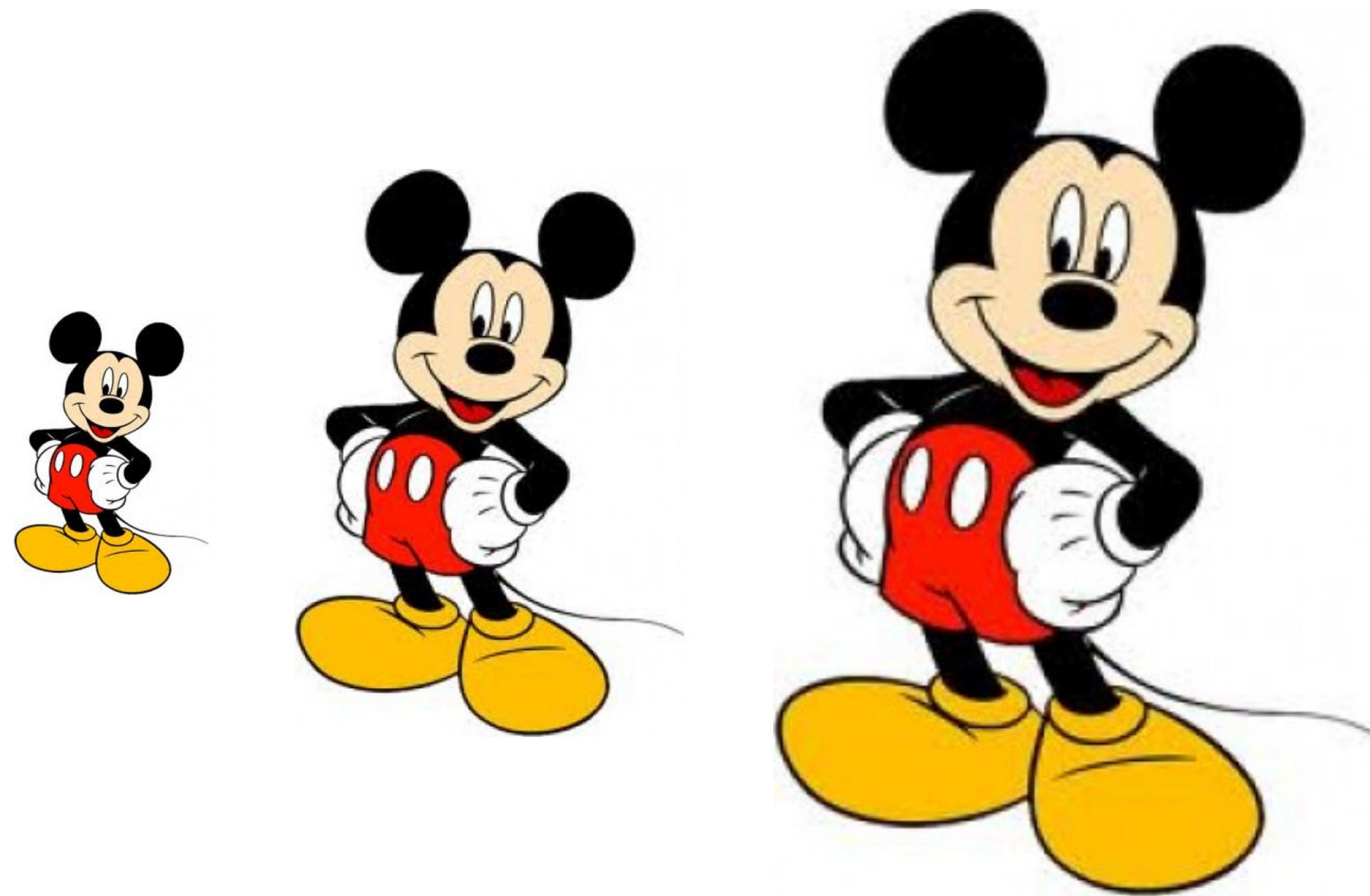




GTK Cyber

# **Visual Encodings: Size**





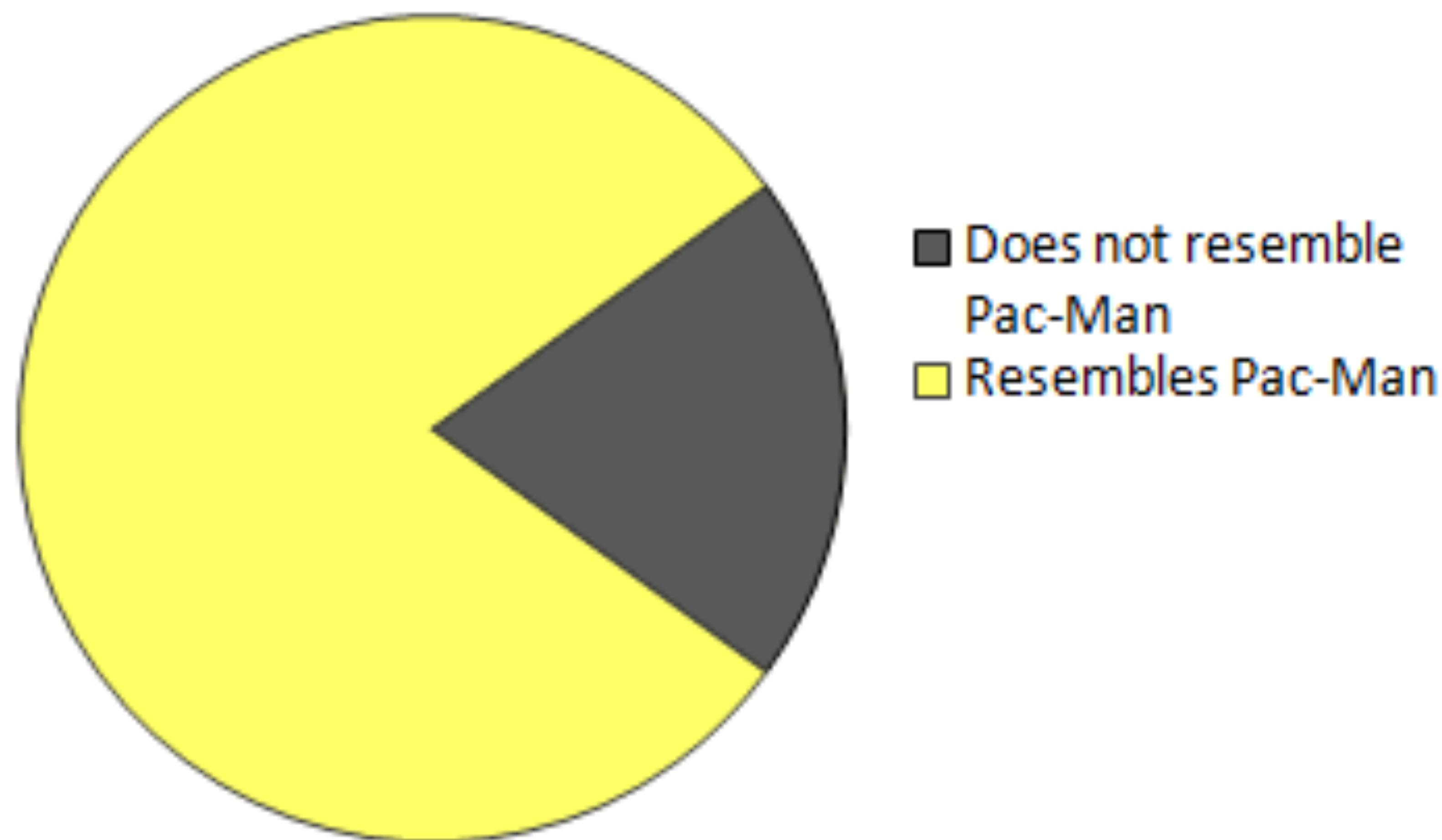
GTK Cyber

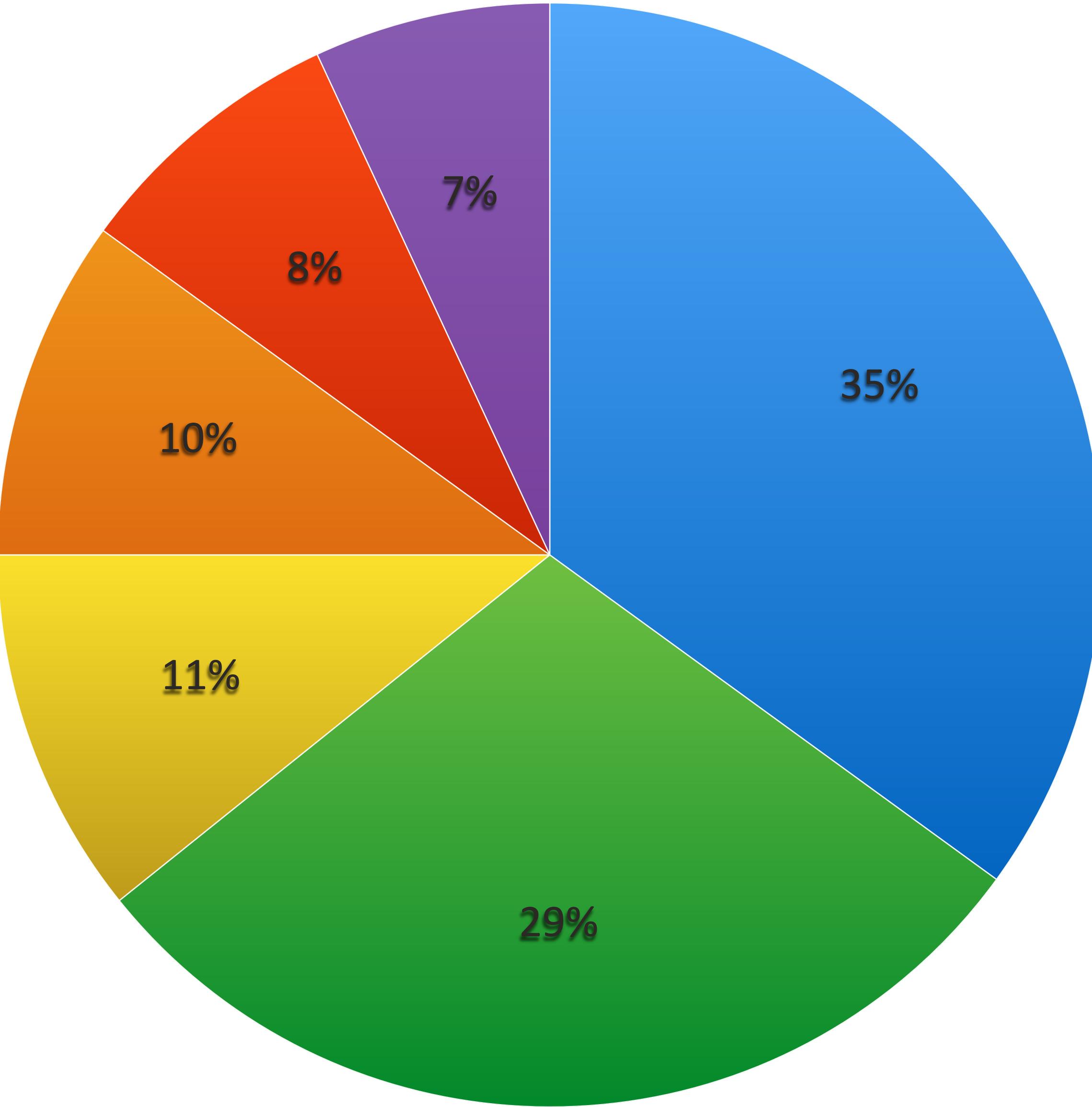


# A Brief Interlude: Why Never to Use a Pie Chart

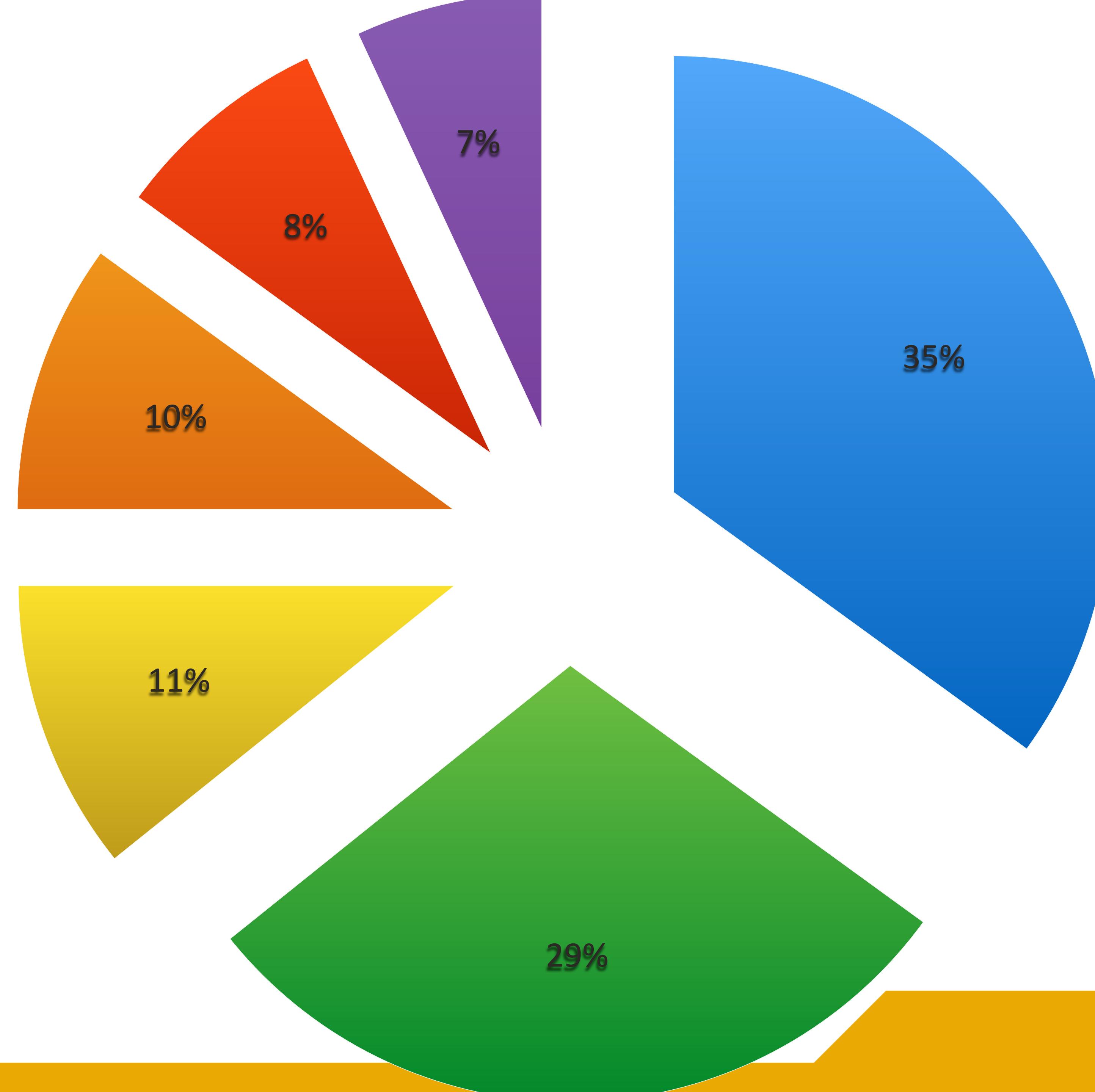


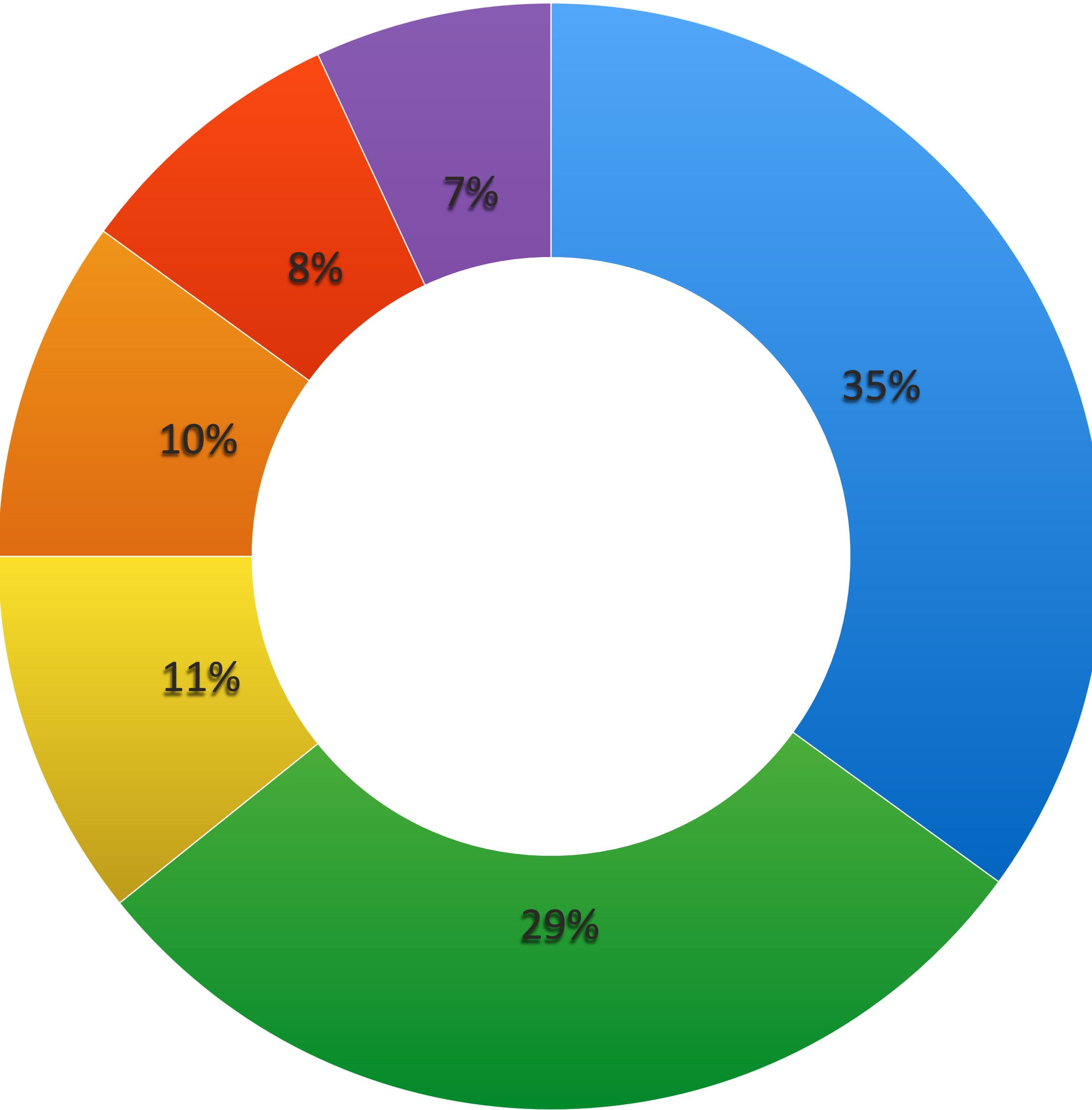
## Percentage of Chart Which Resembles Pac-Man

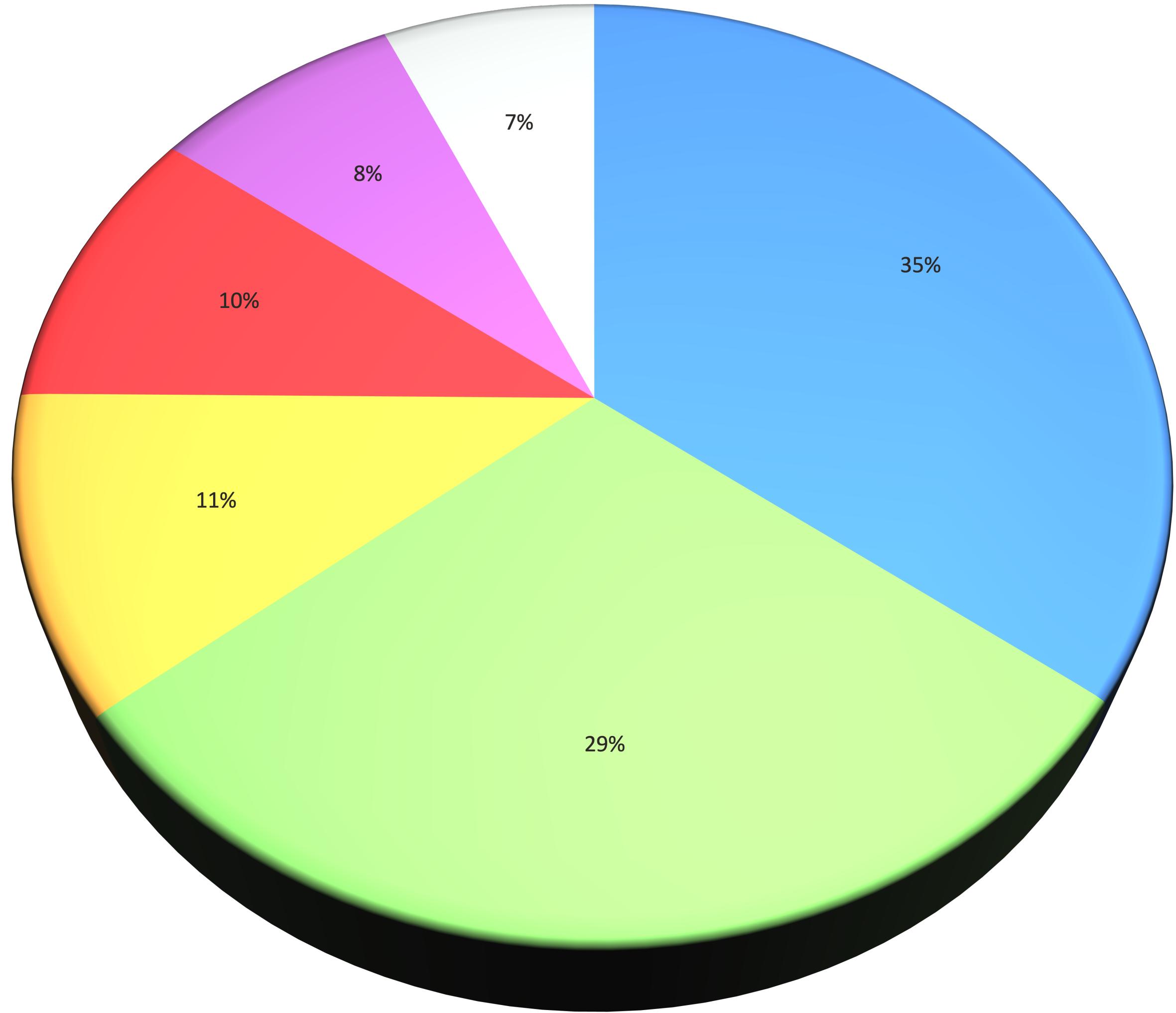




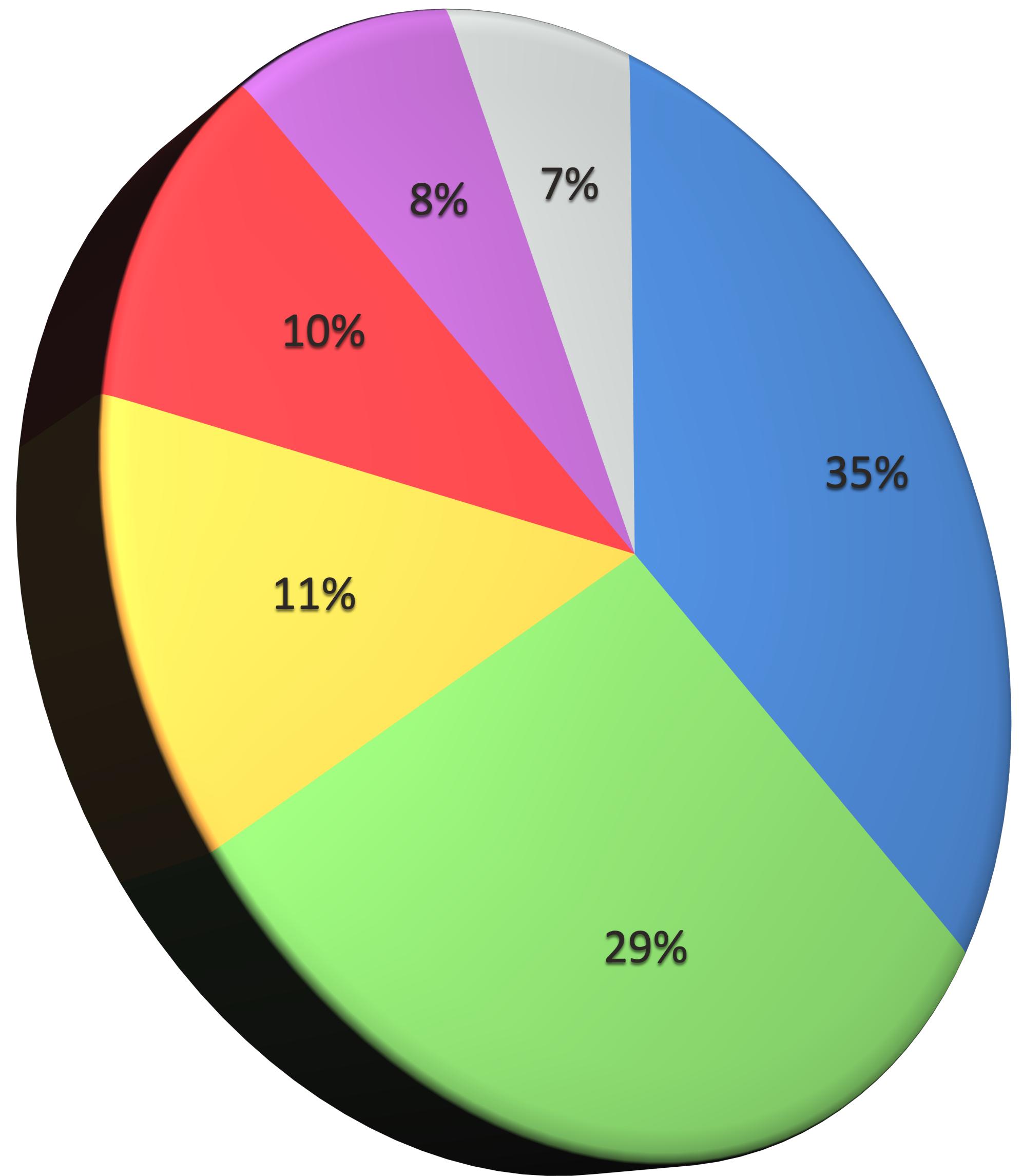
GTK Cyber





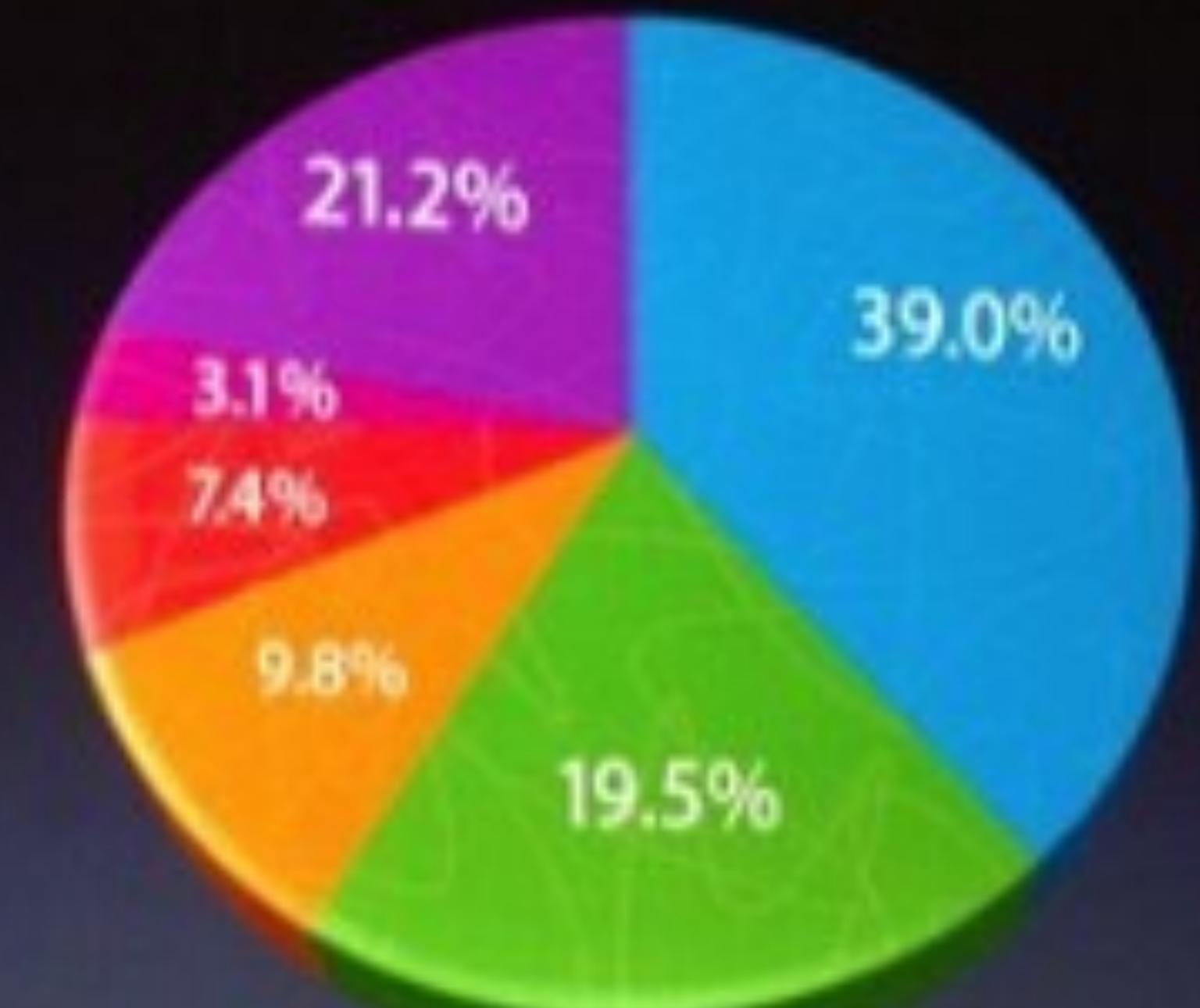


GTK Cyber



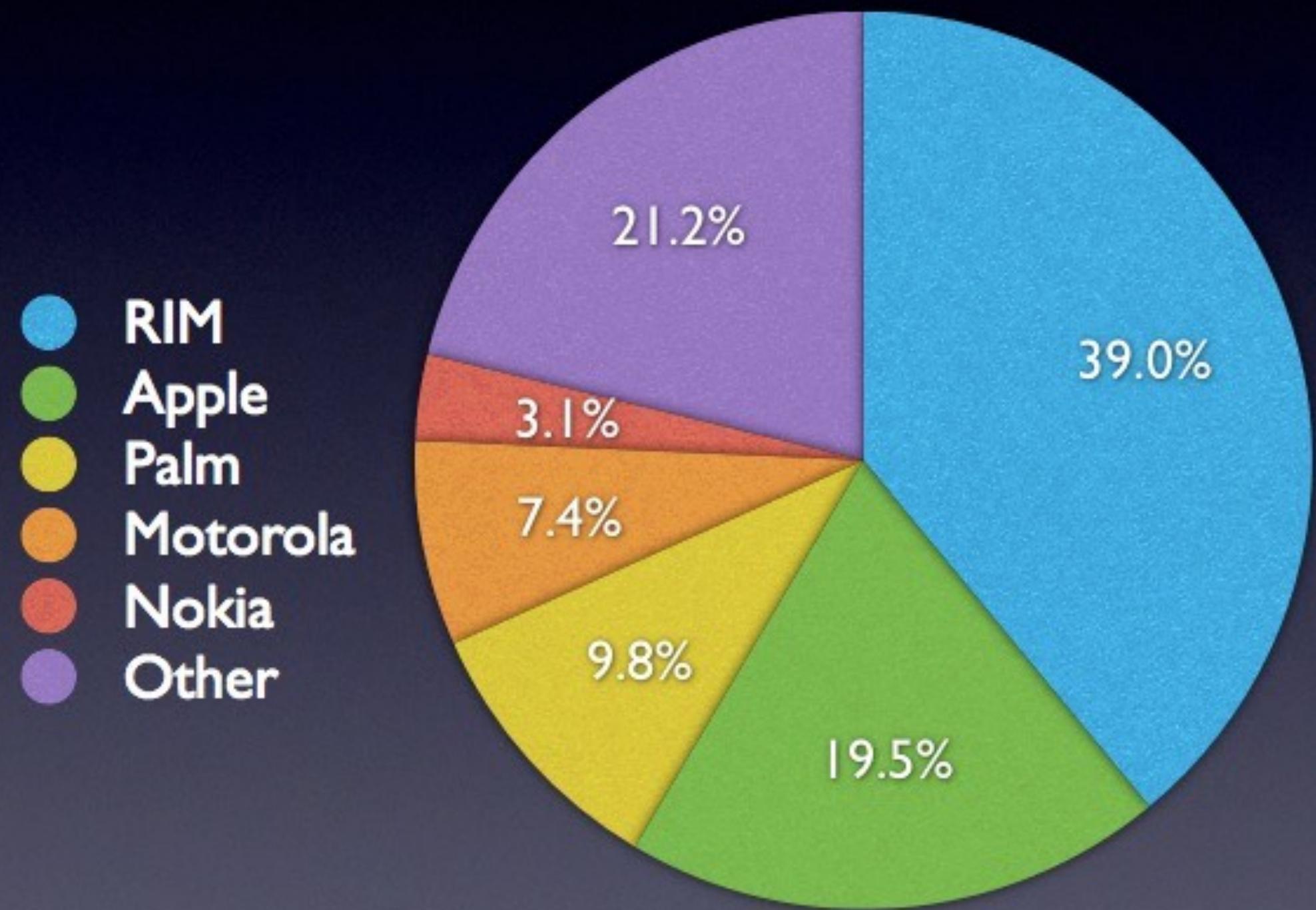
# U.S. SmartPhone Marketshare

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



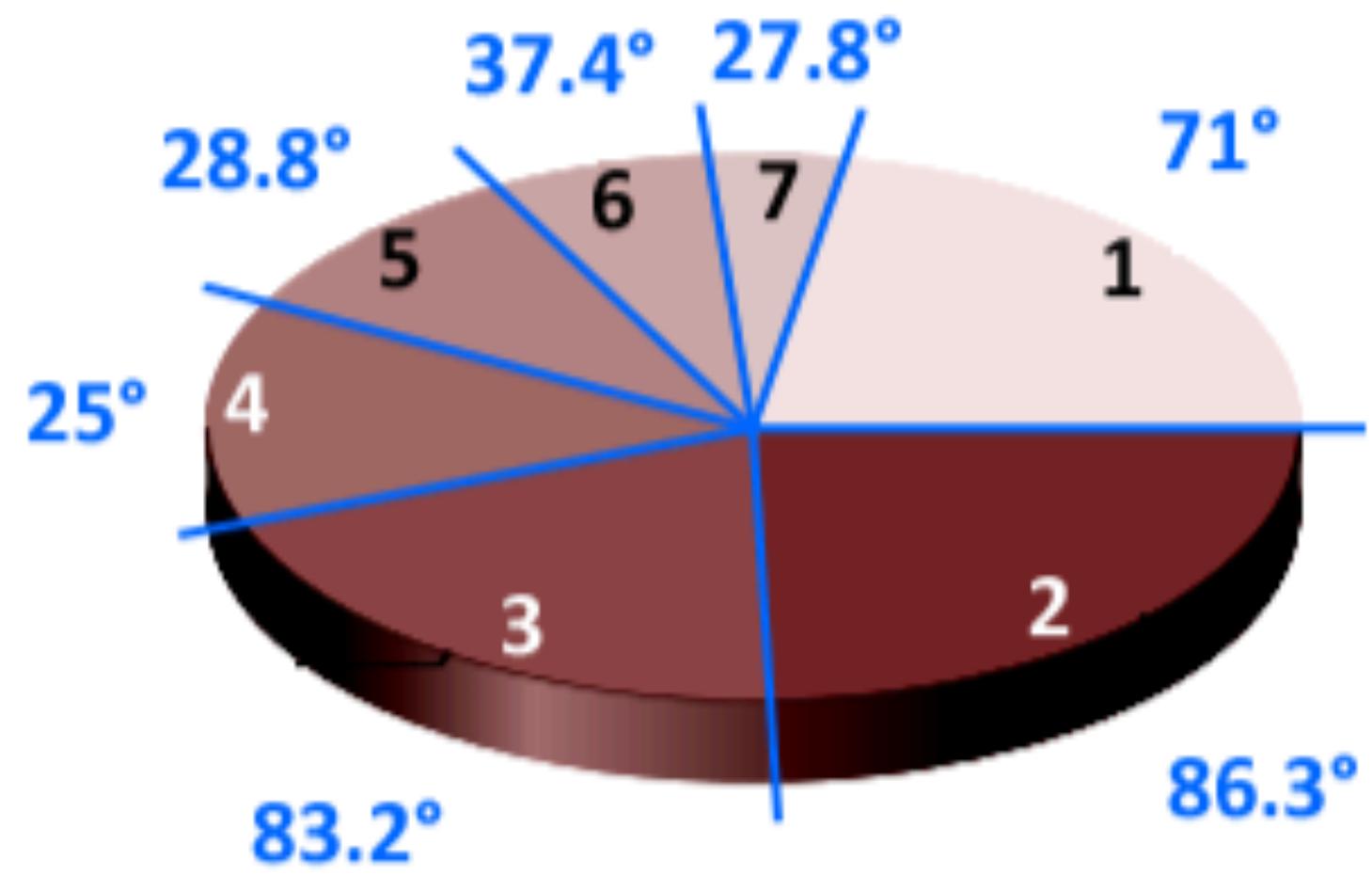
Gartner fo

# U.S. SmartPhone Marketshare

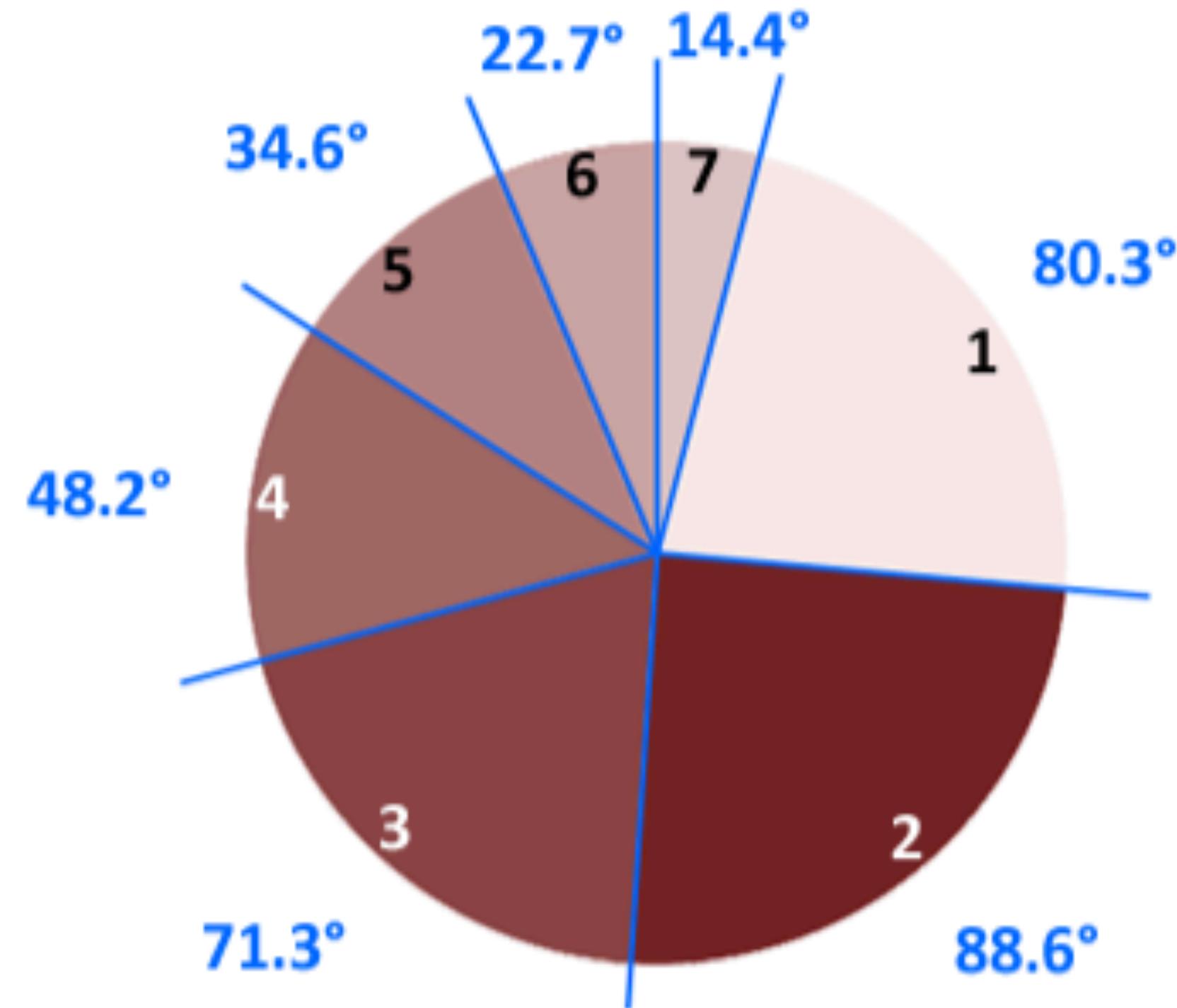


# 3D Pie Charts are BAD!

## 3D Pie Charts Distort Angles



Angles on the original pie chart



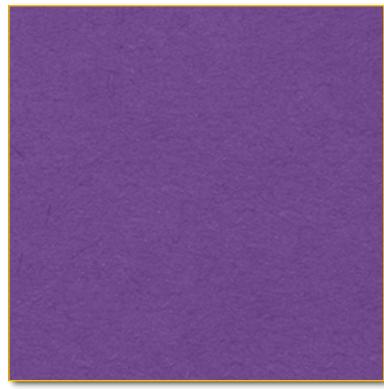
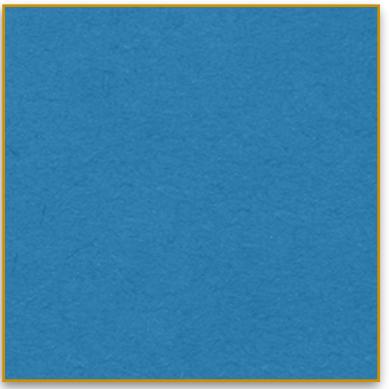
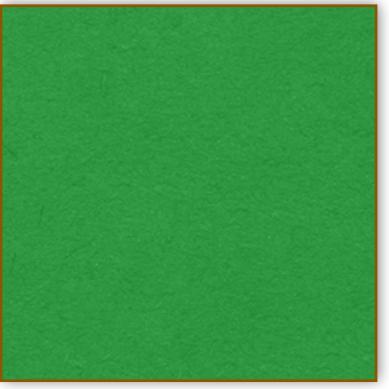
Angles on a non-3D pie chart

# **Visual Encodings: Color**

# Visual Encodings: Color

- **Hue** should be used to encode categorical data
- **Saturation** should be used to encode intensity or a continuous value

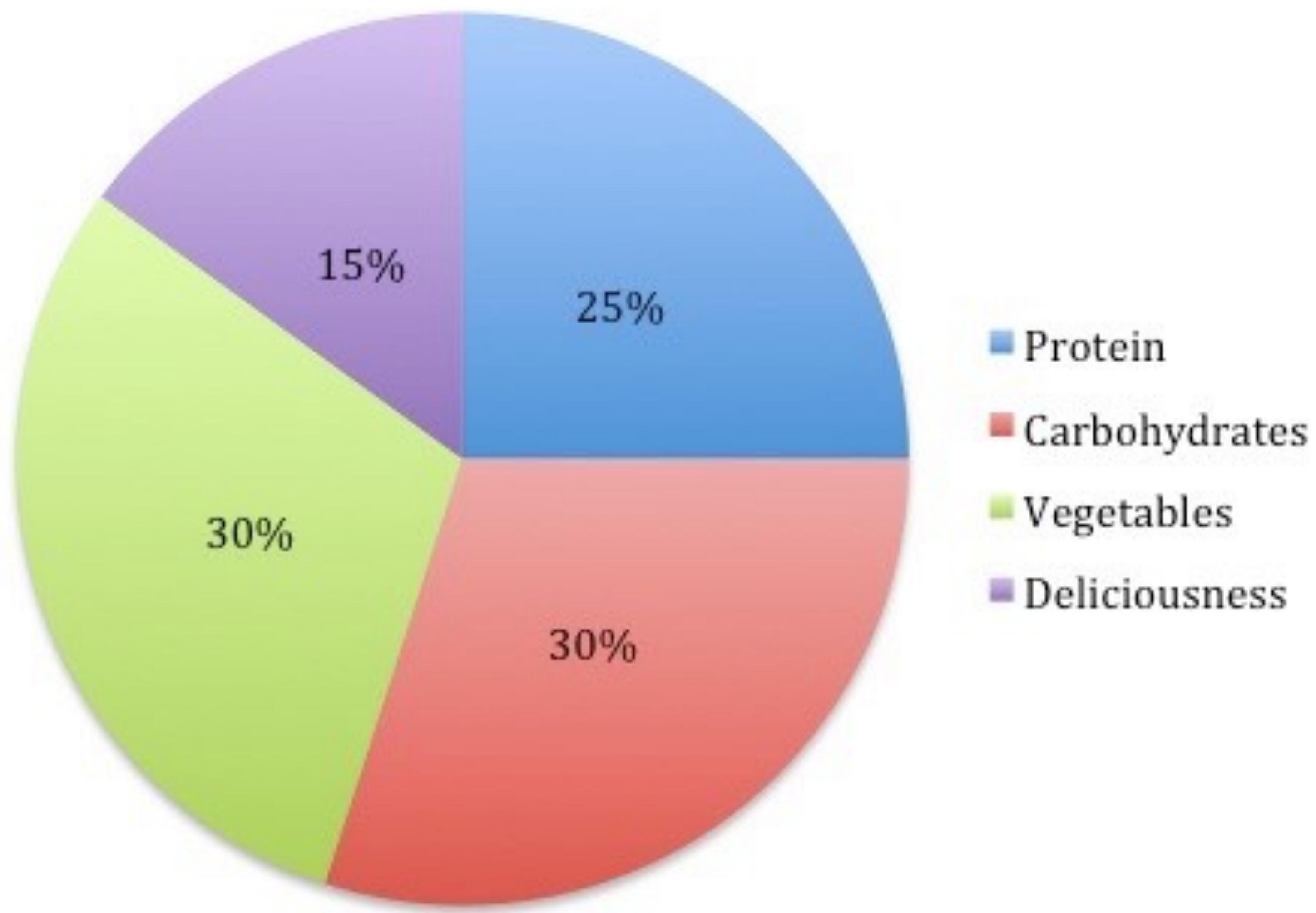
# Hue



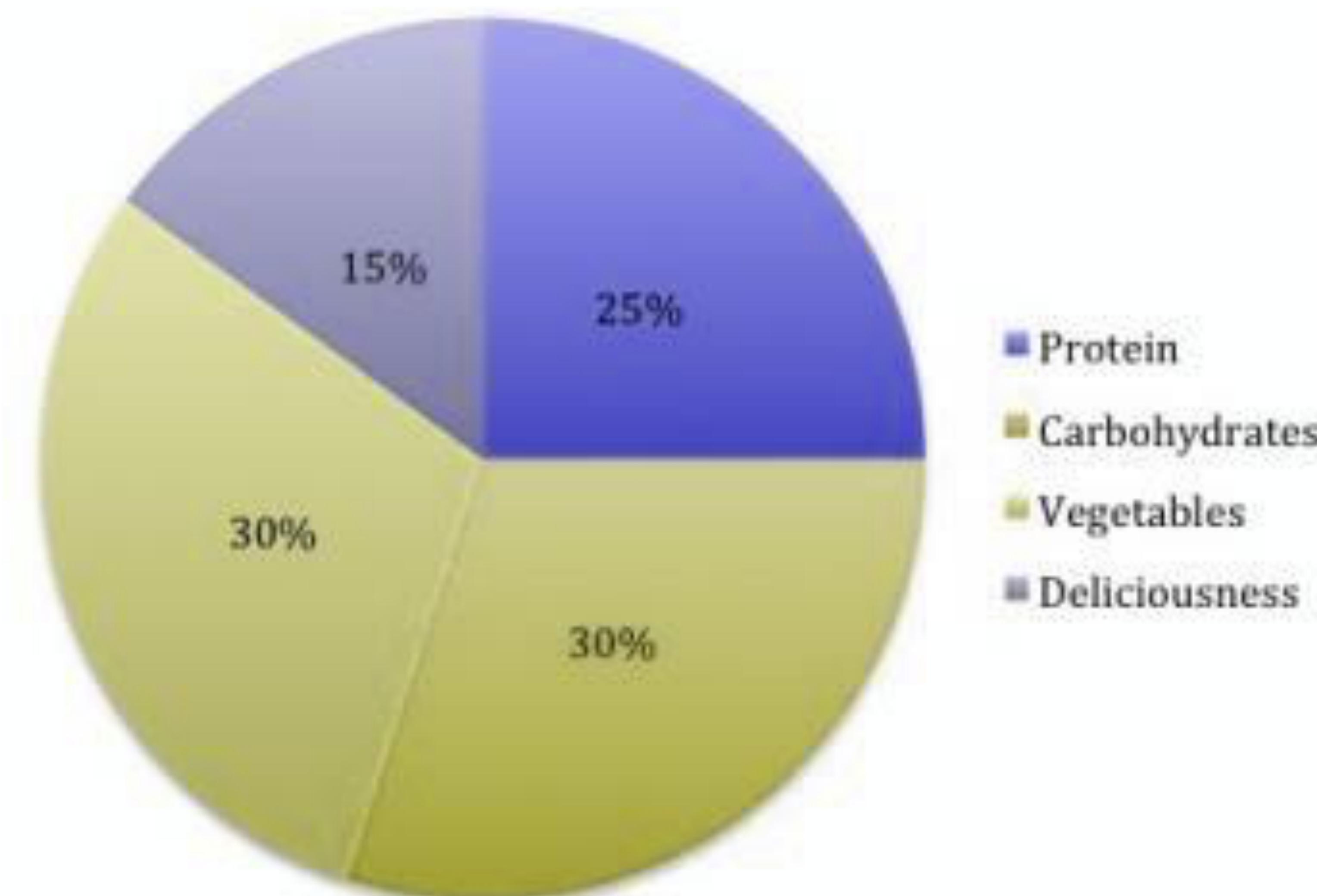
# Saturation



# A Healthy Meal



# A Healthy Meal





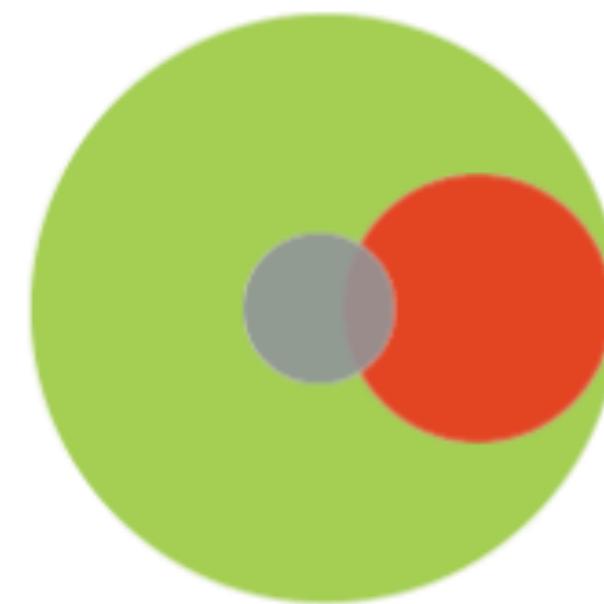
elastica



# Dashboard

## Users

Total (192)



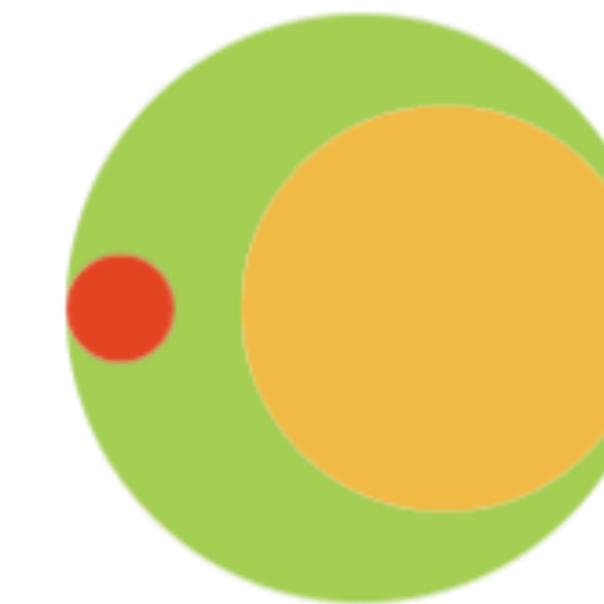
28  
High Risk

0  
Med Risk

4  
Blocked

## Policies

Total (231)



3  
Blocking

142  
Alerting

## Policy Alerts

Alerting



Blocked



Rest



## Threat Alerts

High Risk



Med Risk



Low Risk



## Audited Services

by Users ▾

High Risk (736)

Medium Risk (3k)

Low Risk (3k)



3k  
Users

494.3 GB  
Traffic

963k  
Sessions

243  
Destinations

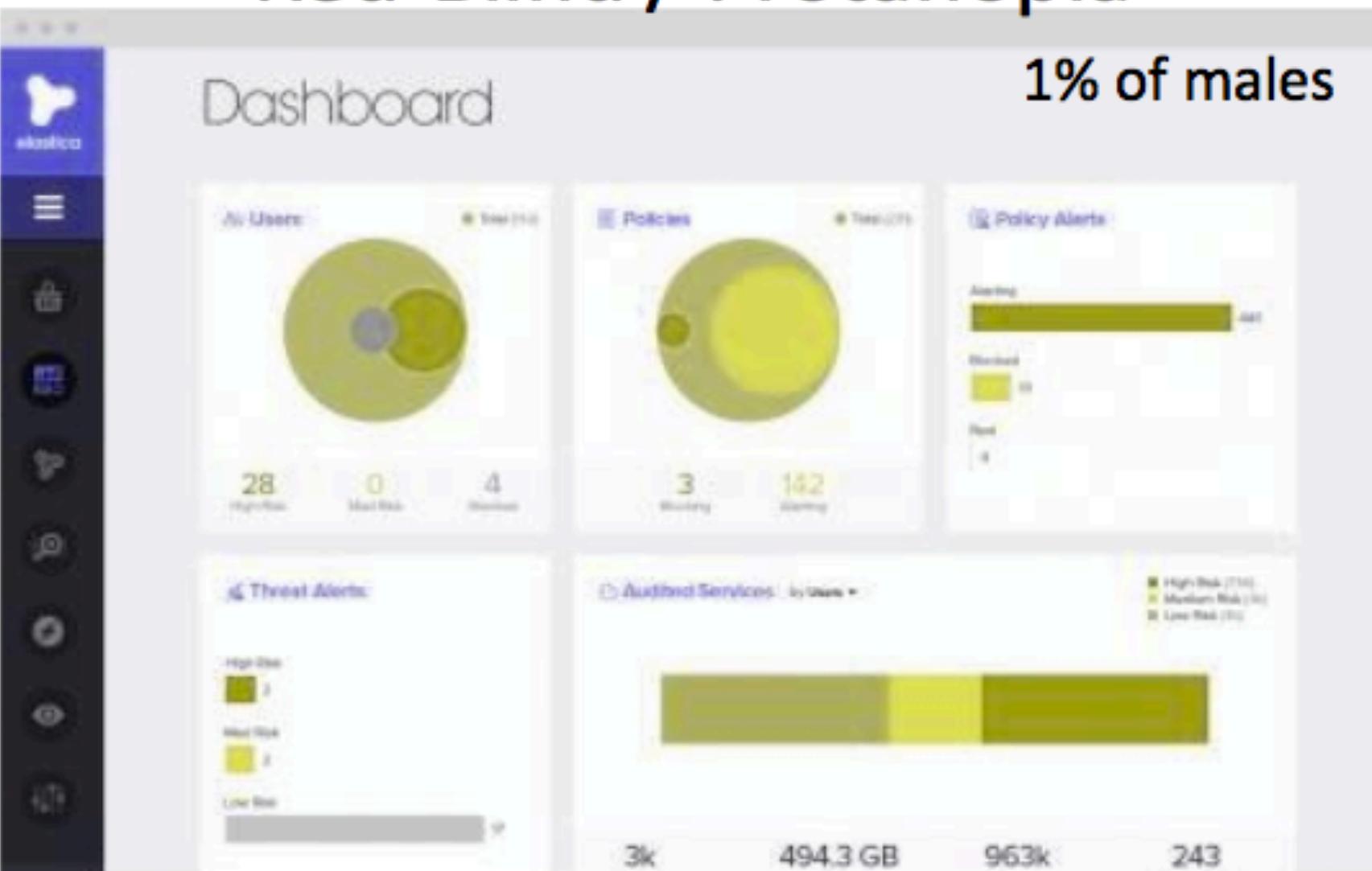
# Original



# Green-Blind / Deutanopia



# Red-Blind / Protanopia



# Blue-Blind / Tritanopia

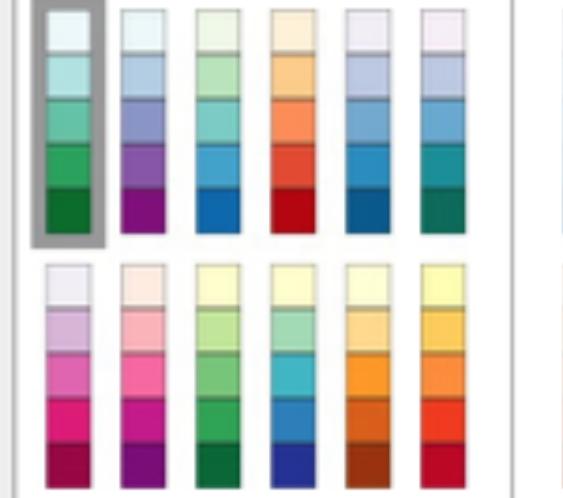


<http://www.color-blindness.com/coblis-color-blindness-simulator/>

# Color

Number of data classes: 3

Nature of your data:  
 sequential  diverging  qualitative

Pick a color scheme:  
Multi-hue:  Single hue: 

Only show:  
 colorblind safe  
 print friendly  
 photocopy safe

Context:  
 roads  
 cities  
 borders

Background:  
 solid color  terrain  
 color transparency

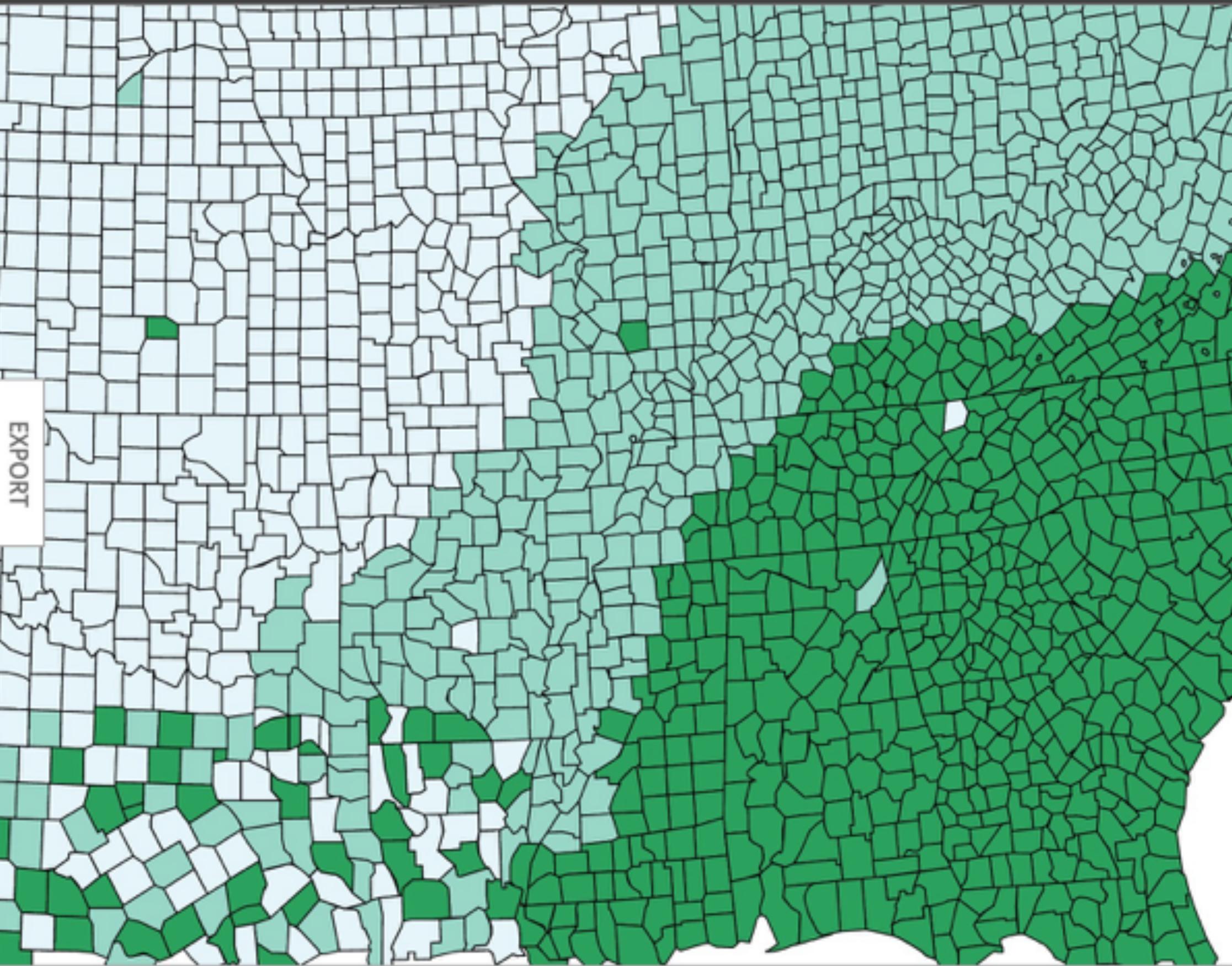
how to use | updates | downloads | credits

**COLORBREWER 2.0**  
color advice for cartography

EXPORT

3-class BuGn

    
HEX   
 #e5f5f9  
 #99d8c9  
 #2ca25f

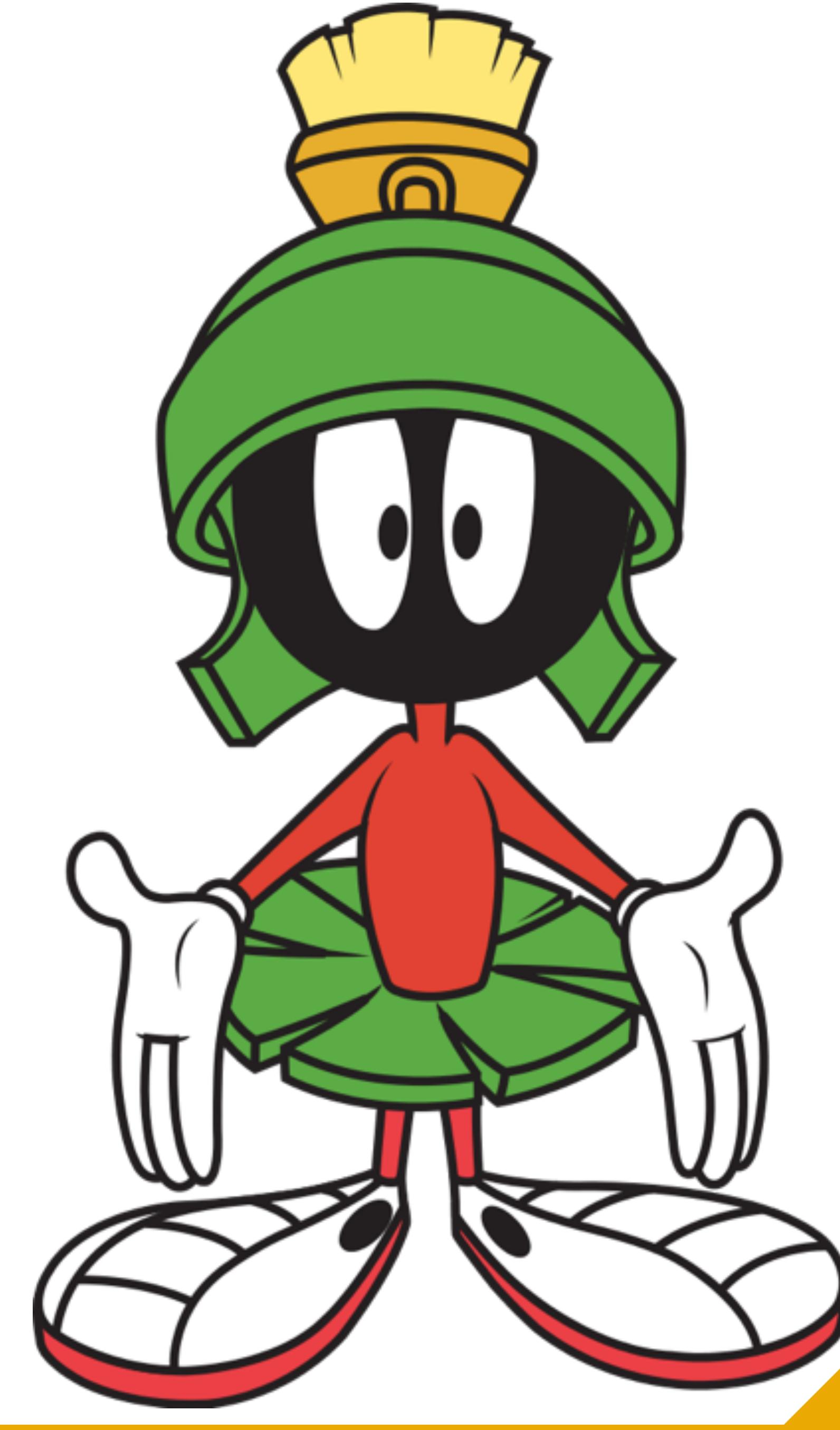


**Brainpower used  
for decoding**

**Brainpower remaining  
for understanding**



**Total brainpower available**



GTK Cyber

*Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.*  
 Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite

Paris, le 20 Novembre 1869.

Les nombres d'hommes perdus sont représentés par les largesses 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 ont péri en Russie; le noir ceux qui en sont sortis. — Les renseignements qui ont servi à dresser la carte ont été pris dans les ouvrages de M. M. Chiers, de Séguir, de Fezensac, de Chambray et le journal intime de Jacob, pharmacien de l'Armée depuis le 28 Octobre.

Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davout, qui avaient été détachés sur Minsk et Mohilow et qui rejoignirent Oroscha et Wiléïsk, avaient toujours marché avec l'armée.

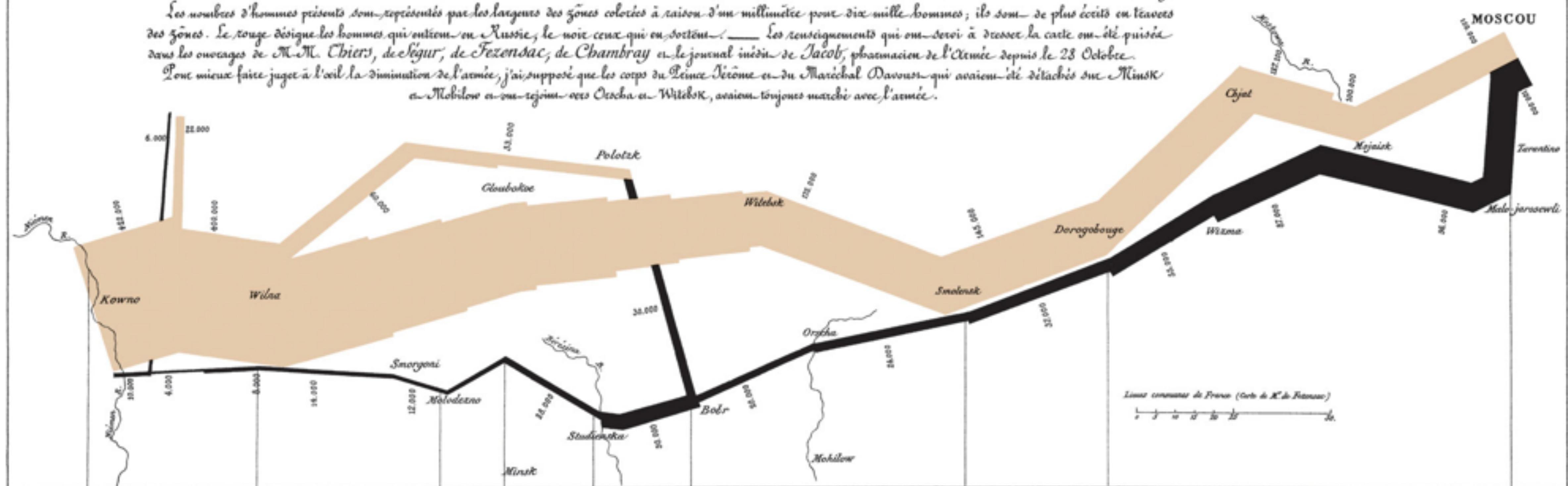
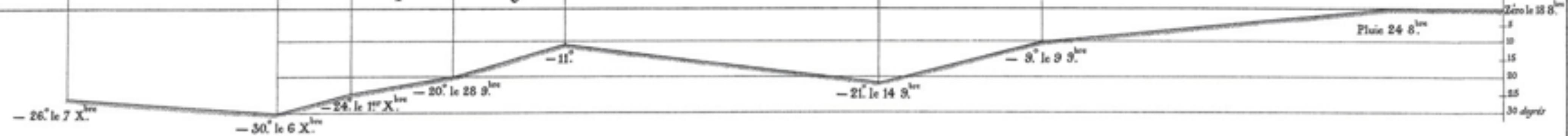


TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.

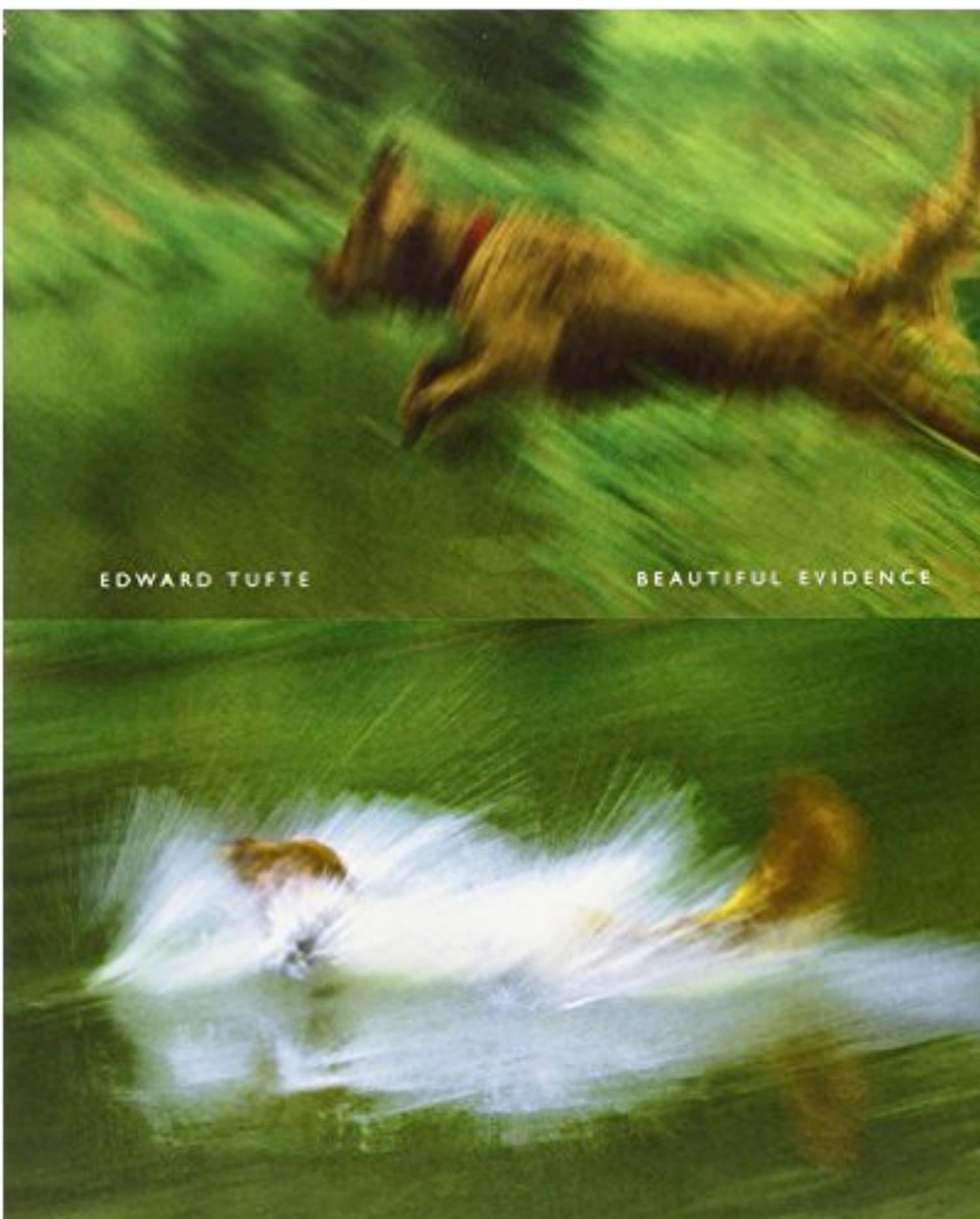
Les cosaques passent au galop  
le Niemen gelé.



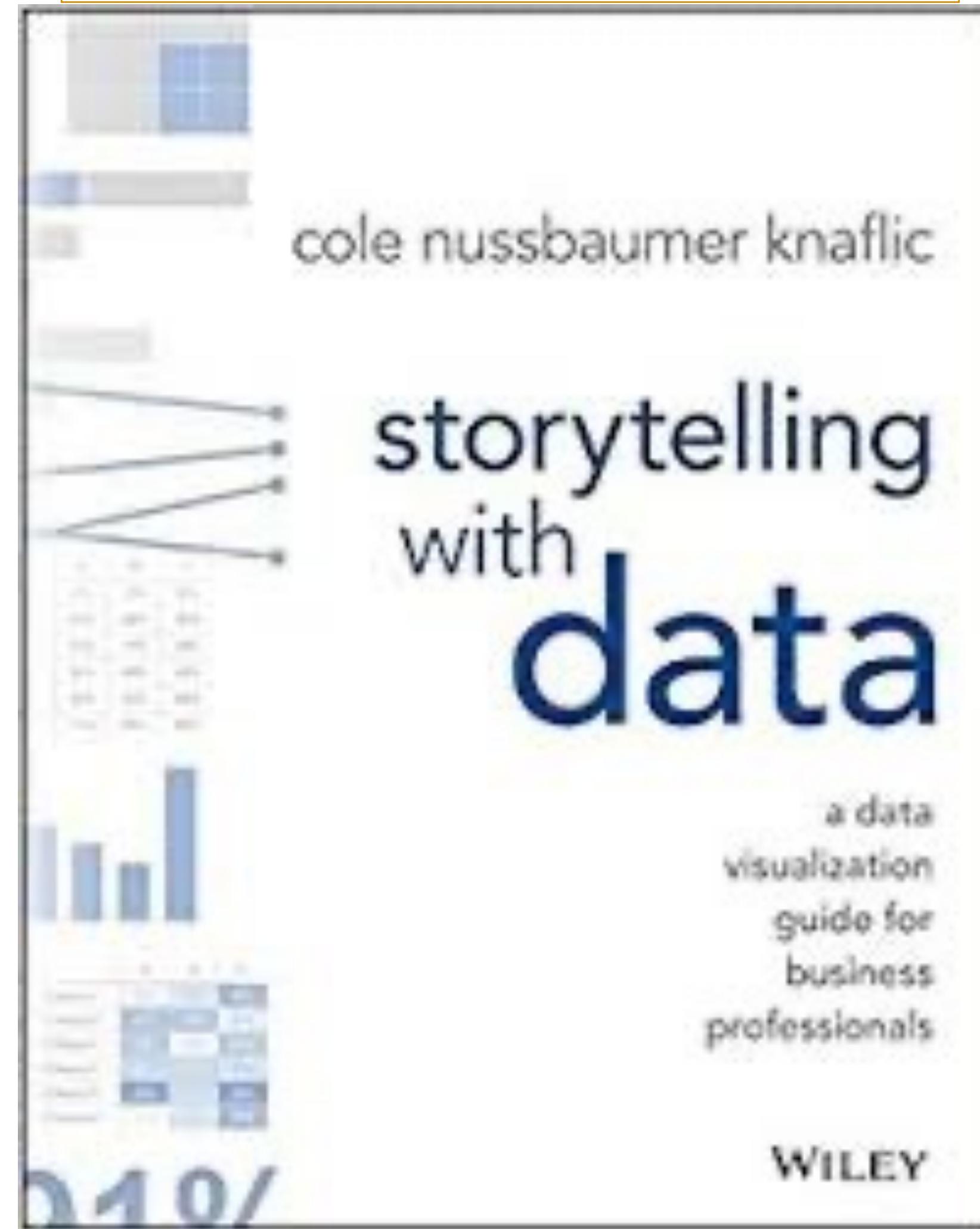
Autog. par Regnier, 8. Rue J<sup>e</sup> Marie 3<sup>e</sup> ét<sup>e</sup> à Paris.

Imp. Litt. Regnier et Desord.

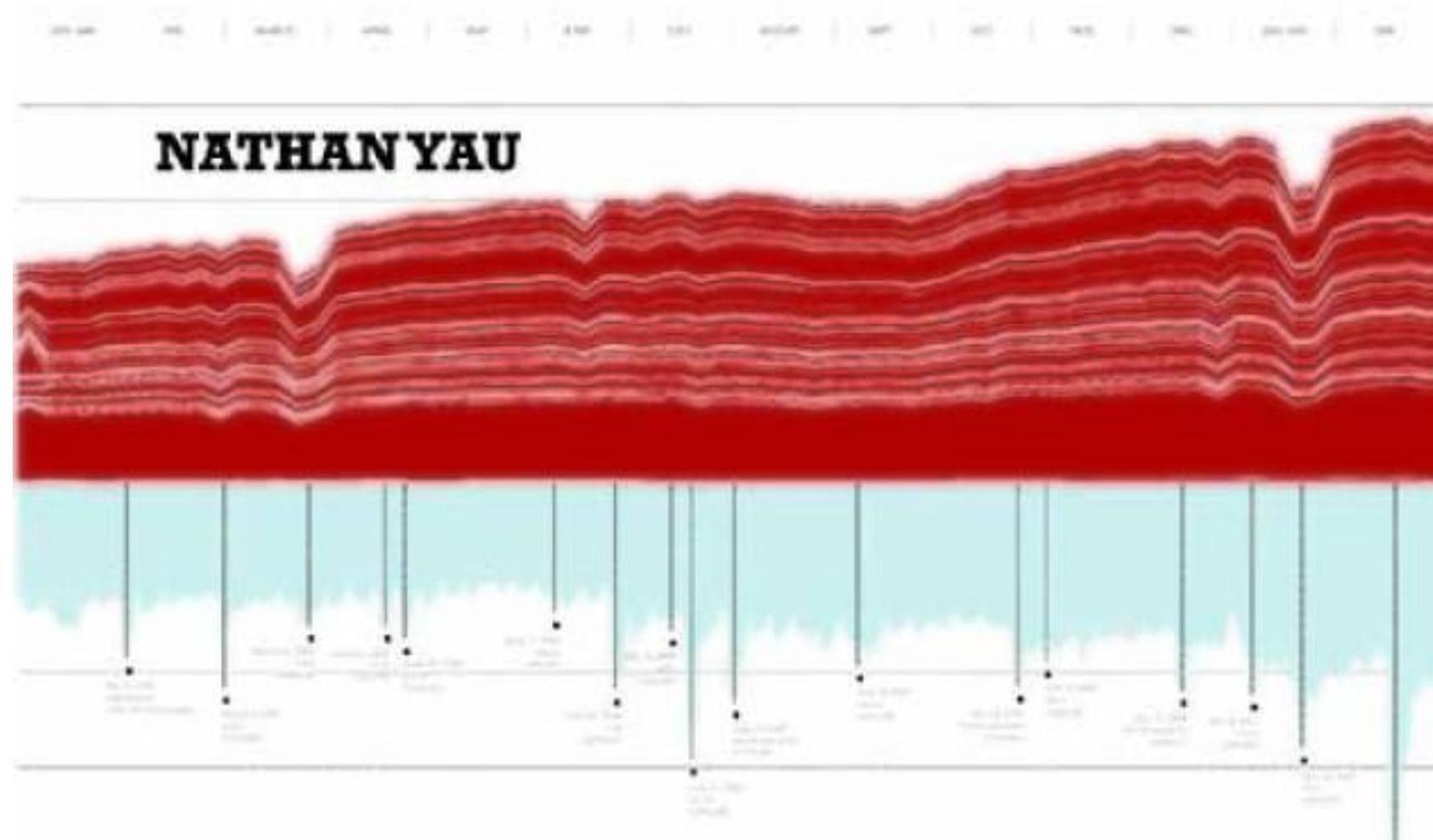
# Recommended Reading



# Recommended Reading



# Recommended Reading



## VISUALIZE THIS

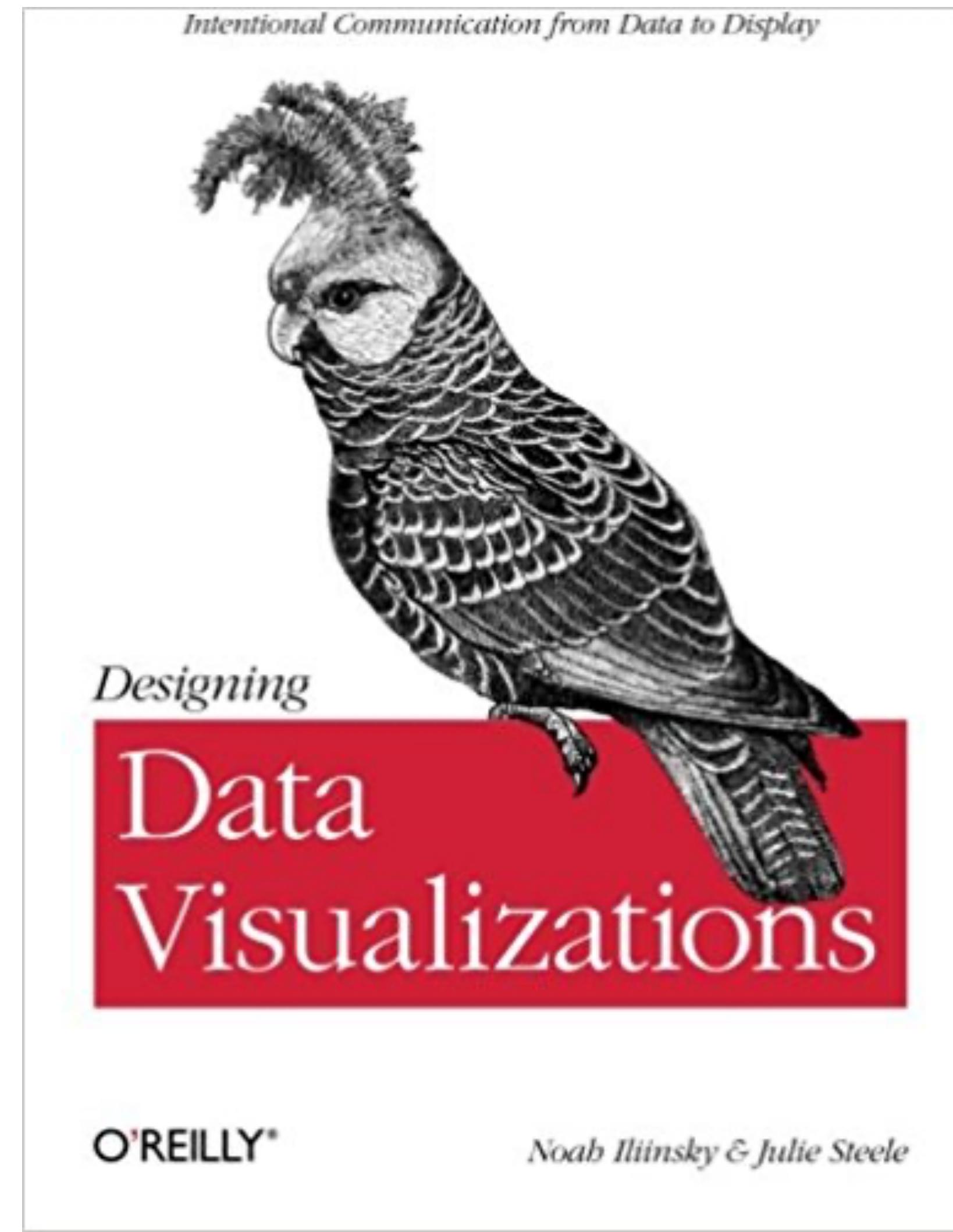
The FlowingData Guide to Design, Visualization, and Statistics



WILEY

GTK Cyber

# Recommended Reading



# Data Visualization in Python

# Python Visualization Libraries

- **Matplotlib**: “Lowest” level visualization library. Very powerful, but little abstraction
- **Pandas/Pyplot**: Easier, but requires transformation for complex visualizations
- **Seaborn**: Good for advanced statistical visualizations
- **Altair**: New library: declarative statistical visualization library
- **Plotly / Cufflinks**: Make it interactive!

# Other Visualization Libraries



**Panel**

- **Panel:** Powerful library for creating apps and dashboards



**hvPlot**

- **hvPlot:** Quickly generate interactive plots from data
- **HoloViews:** Helps you make all data instantly visualizable
- **GeoViews:** HoloViews for geo-spatial data



**HoloViews**

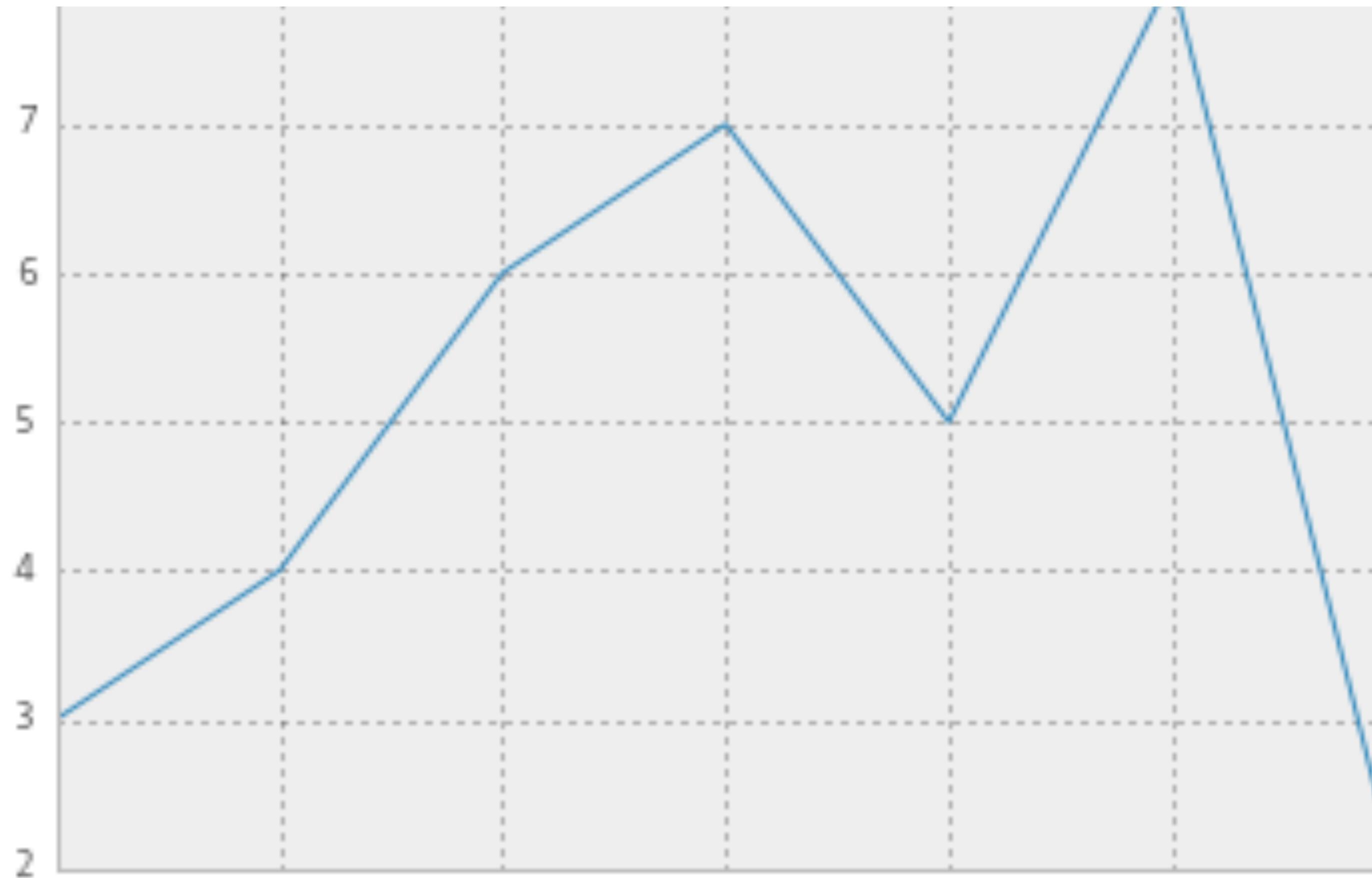
- **ggplot2/PlotNine:** This is Python's implementation of R's ggplot2
- For more useful info about Python visualization, check out [pyviz.org](https://pyviz.org)



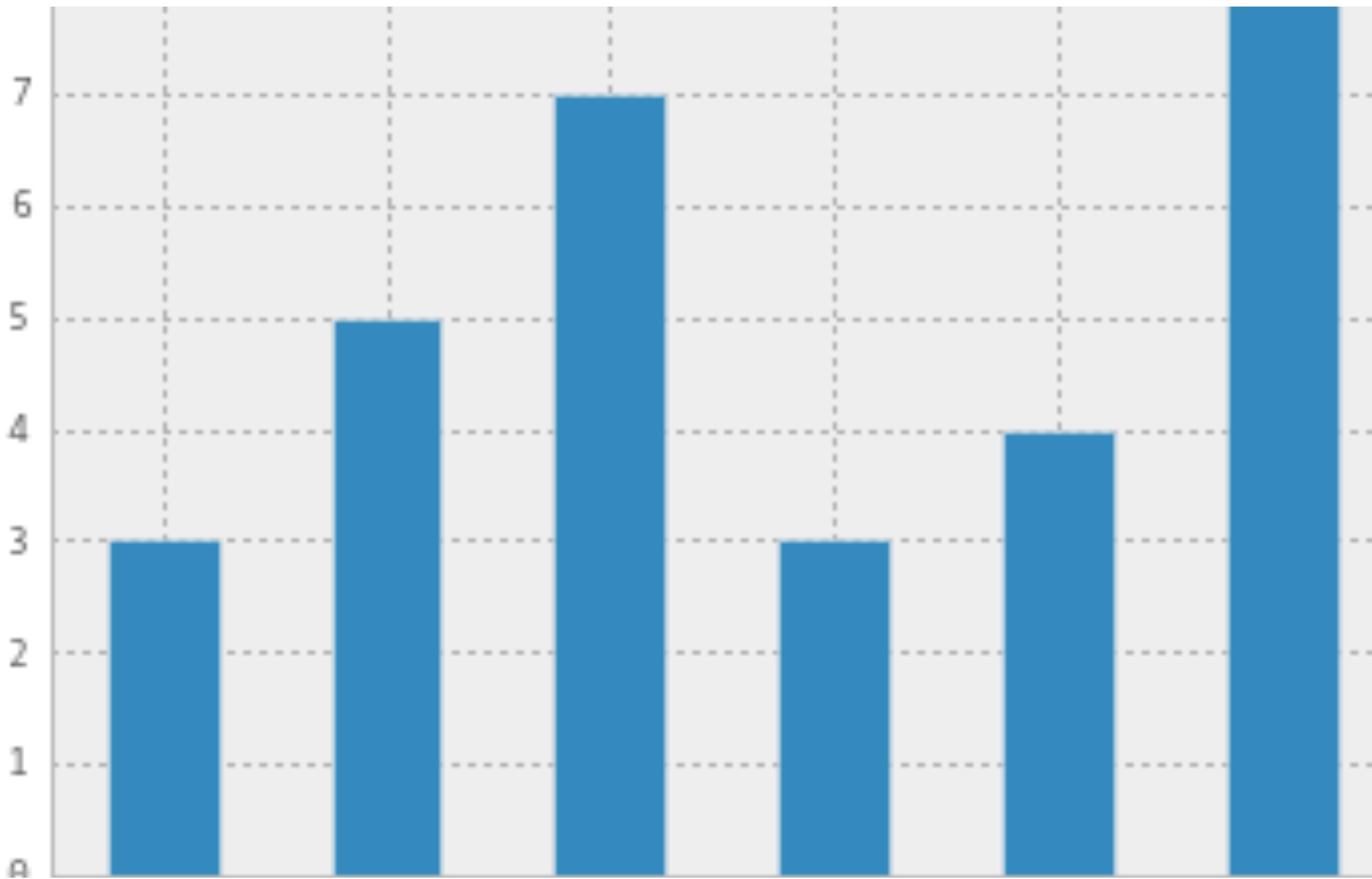
**GeoViews**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
```

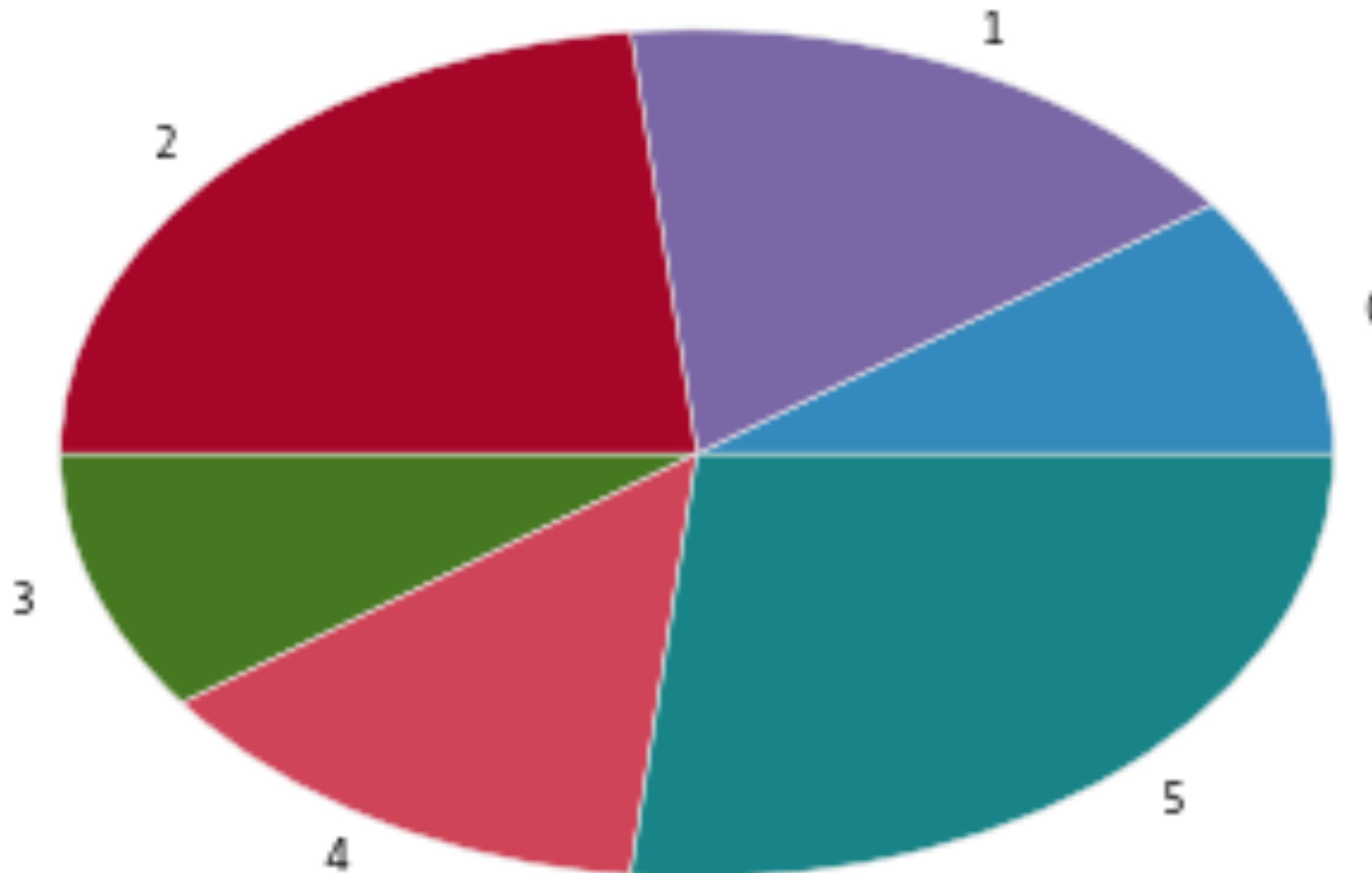
```
data = pd.Series( [3,4,6,7,5,8,2] )  
graph = data.plot()
```

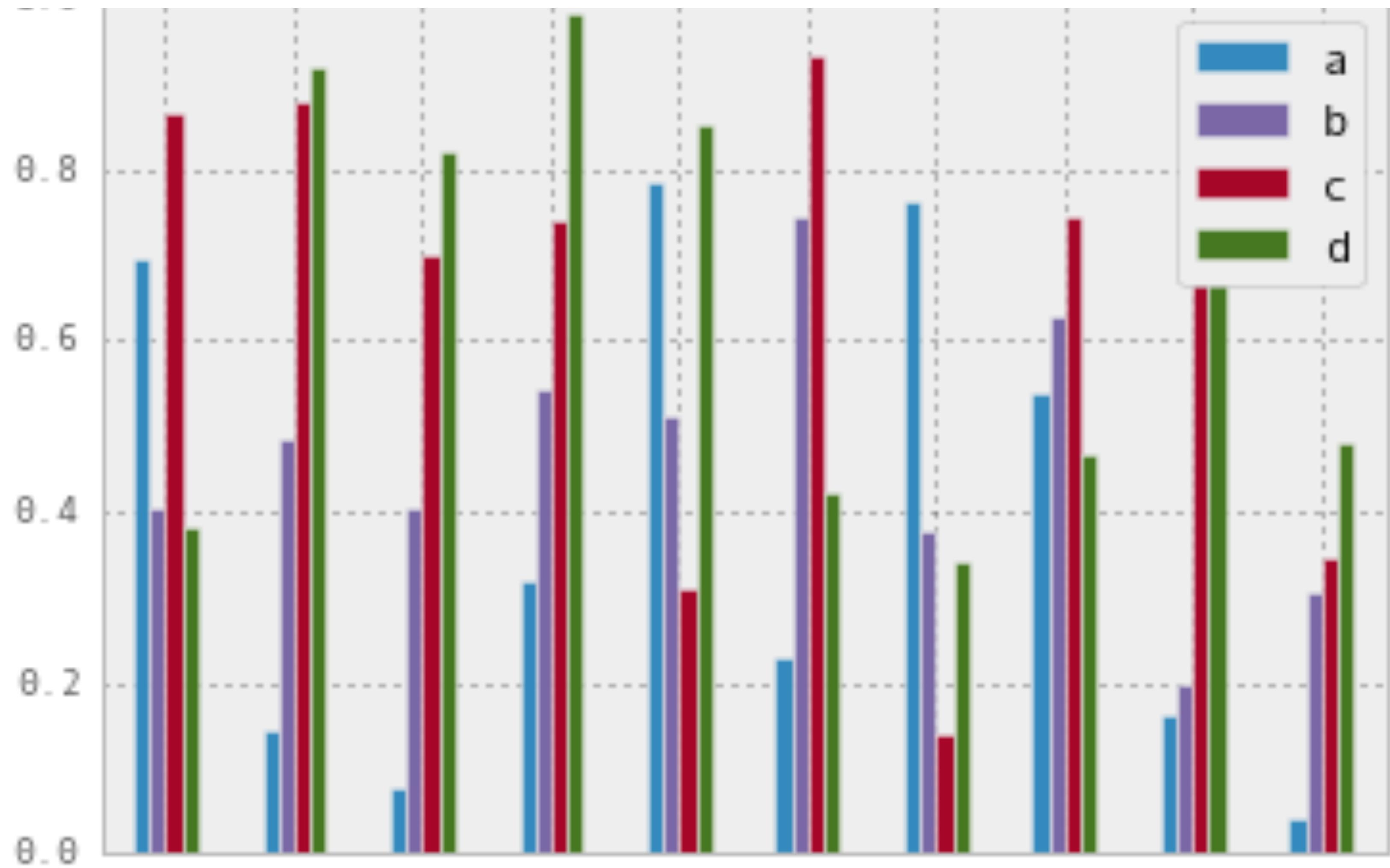


```
barchart = data.plot( kind="bar" )
```



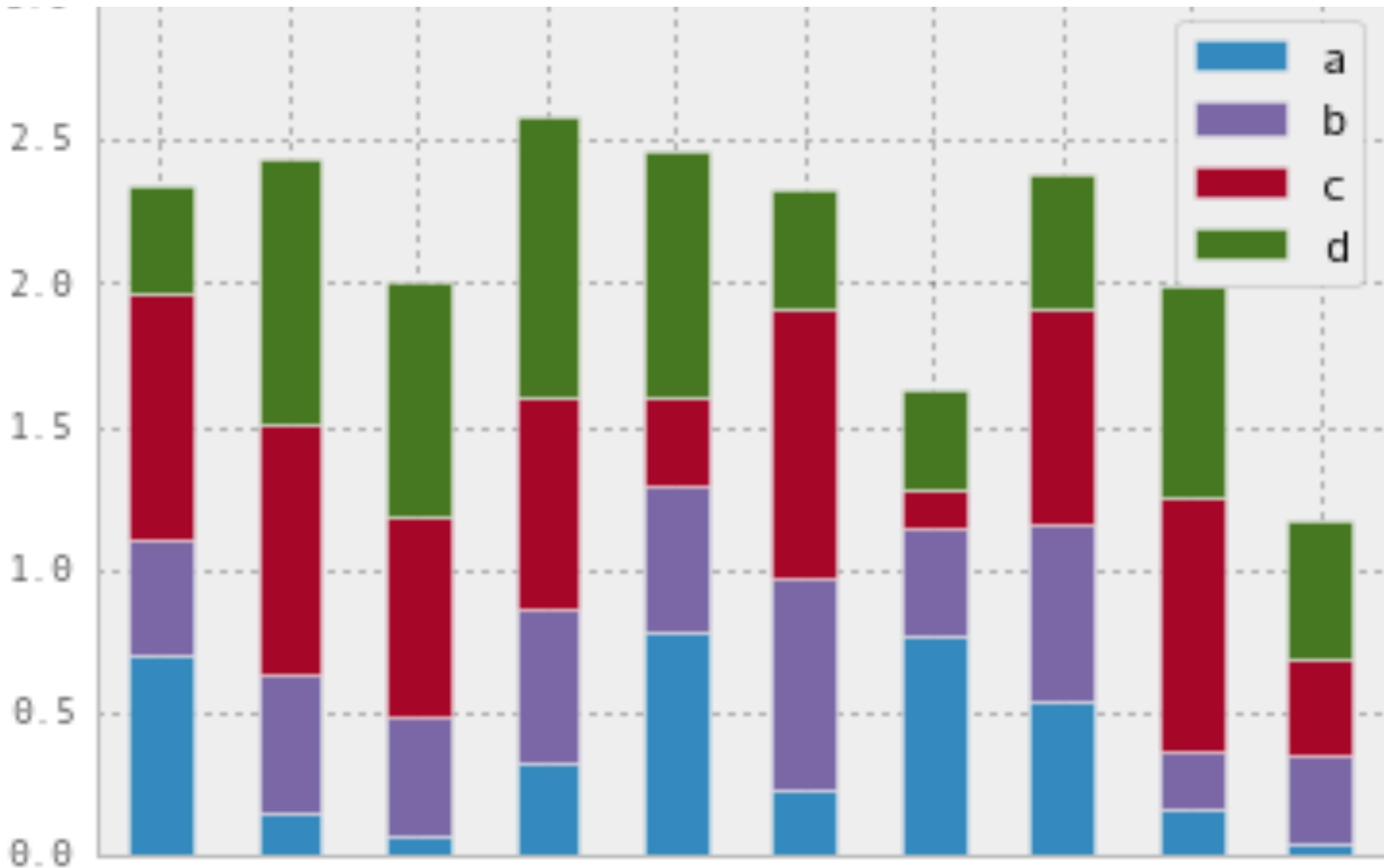
```
piechart = data.plot( kind="pie" )
```



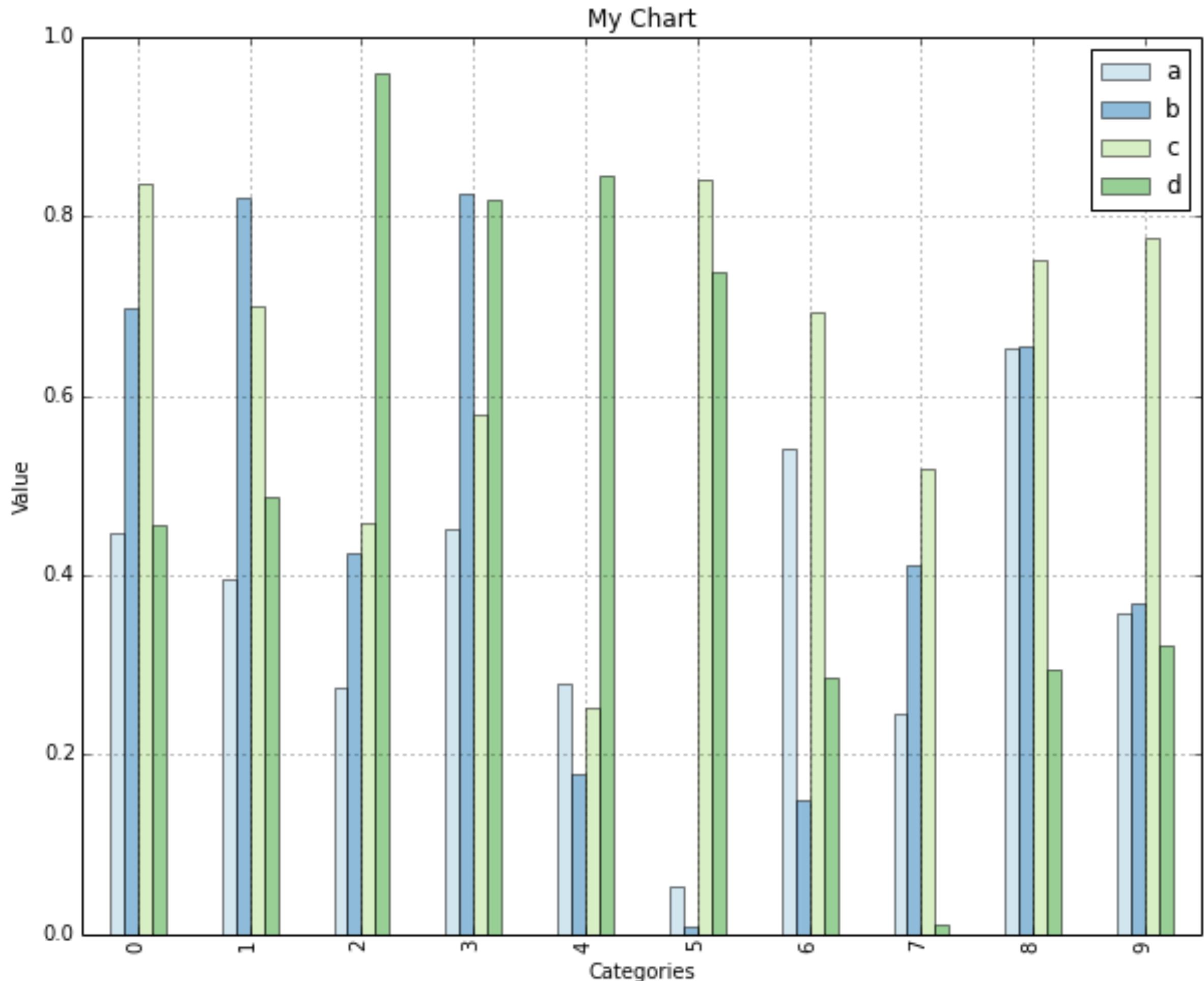


```
df2 = pd.DataFrame(np.random.rand(10, 4), \
columns=[ 'a' , 'b' , 'c' , 'd' ] )
```

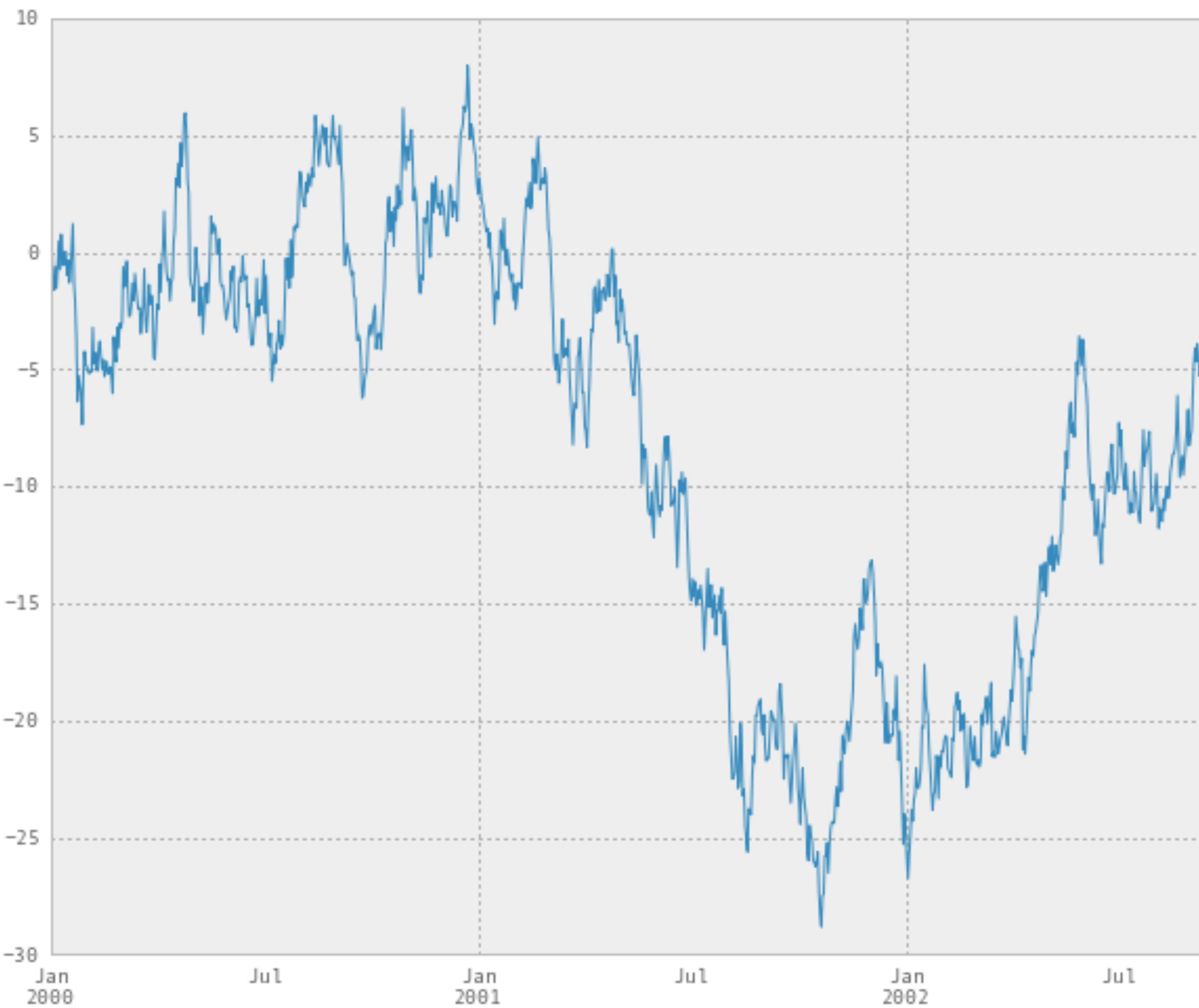
```
df2.plot( kind='bar' )
```



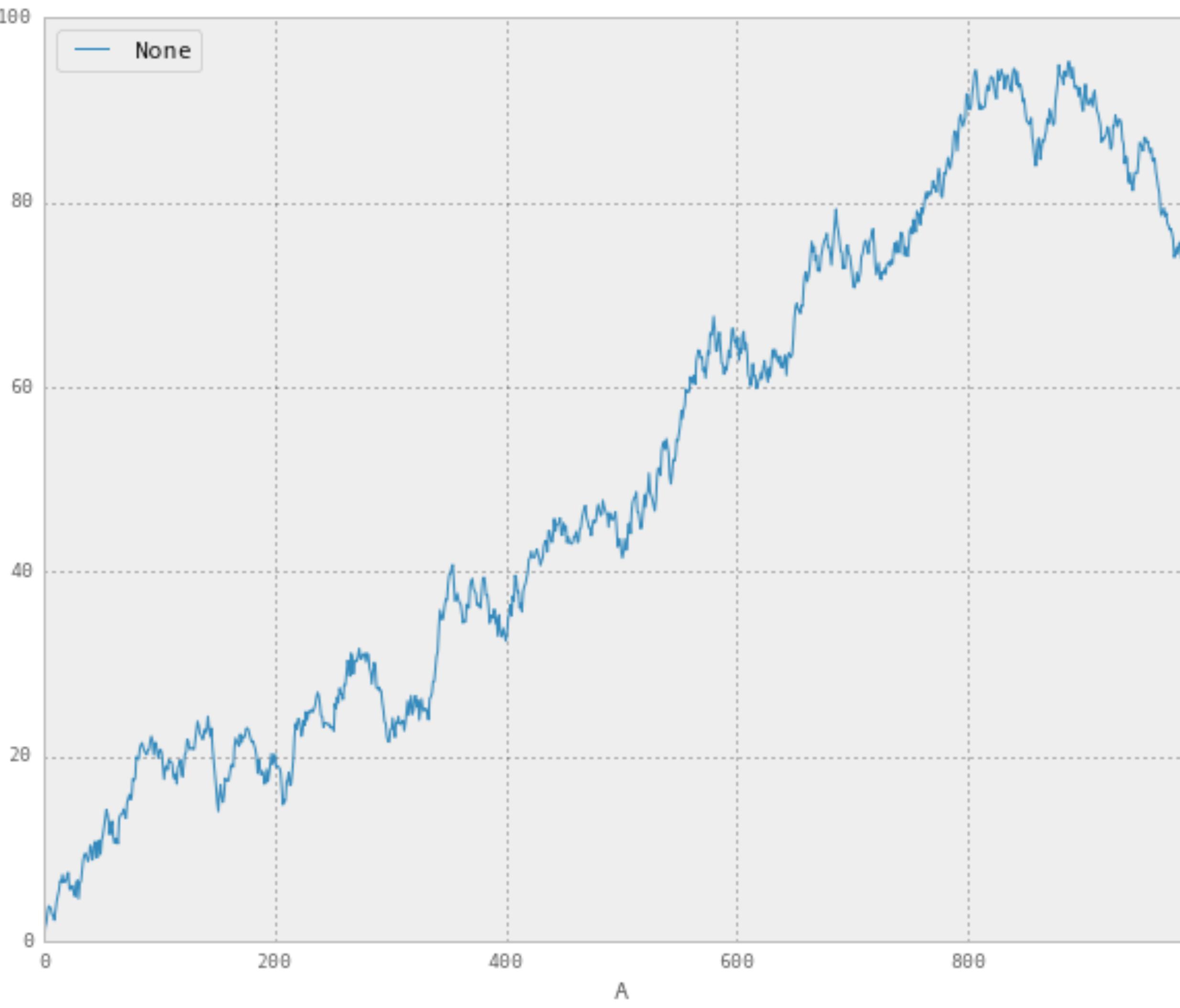
```
df2.plot( kind='bar' , stacked=True )
```



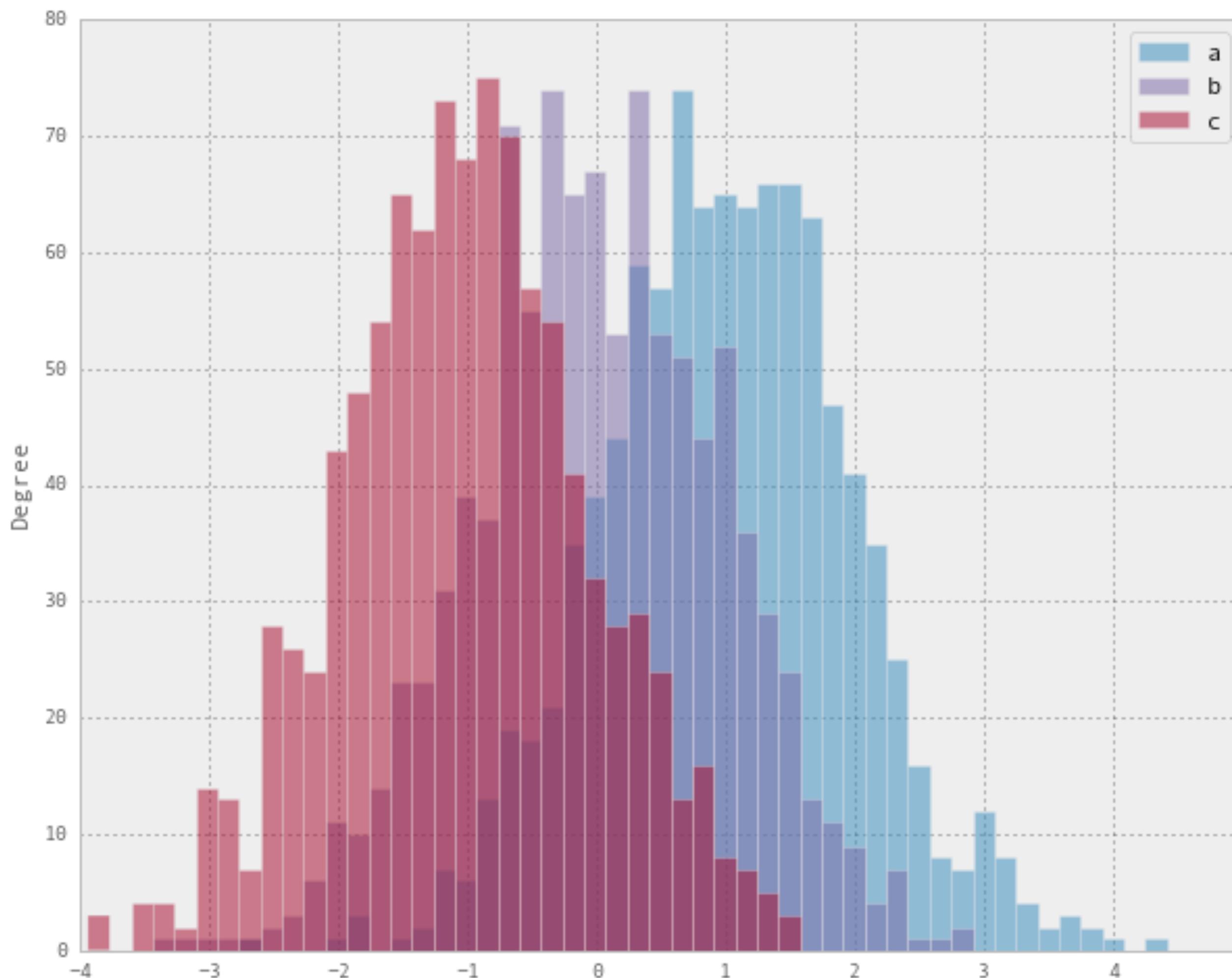
```
df2.plot( kind='bar' ,  
          color=( '#a6cee3' , '#1f78b4' , '#b2df8a' , '#33a02c' ) ,  
          alpha=0.5 ,  
          width=0.5 ,  
          figsize=( 10 , 8 ))  
plt.title( "My Chart" )  
plt.xlabel( "Categories" )  
plt.ylabel( "Value" )
```



```
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) )
```

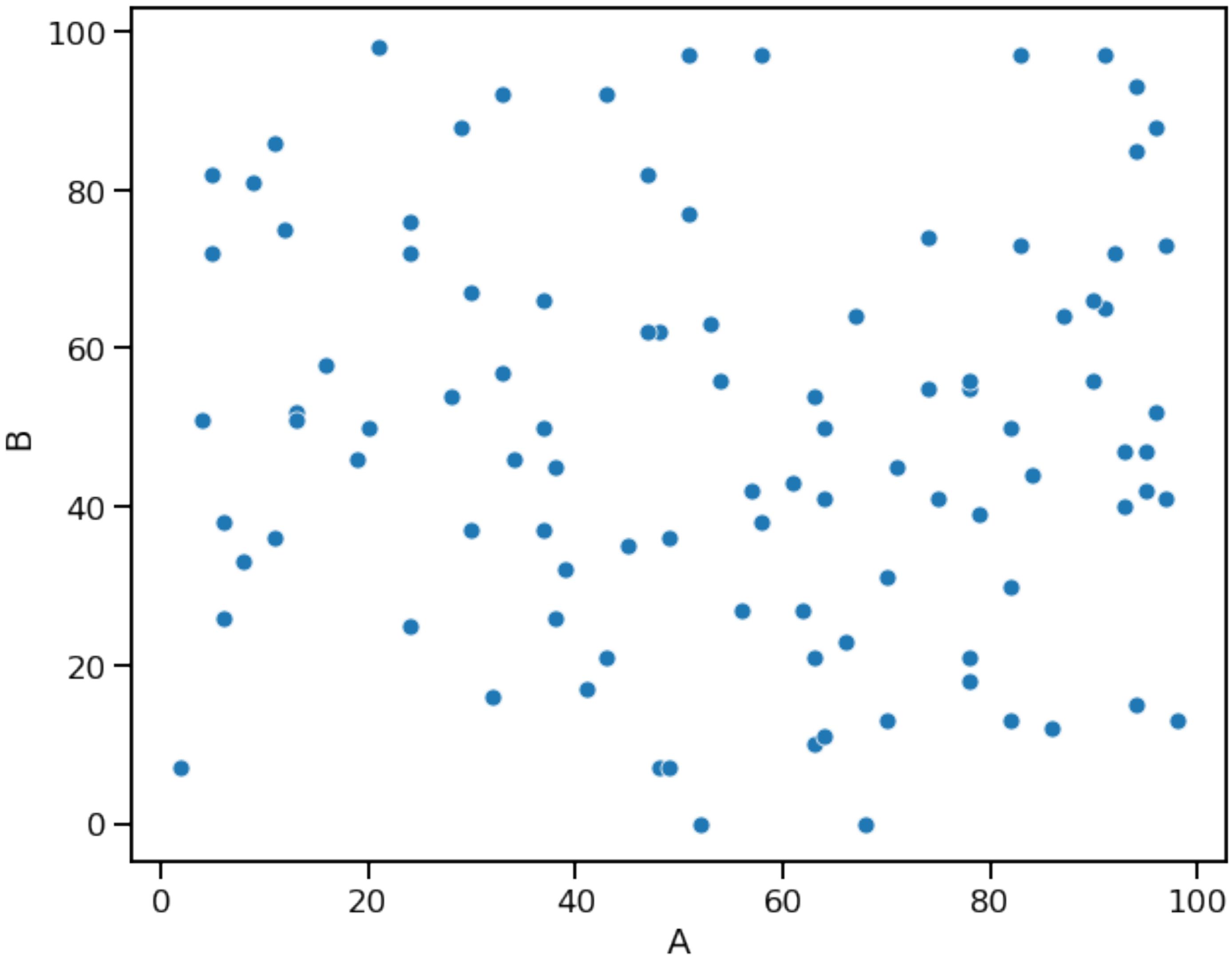


```
df3 = pd.DataFrame(np.random.randn(1000, 2),  
                   columns=[ 'B' , 'C' ]).cumsum()  
df3[ 'A' ] = pd.Series(list(range(len(df3))))  
  
df3.plot( x='A' , y='B' )
```

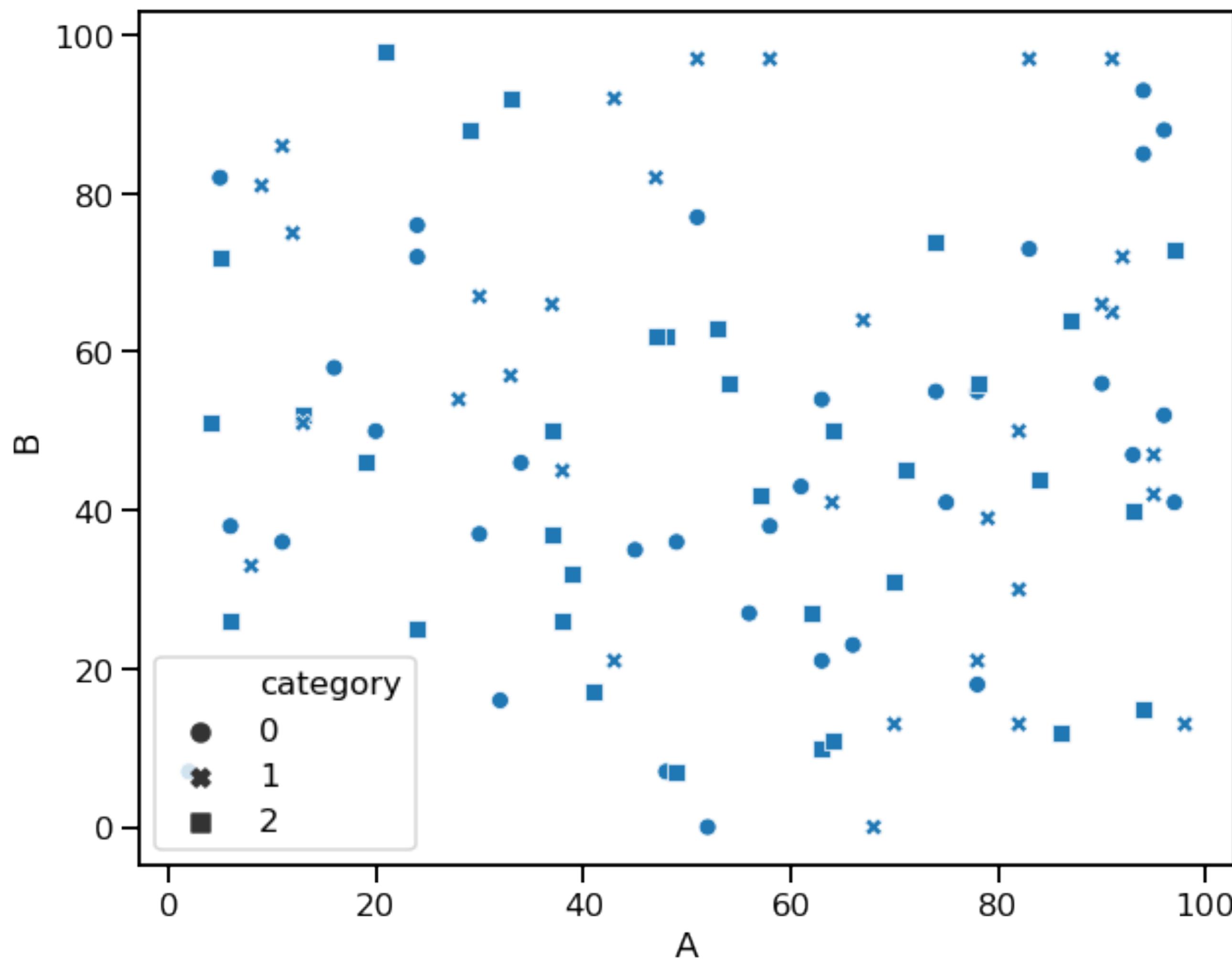


```
df4.plot(kind='hist',  
alpha=0.5,  
bins=50 )
```

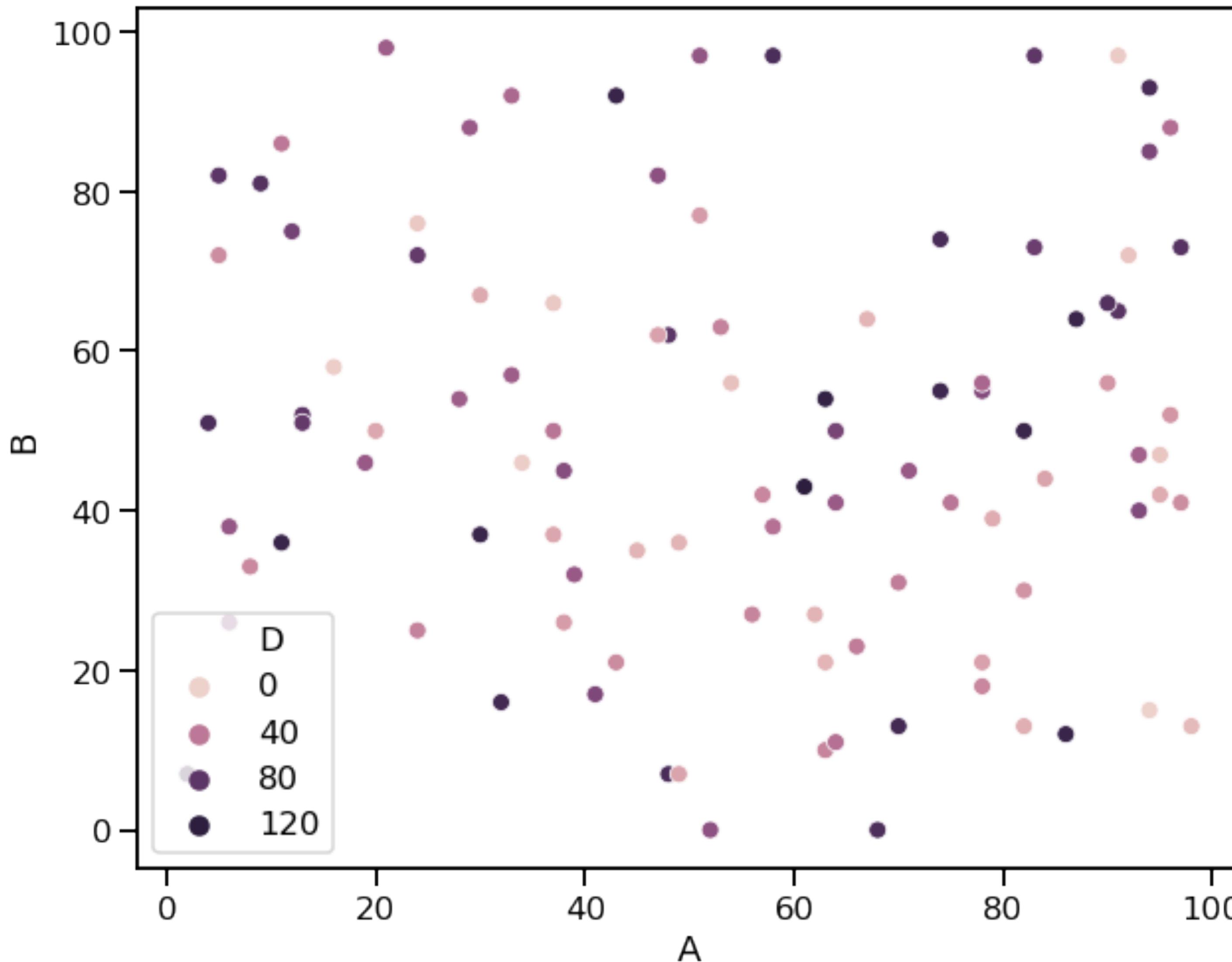
```
import seaborn as sns
```



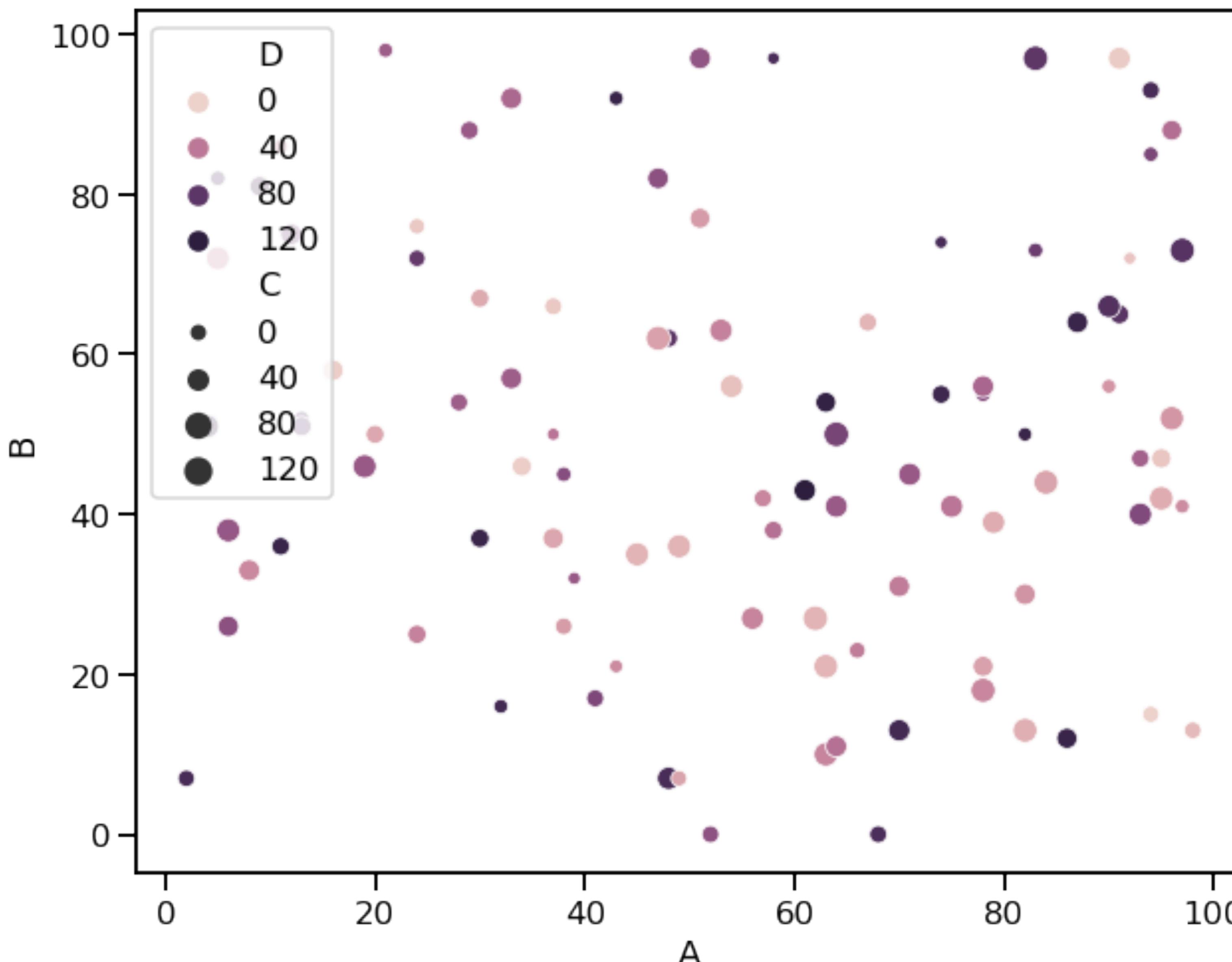
```
df = pd.DataFrame(np.random.randint(0,100,size=(100, 4)),  
                  columns=list('ABCD'))  
df['category'] = df['C'] % 3  
  
sns.scatterplot(x=df['A'], y=df['B'], data=df)
```



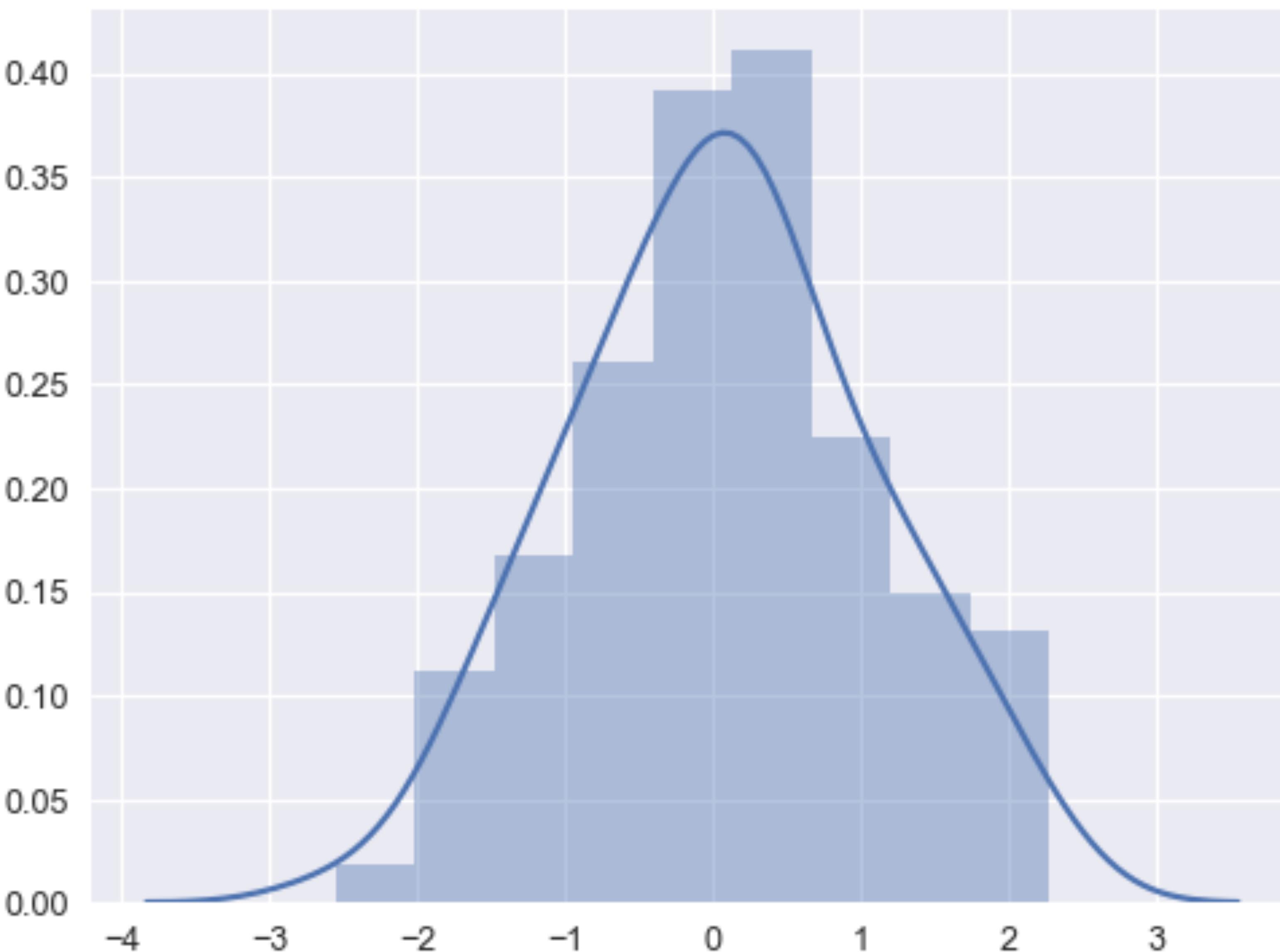
```
sns.scatterplot(x=df[ 'A' ] , y=df[ 'B' ] ,  
style=df[ 'category' ] )
```



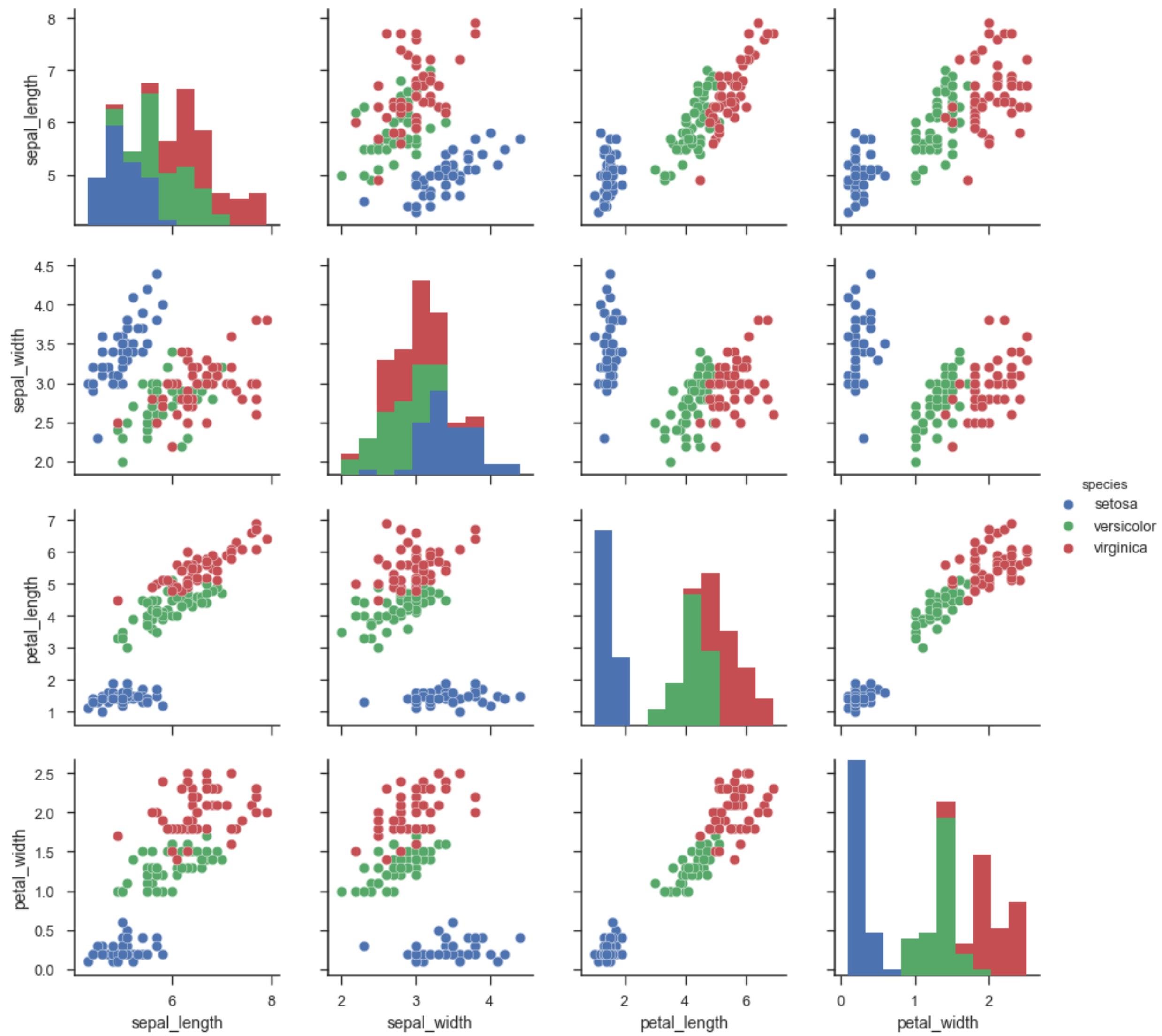
```
sns.scatterplot(x=df[ 'A' ], y=df[ 'B' ], hue=df[ 'D' ])
```



```
sns.scatterplot(x=df[ 'A' ], y=df[ 'B' ], size=df[ 'C' ],  
hue=df[ 'D' ] )
```

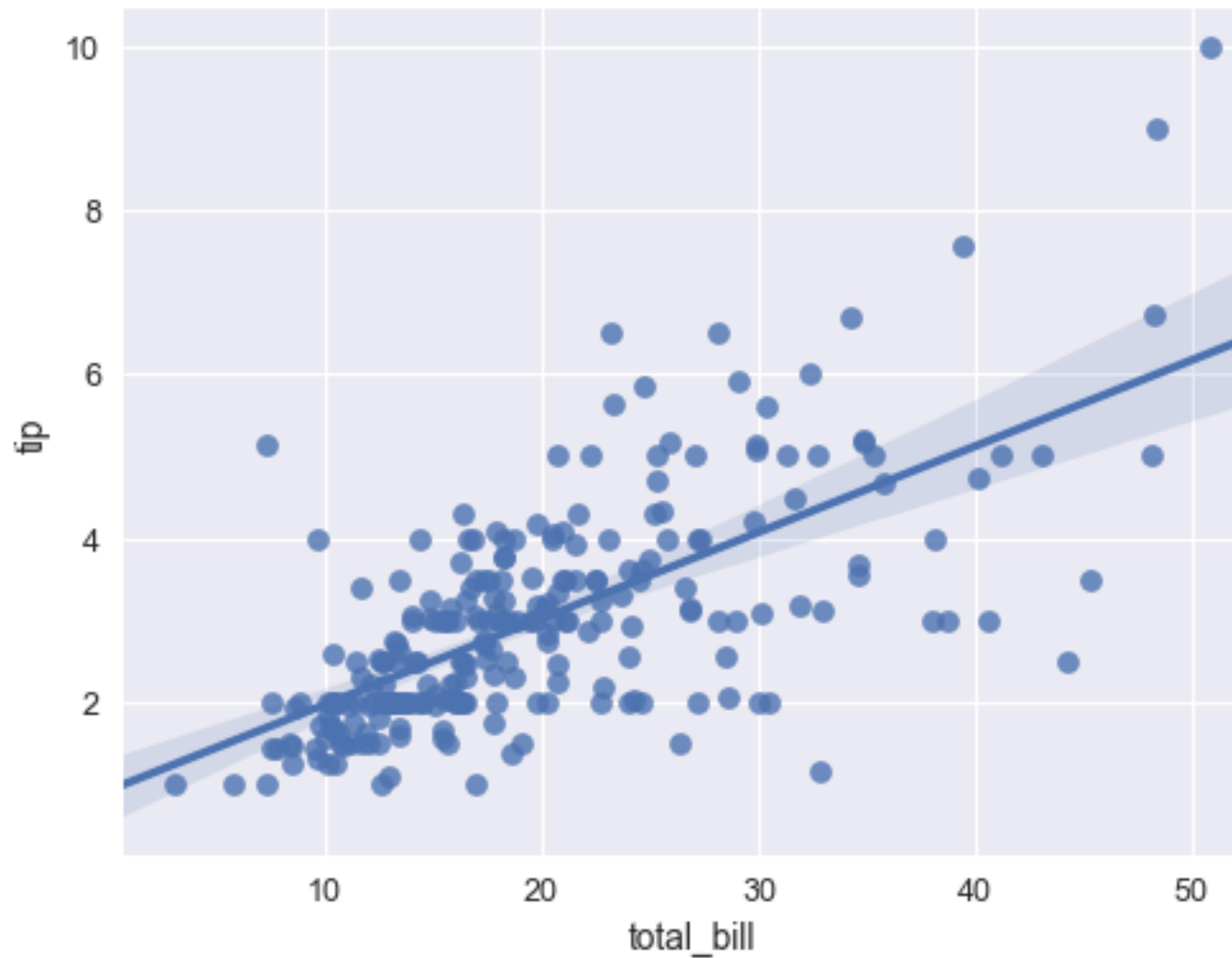


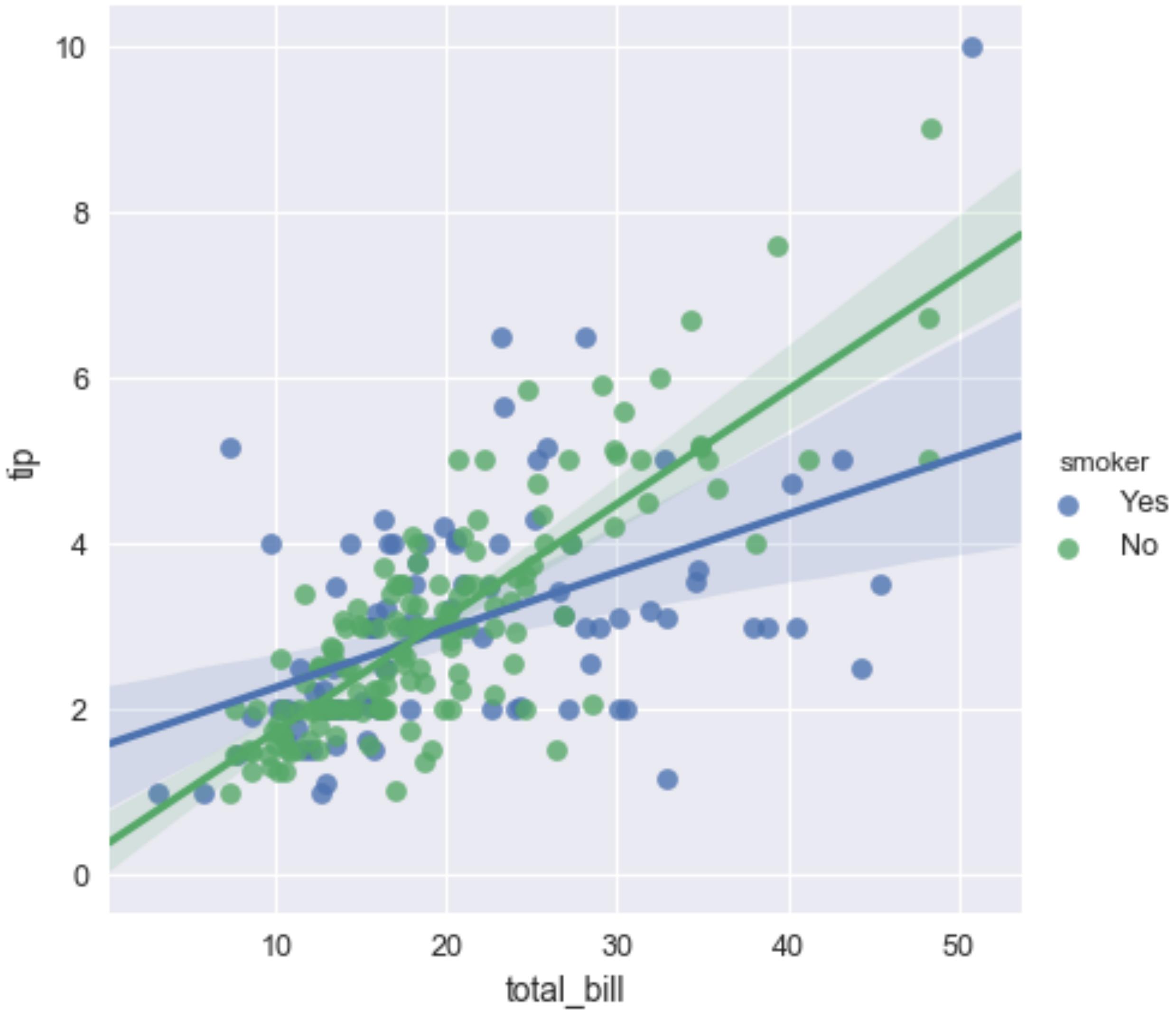
```
ax = sns.distplot(<data>)
```



```
g = sns.pairplot(<data>, hue="<target>" )
```

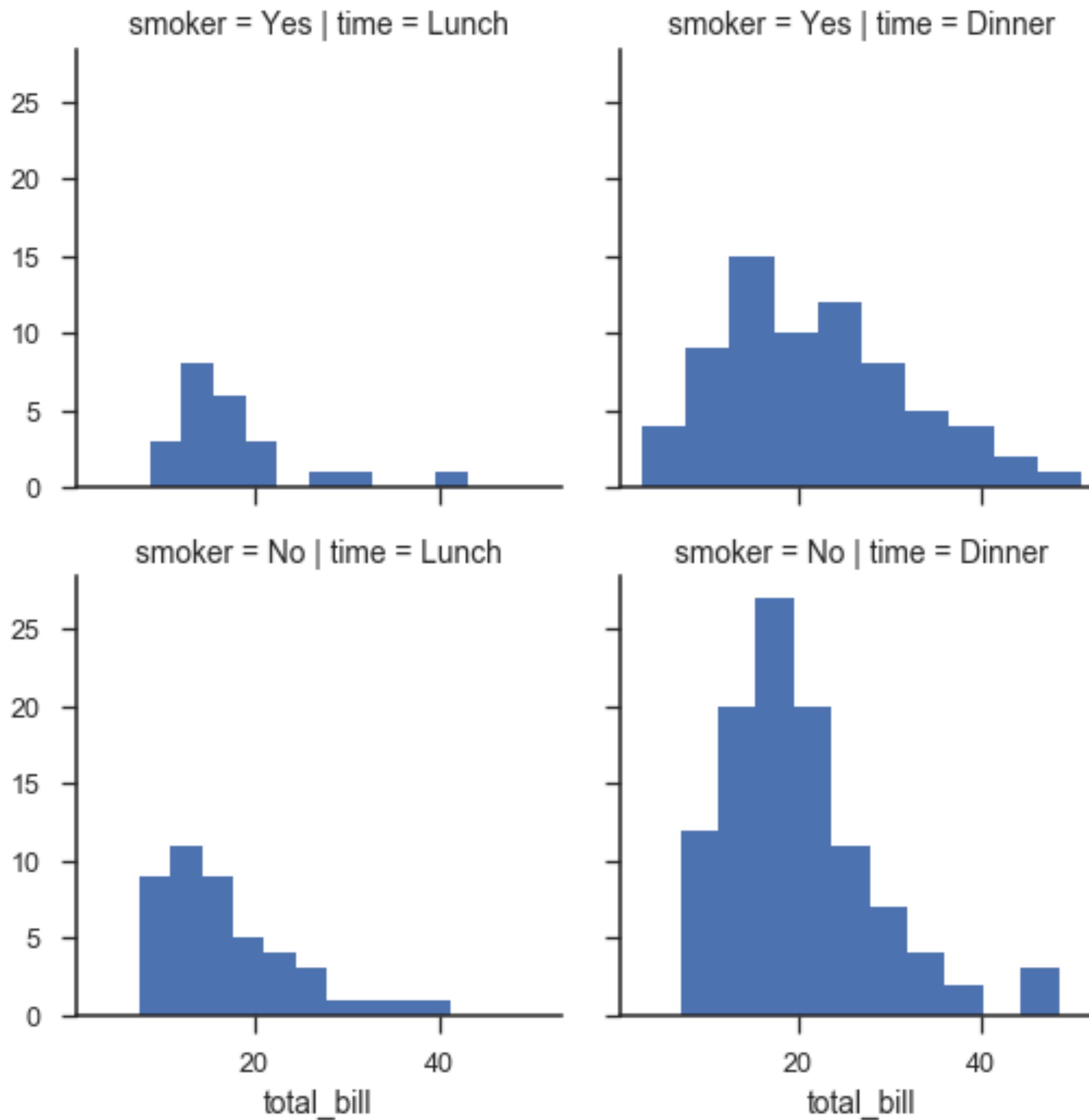
```
ax = sns.regplot(x="total_bill", y="tip",  
data=tips)
```





```
g = sns.lmplot(x="total_bill", y="tip",
hue="smoker", data=tips)
```

```
g = sns.FacetGrid(tips, col="time", row="smoker")
>>> g = g.map(plt.hist, "total_bill")
```



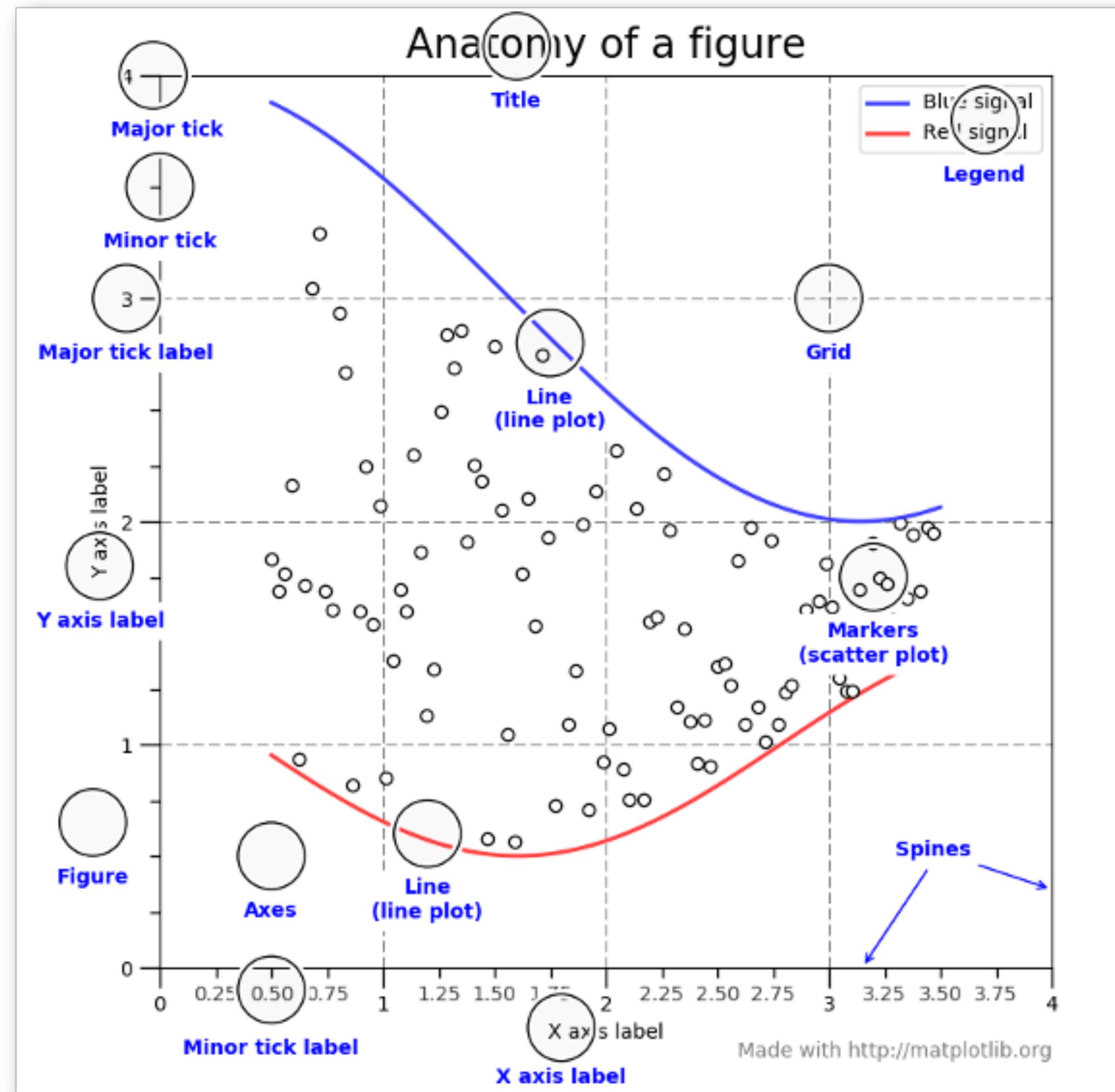
```
scatterPlot.get_figure().savefig( "scatterPlot.png" )
```

```
print(plt.style.available)

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

# **Customizing your Charts**

# Customizing your Charts



# Customizing your Charts

- **Axes:** Axes represent an individual plot or chart.
- **Figure:** This is the final complete image and may contain 1 or more axes.

# Customizing your Charts

- Matplotlib has both an object-oriented and state-based interface. This can be confusing when you look up answers on Stack Overflow.
- Use basic pandas plotting for simple plots
- Seaborn, YellowBrick and others for more complex visualizations
- Use the OO interface in MatPlotLib to customize simple visualizations.



# Customizing your Charts

```
fig, ax = plt.subplots()  
df.plot(..., ax=ax)
```

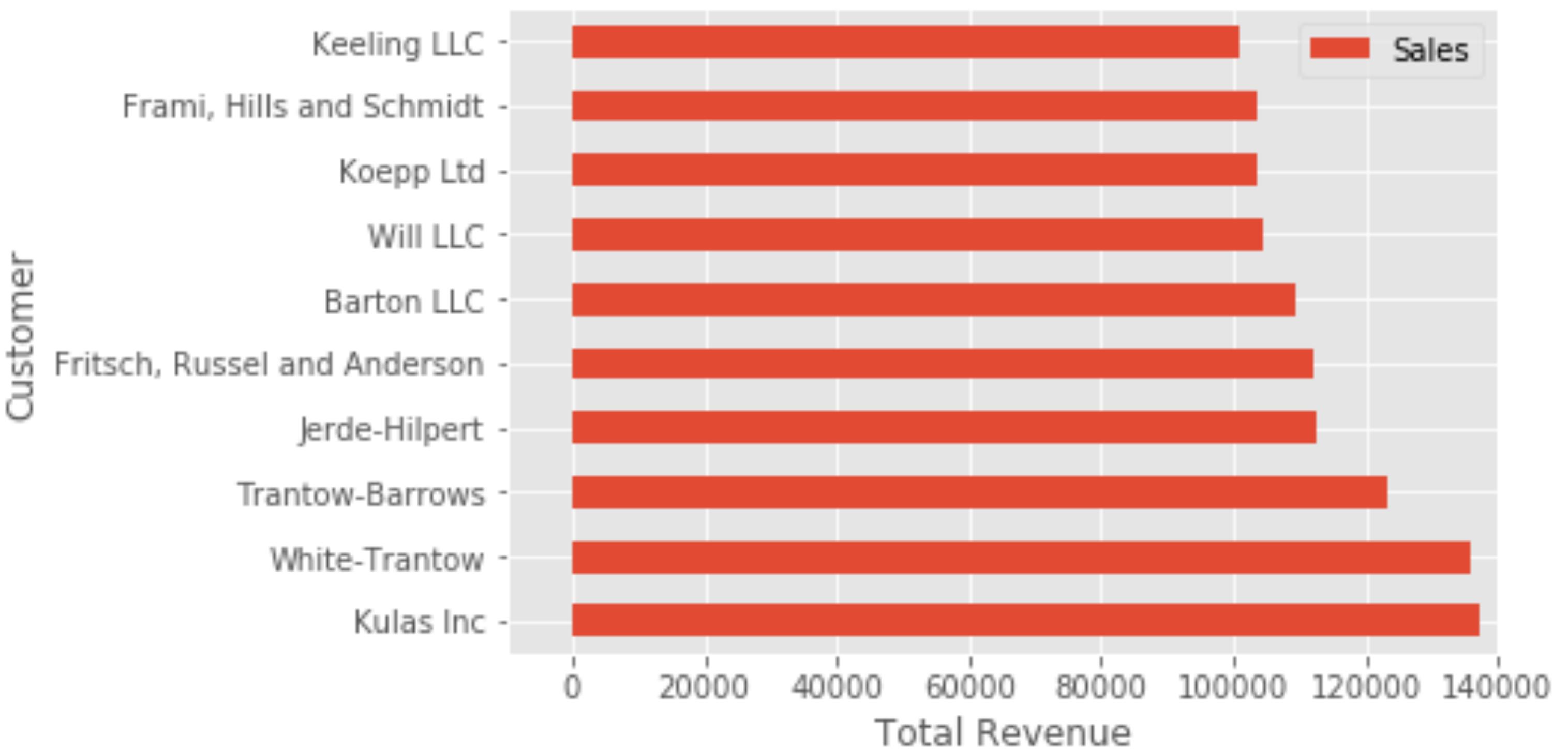
This method gets you access to the **figure** and **axes**.

# Customizing your Charts

- **Axes:** Long list of possible options to configure your chart: [https://matplotlib.org/3.1.0/api/axes\\_api.html#the-axes-class](https://matplotlib.org/3.1.0/api/axes_api.html#the-axes-class)

# Add Axis Labels

```
fig, ax = plt.subplots()  
df.plot(kind='barh', ax=ax)  
ax.set_xlim([-10000, 140000])  
ax.set_xlabel('Total Revenue')  
ax.set_ylabel('Customer');
```

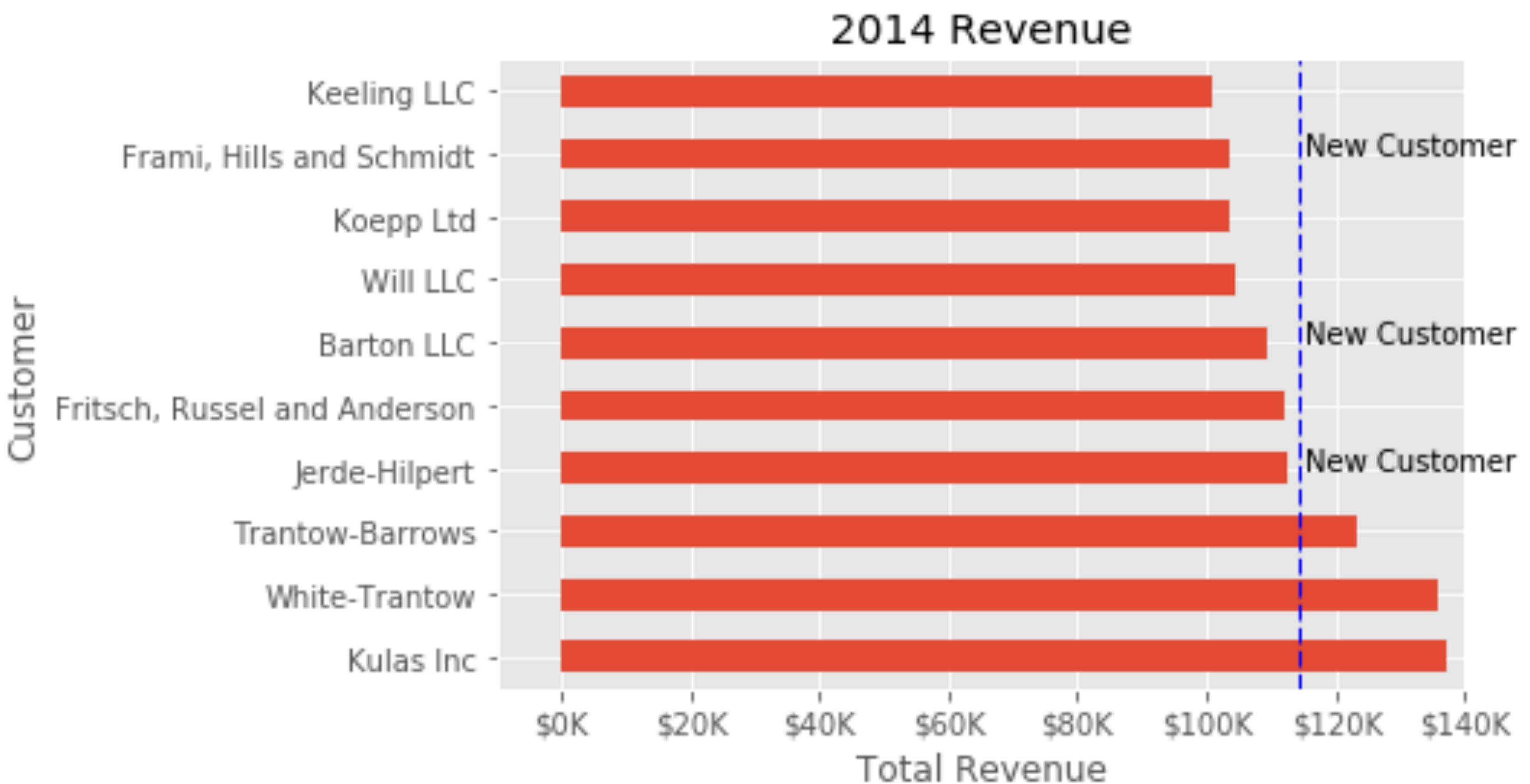


# Add Annotations

```
fig, ax = plt.subplots()
df.plot(kind='barh', ax=ax)
avg = df['Sales'].mean()

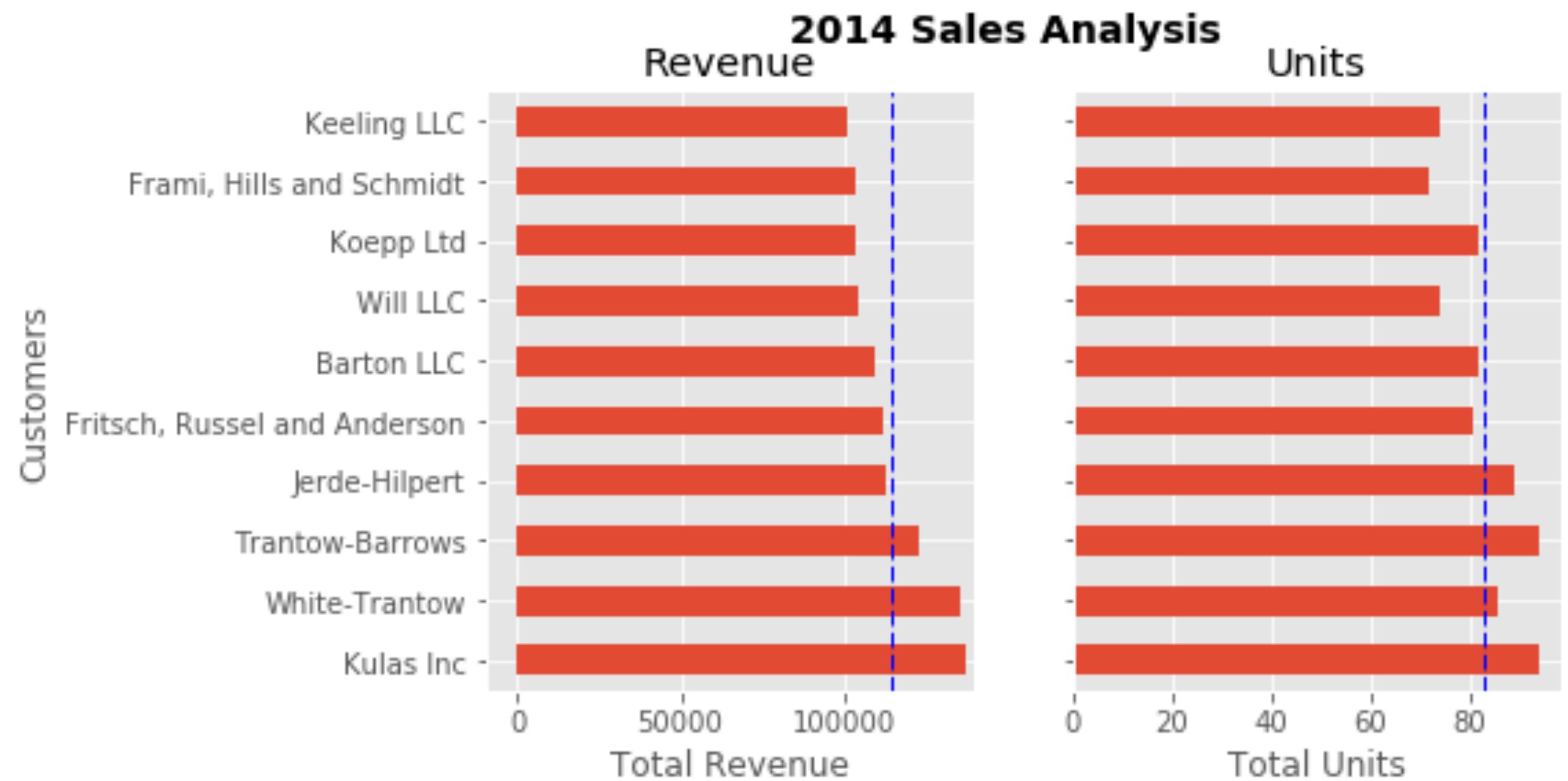
ax.axvline(x=avg,
            color='b',
            label='Average',
            linestyle='--',
            linewidth=1)

for cust in [3, 5, 8]:
    ax.text(115000, cust,
            "New Customer")
```



# Subplots

```
fig, (ax0, ax1) = plt.subplots(nrows=1,
                               ncols=2,
                               sharey=True,
                               figsize=(7, 4))
df.plot(kind='barh', ax=ax0)
ax0.set_xlim([-10000, 140000])
...
df.plot(kind='barh', ax=ax1)
avg = df['Purchases'].mean()
ax1.set(title='Units',
        xlabel='Total Units',
        ylabel=' ')
ax1.axvline(x=avg, color='b',
             label='Average',
             linestyle='--',
             linewidth=1)
```



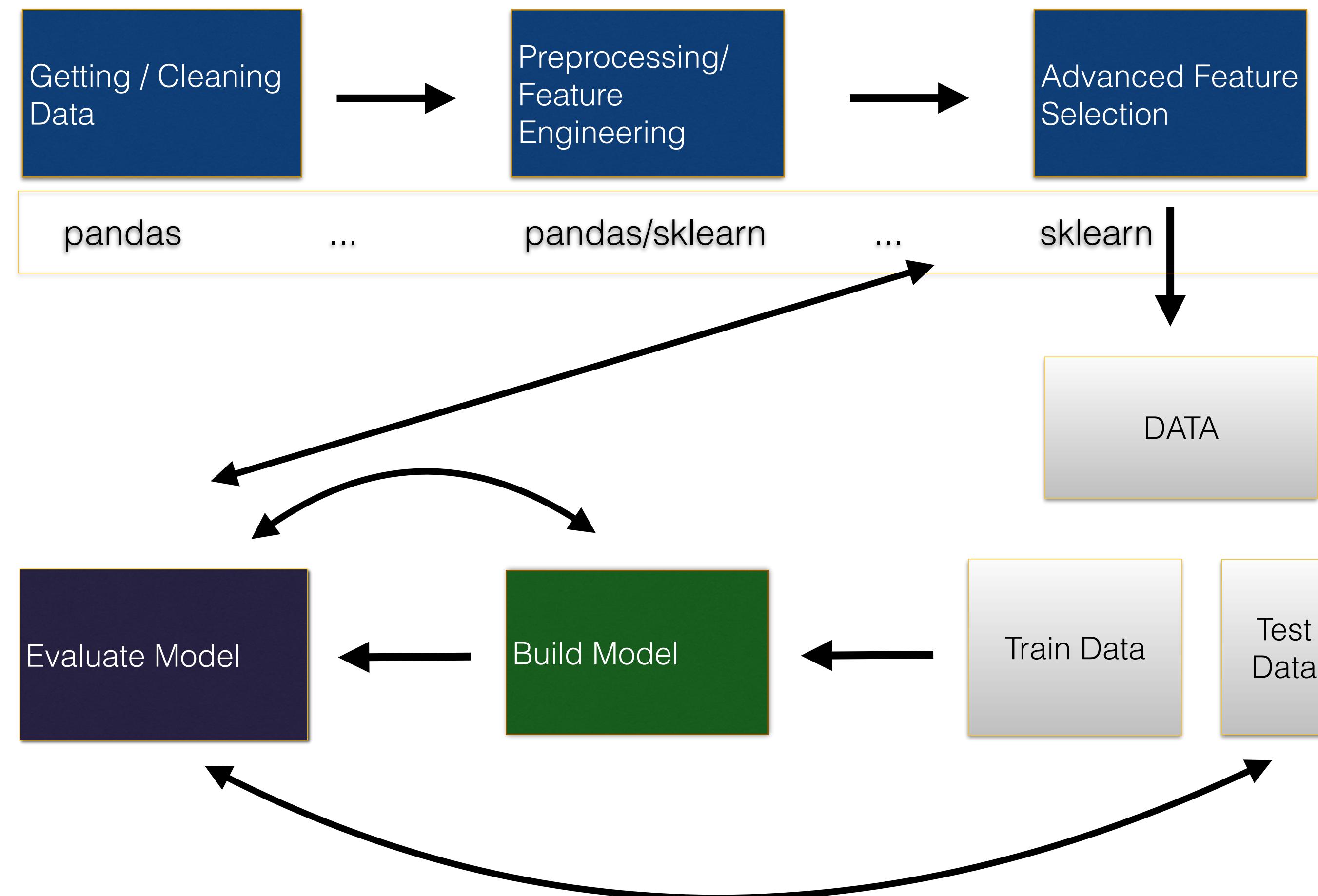
# **Module 4: Machine Learning Part 1**

## **Feature Engineering**

# Agenda

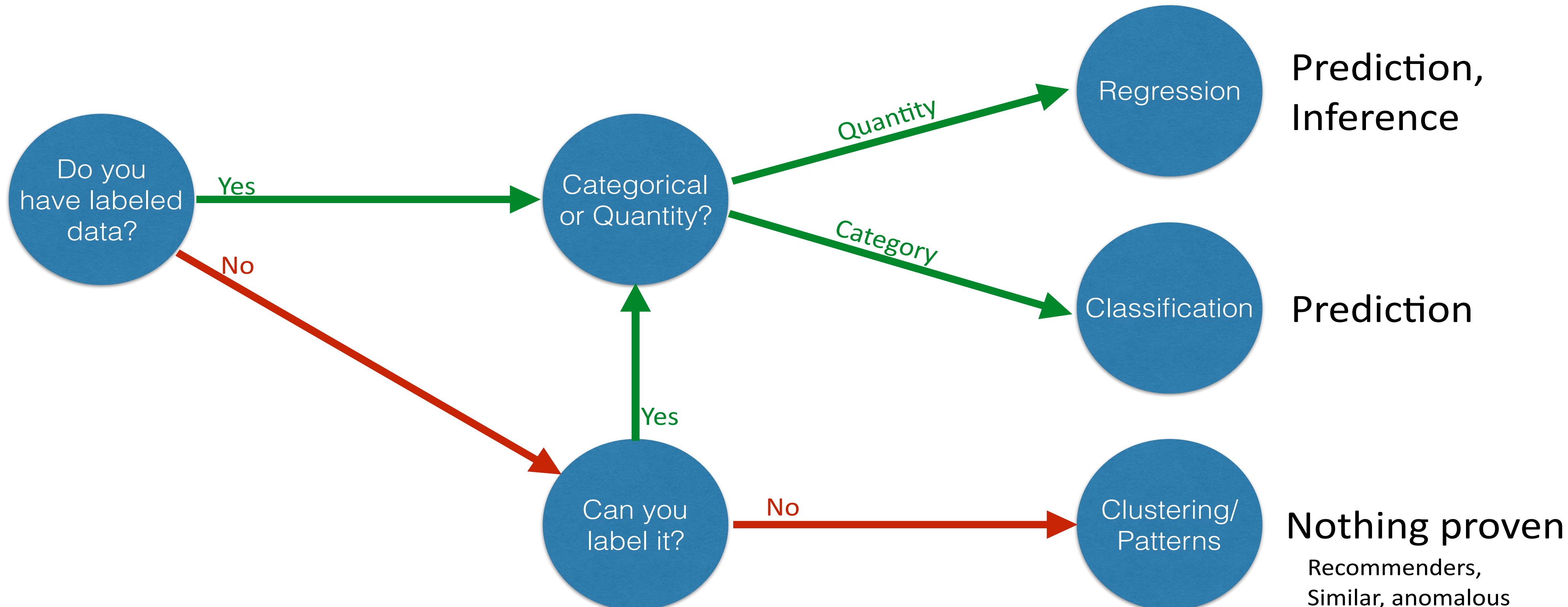
- Feature Selection & Engineering
- Math free overview of classification models
- Evaluating Model Performance
- Improving model performance

# Machine Learning Process



# Machine Learning Terms

- **Features:** The mathematical representation of the original data. The features are the columns in your data set. Since the features will be a matrix, they are often written as  $X$ .
- **Observations:** The rows of your feature set.
- **Target:** The variable that you are trying to predict. Often represented as  $y$ .



# Features

<http://www.google.com>



# Features

http://www.google.com



domain_length	vowel_count	digit_count
6	3	0

# Representation of URL Knowledge

- Come up with a **representation/set of knowledge** that has **enough complexity** to accurately describe the problem for the computer
- Knowledge here does not mean hard-coded knowledge or formal set of rules
- **The computer rather uses the knowledge we provide to extract patterns and acquire own knowledge**
- We should provide knowledge about reality that has **high variance about the problem** it describes (e.g. a feature that is high when it rains and low when it's sunshine)

# URL Definition

https://www.google.com/search?  
q=URL&source=lnms&tbo=isch&sa=X&ved=0ahUKEwjcl6ut-  
IDUAhVEPCYKHdJGDsYQ\_AUIDCgD&biw=1215&ampbih=652

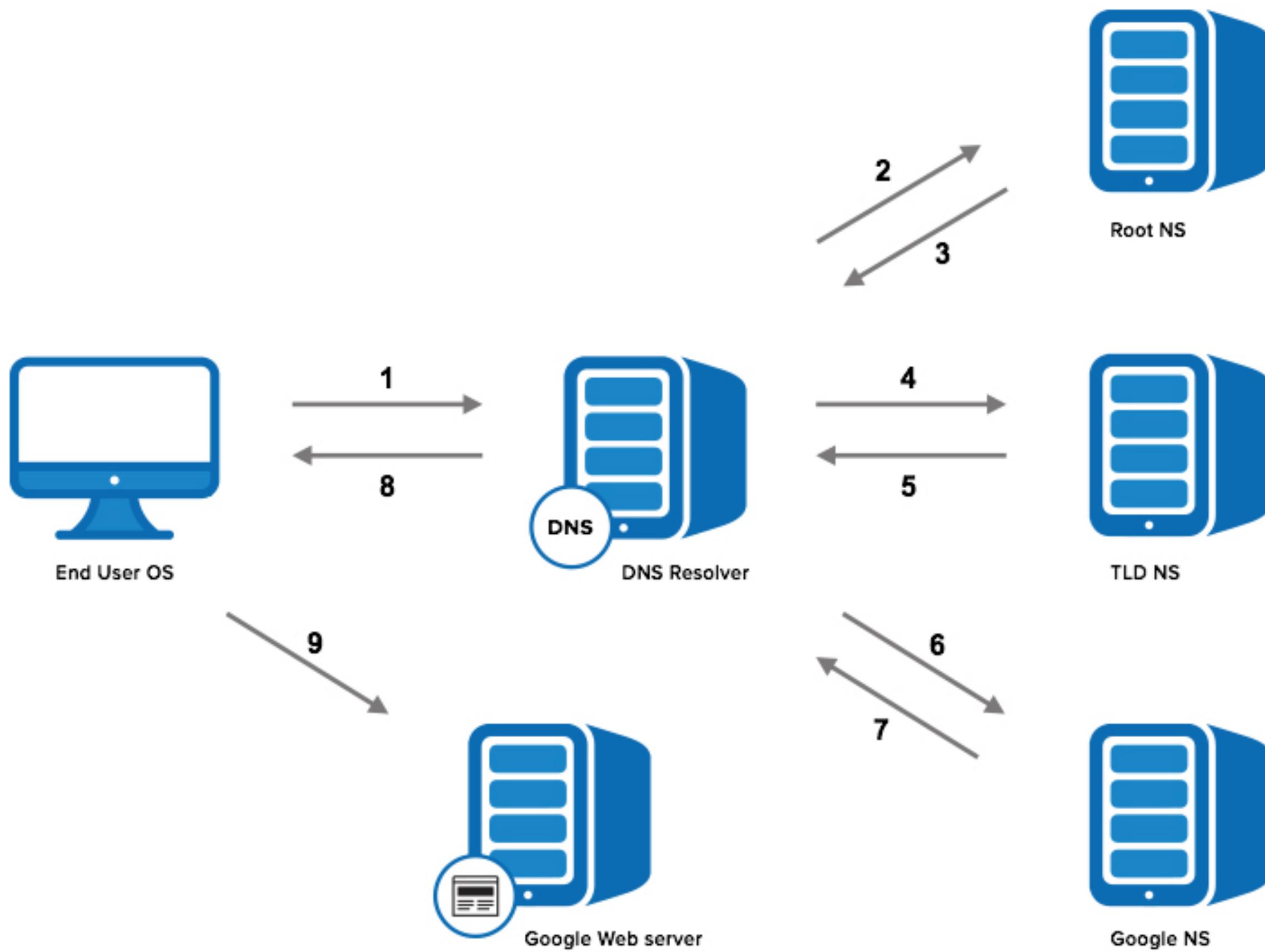
https://	protocol
www	subdomain
google.com	zone apex
google	domain
.com	top-level-domain (tld)
/search?q=URL...	path

# DNS 101

- **Domain Name Service** (DNS) resolves domain names to IP addresses (like a phone book)
- **Domain Registrars**: authority that signs unique domain names (GoDaddy, BlueGtaor)
- **State of Authority** (SOA): Contains for example name of server for zone, administrator of zone, default time-to-live (ttl = time a DNS record is cached), seconds of secondary name server should wait before checking for updates
- **Root Zone** controlled by Internet Assigned Numbers Authority (IANA)
- **Name Servers** (NS Records): used by tld servers to direct traffic to DNS server (which contains authoritative DNS records)
- **A records** (part of DNS record): “A” stands for IP Address
- **CNAME** (part of DNS record): resolves one domain name to another
- **Autonomous System** (AS) and Border gateway Protocol (BGP) info

Python libraries: `python-whois`, `dnspython`, `tldextract`, `ipaddress`

# DNS Flow



# What makes them different?

## URL BlackList

amazon-sicherheit.kunden-ueberpruefung.xyz  
eclipsehotels.com/language/en-GB/eng.exe  
bohicacapital.com/page  
summerweb.net  
ad.getfond.info  
vdula.czystykod.pl/rxdjna2.html  
svision-online.de/mgfi/administrator/components/com\_babackup/classes/fx29id1.txt

## URL WhiteList

gurufocus.com/stock/PNC  
dvdtalk.ru/review  
333cn.com/zx/zhxw.html  
made-in-china.com/special/led-lighting  
google.com/u/0/112261544981697332354/posts  
youtube.com/watch?v=Qp8MQ4shN6U  
unesco.org/themes/education-sustainable-developm  
thisisfirst.com/page/5

# Malicious URL Detection Features (Literature)

1. **BlackList Features:** BlackLists suffer from a high false negative rate, but can still be useful as machine learning feature.
2. **Lexical Features:** Capture the property that malicious URLs tend to "look different" from benign URLs. **Contextual information** such as the length of the URL, number of digits, lengths of different parts, entropy of domain name.
3. **Host-based Features:** Properties of web site host. **"Where"** the site is hosted, **"who" owns it** and **"how" it is managed**. API queries are needed (WHOIS, DNS records). Examples: Date of registration, the geolocations, autonomous system (AS) number, connection speed or time-to-live (TTL).
4. **Content-based Features:** Less commonly used feature as it requires **execution of web-page**. Can be not only be not safe, but also increases the computational cost. Examples: HTML or JavaScript based.

# Preparation In Class Exercise

## ML Feature Engineering

### Lexical Features

1. Length of URL
2. Length of domain
3. Count of digits
4. Entropy of domain
5. Position (or index) of the first digit
6. **Bag-of-words** for tld, domain and path parts of the URL

### Host-based Features

1. Time delta between today's date and creation date
2. Check if it is an IP address

# Data Set (Features and Target)

	url	isMalicious	domain	created
56675	jeita.biz/w/google/drive/document.html?ssl=yes	1	jeita.biz	2012-04-11 17:08:19
73229	sosnovskoe.info/layouts/plugins/mailbox	1	sosnovskoe.info	2011-09-19 09:53:07
60112	teothemes.com/html/mp3pl/blue-preview.jpg	1	teothemes.com	2011-09-08 21:43:00
66946	kfj.cc:162/17852q	1	kfj.cc	2013-08-18 05:52:47
81906	verapdpf.info/db/6d1b281b5c4bbcfe3b99228680c232fa	1	verapdpf.info	2016-08-18 07:09:03

Features or X

	isMalicious	isIP	Length	LengthDomain	DigitsCount	EntropyDomain	FirstDigitIndex	com	org	net	...	w	waset
73320	1	0	27	21	0	3.558519	0	0	1	0	...	0	0
30785	0	0	77	11	14	3.095795	22	1	0	0	...	0	0
60789	1	0	141	11	5	3.459432	103	0	1	0	...	0	0
19495	0	0	59	13	20	3.546594	31	1	0	0	...	0	0
45022	1	0	23	11	7	3.277613	13	0	0	0	...	0	0

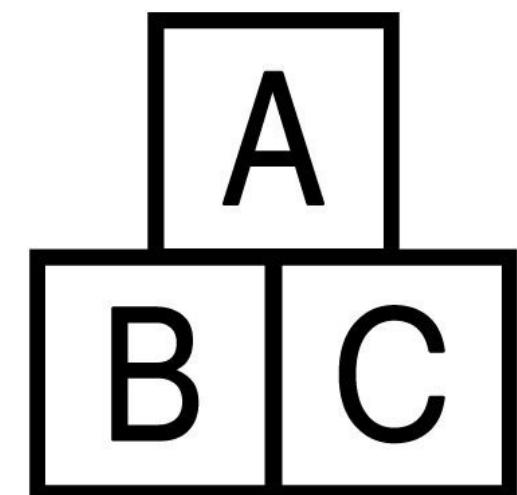
Target or y

# Preprocessing - an Art Work!

- Imputing missing values
- Scaling/Normalization
- One-Hot Encoding (Encoding categorical features)
- Embedding (e.g. word2vec)
- Binarizing (e.g. needed for Deep Learning multi-class target vector encoding)
- Encoding strings as int
- Dimensionality Reduction (e.g. PCA)
- Augmentation (e.g. tild/zoom images)
- Feature selection based on classifier
- Variance threshold

Primary Python libraries: **pandas**, **sklearn**, **scipy**

Data Types



01

# Imputing Missing Values

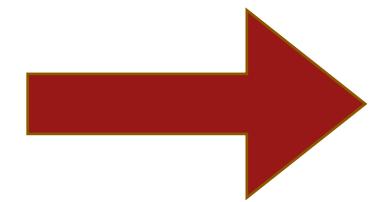
```
# using the most_frequent value  
df['src_bytes'] = df['src_bytes'].fillna  
(df['src_bytes'].value_counts().index[0])  
  
# using the mean value  
df['dst_bytes'] = df['dst_bytes'].fillna(df['dst_bytes'].mean())
```

# One Hot Encoding

Color
Red
Red
Blue
Green
Yellow
Red

# One Hot Encoding

4 Categories



4 Columns with 1 when Category is True and delete original column!

Color
Red
Red
Blue
Green
Yellow
Red

	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

# One Hot Encoding

```
colors = [ 'Red' , 'Red' , 'Blue' , 'Green' , 'Yellow' , 'Red' ]  
series_data = pd.Series( colors )  
pd.get_dummies( series_data )  
  
# df scenario  
df=pd.get_dummies(df, prefix=None, prefix_sep='_',  
dummy_na=False, columns=[ 'protocol_type' , 'flag' ],  
sparse=False)
```

# Encoding Strings as Integers

```
PROBE = [ 'portsweep.', 'satan.', 'nmap.', 'ipsweep.' ]  
df = df.replace(to_replace = PROBE, value=1)
```

# Feature Scaling

When you're creating a scaling object, you should first "fit" it to the **training data**, then **transform** both the **training and testing data** using the "fit" scaler.

If you try to fit the training and testing data separately, you will get inaccurate results.

# Standard Scaling

$$zscore(x_i) = \frac{x_i - \mu}{\sigma}$$

```
scaled_feature = (feature - column_mean) / standard_deviation
```

# Standard Scaling

$$zscore(x_i) = \frac{x_i - \mu}{\sigma}$$

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()  
scaler.fit(features)  
features_scaled = scaler.transform(features)
```

or

```
features_scaled = scaler.fit_transform(features)
```

# Standard Scaling

original values:

```
[ [ 0.9   0.1   40. ]
[ 0.3   0.2   50. ]
[ 0.6   0.8   60. ] ]
```

Mean of each column:

```
[ 0.6   0.3667  50.]
```

SD of each column:

```
[ 0.2449  0.3091  8.165 ]
```

scaled values:

```
[ [ 1.2247 -0.8627 -1.2247 ]
[ -1.2247 -0.5392  0.      ]
[ 0.        1.4018   1.2247 ] ]
```

**Means of scaled data, per column:**

```
[ 0. -0.  0.]
```

**SD's of scaled data, per column:**

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

Notice that the mean of standard scaled data is zero and the StdDev is 1.

# Min/Max Scaling

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
normed_feature = (feature - col_min) / (col_max - col_min)
```

# Min/Max Scaling

```
from sklearn.preprocessing import MinMaxScaler  
minmax = MinMaxScaler()
```

```
minmax.fit(features)  
features_scaled_minmax = minmax.transform(features)
```

or

```
features_scaled_minmax = minmax.fit_transform(features)
```

# Min/Max Scaling

original values:

```
[ [ 0.9  0.1  40. ]
[ 0.3  0.2  50. ]
[ 0.6  0.8  60. ] ]
```

scaled values:

```
[ [ 1.      0.      0.      ]
[ 0.      0.1429  0.5     ]
[ 0.5     1.      1.      ] ]
```

Mean of each column:

```
[ 0.6  0.3667  50.]
```

Means of scaled data, per column:

```
[ 0.5  0.381   0.5   ]
```

SD of each column:

```
[ 0.2449 0.3091 8.165 ]
```

SD's of scaled data, per column:

```
[ 0.4082 0.4416 0.4082 ]
```

# Dealing with Imbalanced Classes

# Dealing with Imbalanced Classes

- A lot of real-world security data will have very imbalanced targets.
- Fortunately, there is a library called imbalanced-learn which can assist.
- Imbalanced-learn provides a series of options to resample the data so that you have more balanced classes
- Docs available here: <http://imbalanced-learn.org/>

# Dealing with Imbalanced Classes

```
from imblearn.over_sampling import  
RandomOverSampler  
ros = RandomOverSampler(random_state=0)  
X_resampled, y_resampled = ros.fit_resample(X, y)
```

# Selecting Features

# **Should we use all of them?**

**How do we know which features  
to use and which to discard?**

Selects  $k$  features according to the highest score

```
best_features = SelectKBest(score_func=chi2, k=3).fit_transform(features, target)
```

Selects all features above a given threshold in the scoring function

```
best_features =  
SelectPercentile(score_func=chi2, percentile=3).fit_transform(features, target)
```

Available Scoring Functions:

- **For regression:** f\_regression, mutual\_info\_regression
- **For classification:** chi2, f\_classif, mutual\_info\_classif

References:

[http://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.SelectKBest.html](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html)

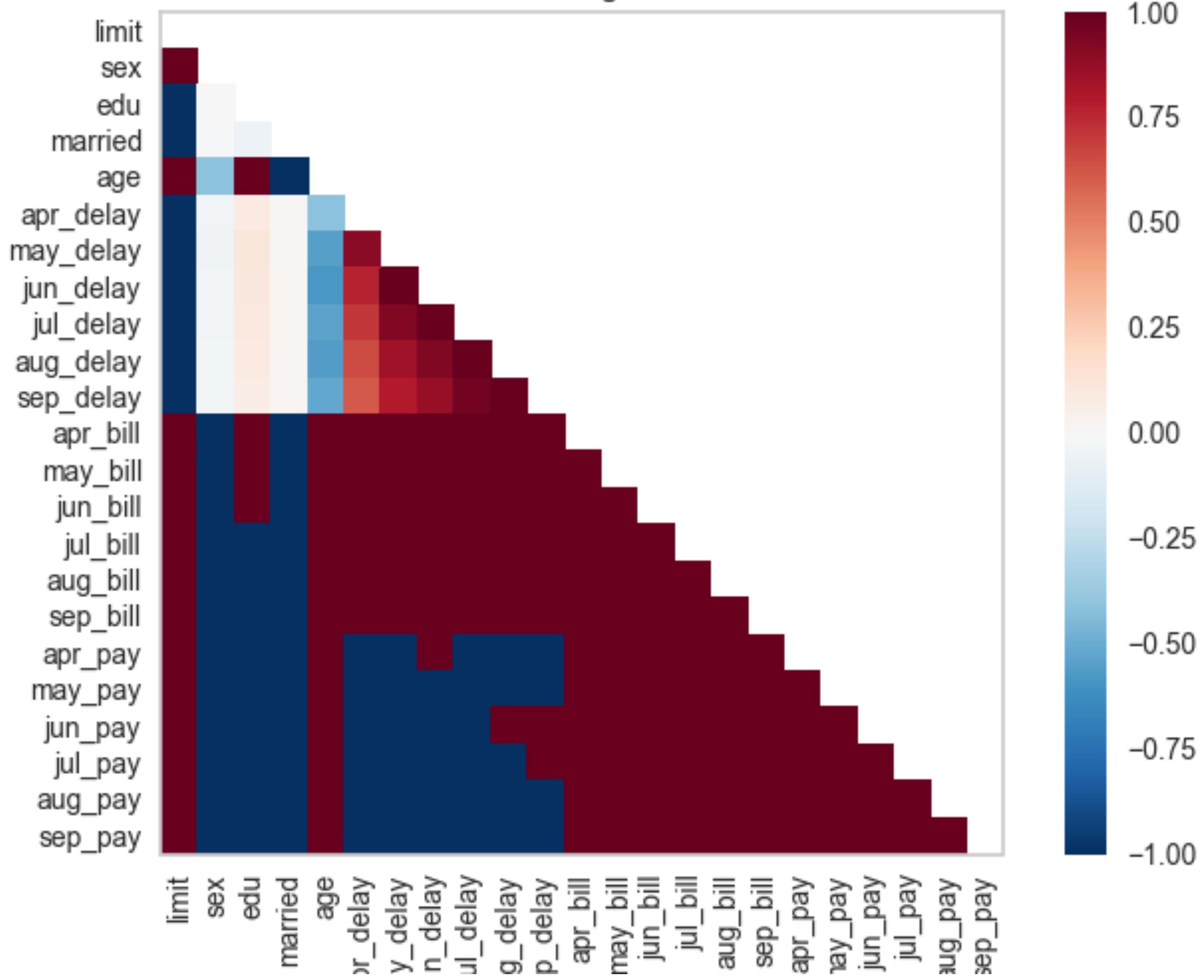
[http://scikit-learn.org/stable/modules/feature\\_selection.html#univariate-feature-selection](http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection)

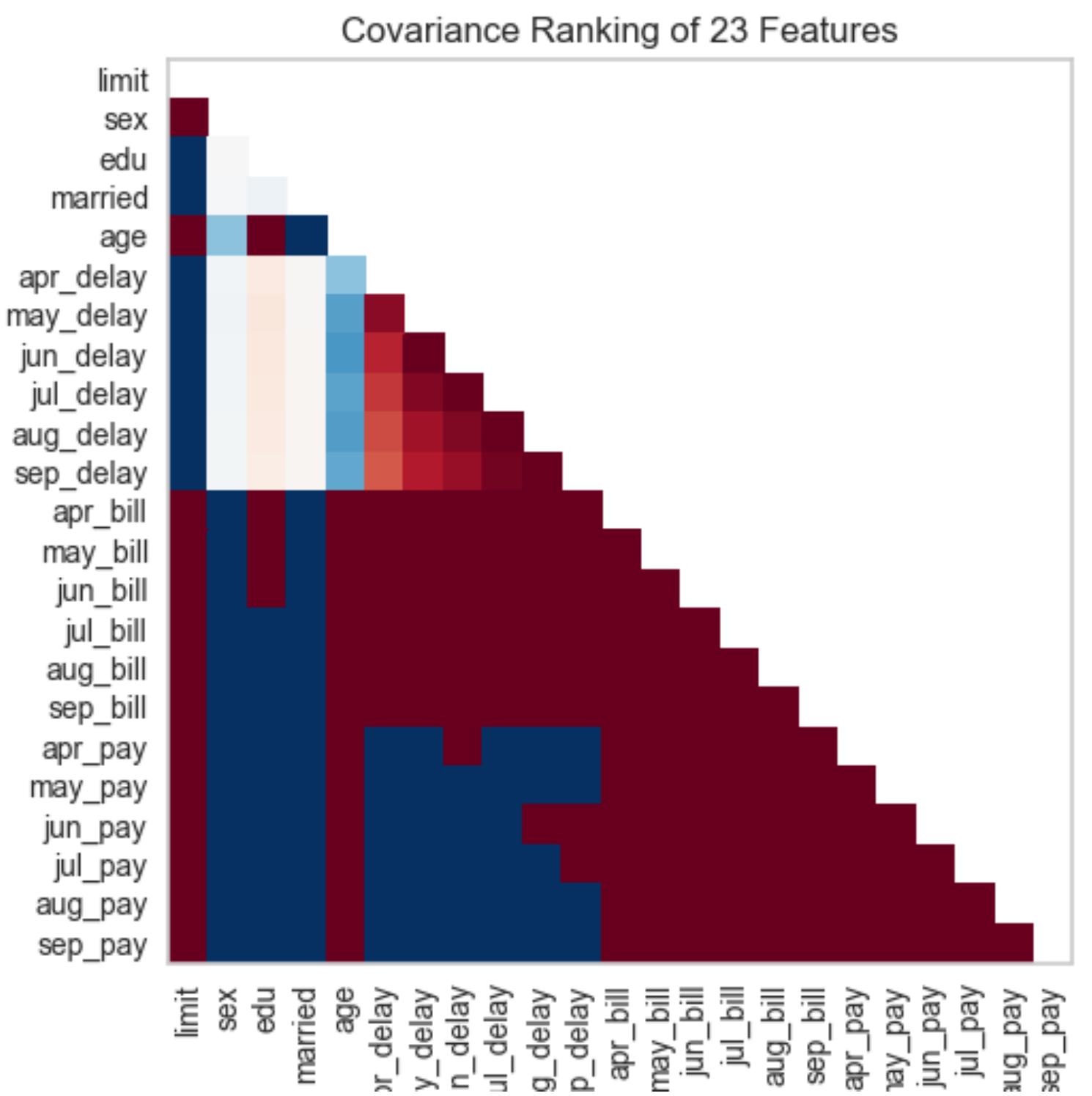
**How do we know which features  
to use and which to discard?**

Visualize them!!

# Introducing Yellowbrick and scikit-plot

Covariance Ranking of 23 Features





```

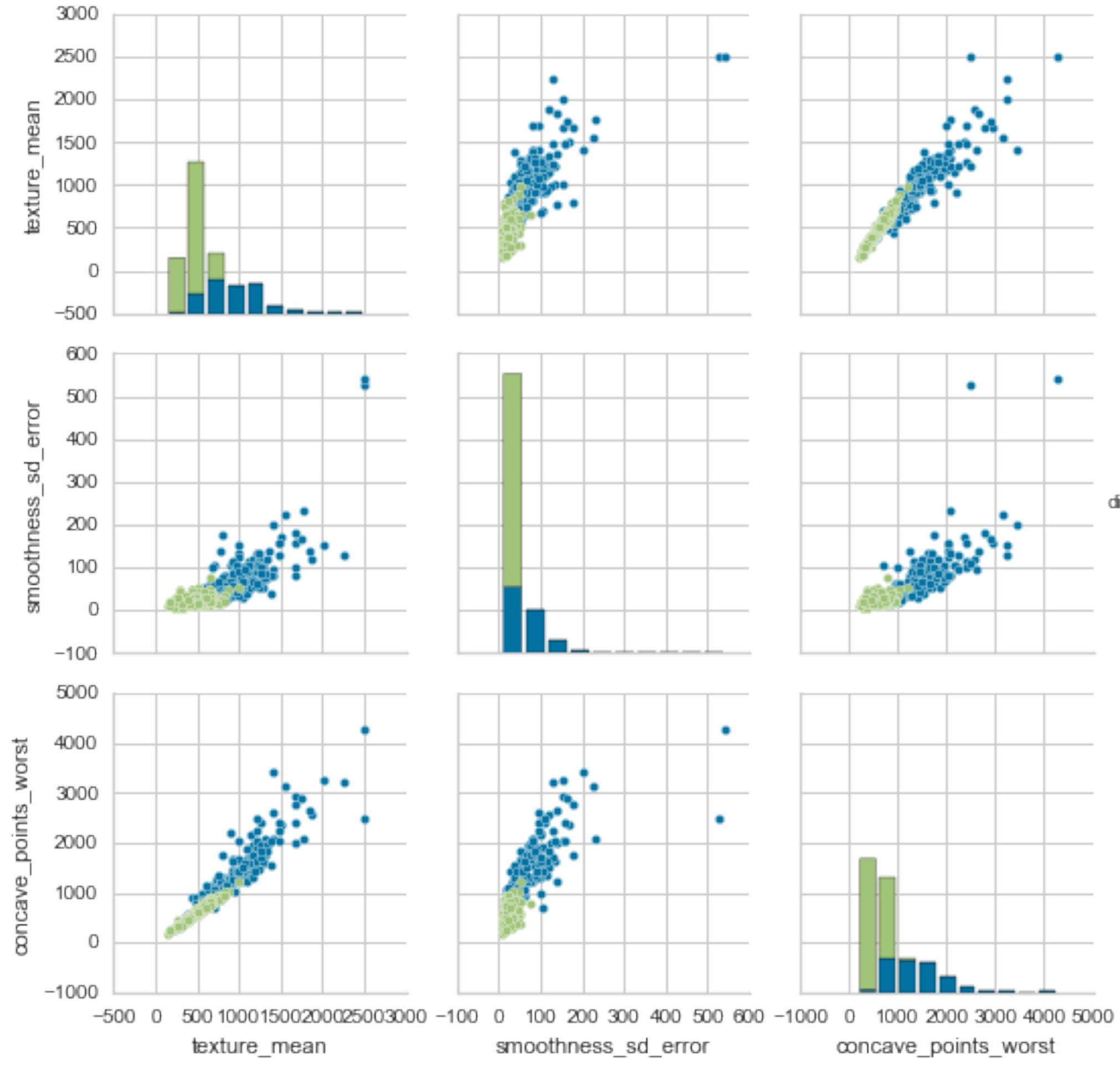
import pandas as pd
import seaborn as sns
from yellowbrick.features.rankd import Rank2D

# Extract the numpy arrays from the data frame
features = df[features]
target = df['diagnosis']

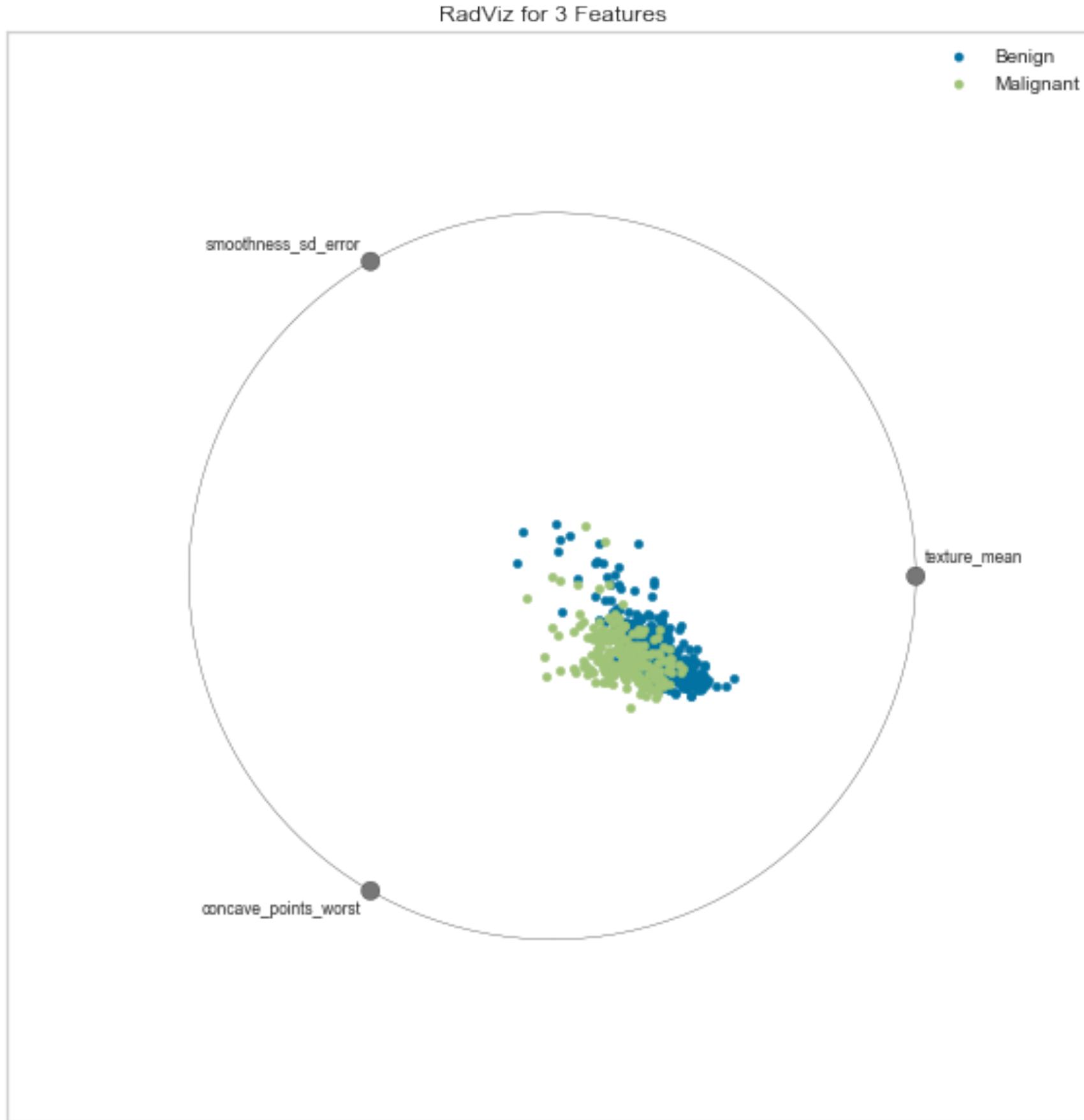
# Instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=features, algorithm='covariance')

visualizer.fit(features, target)
visualizer.transform(features)
visualizer.poof()

```

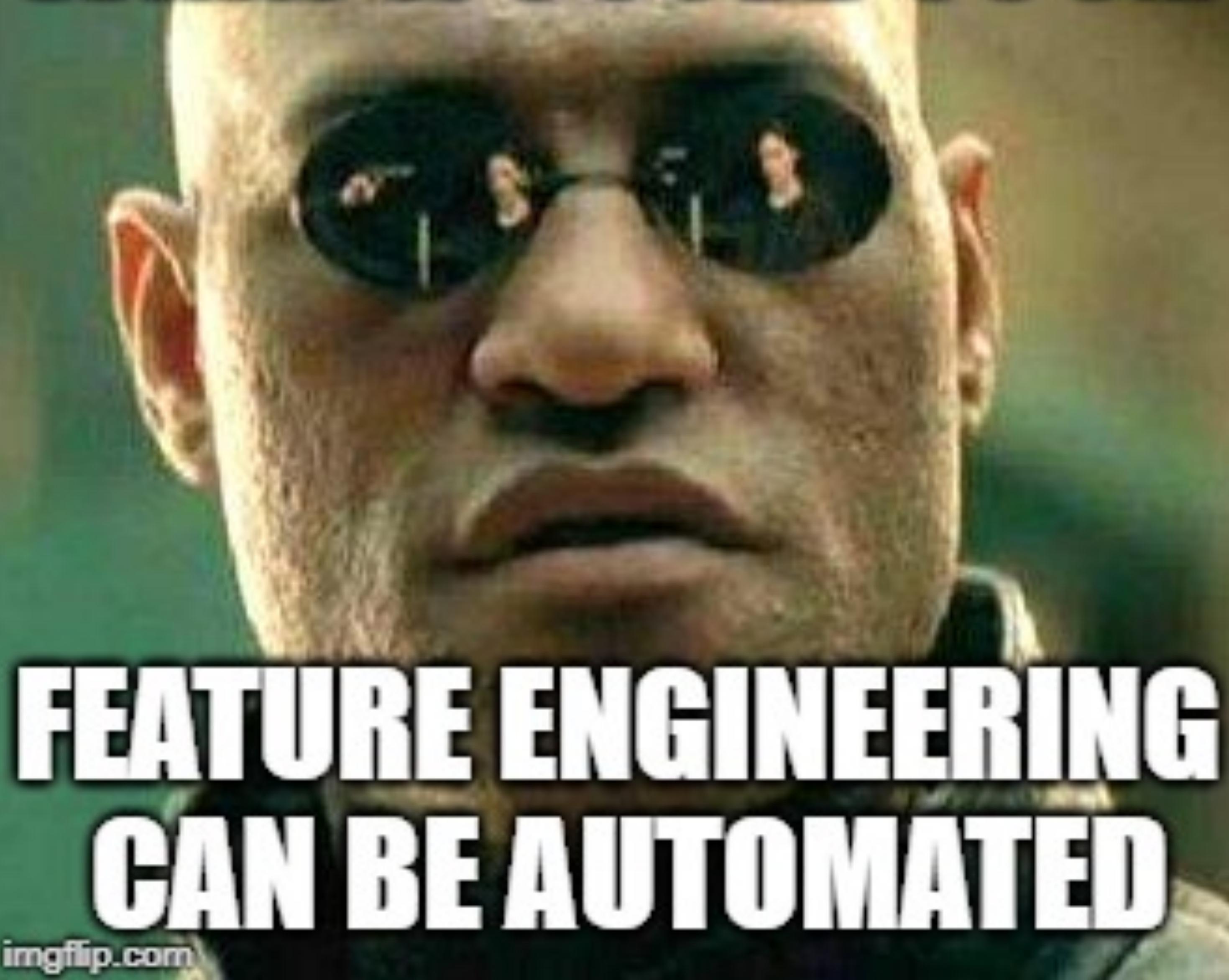


```
import seaborn as sns  
sns.pairplot(<features>, hue='<target>' )
```

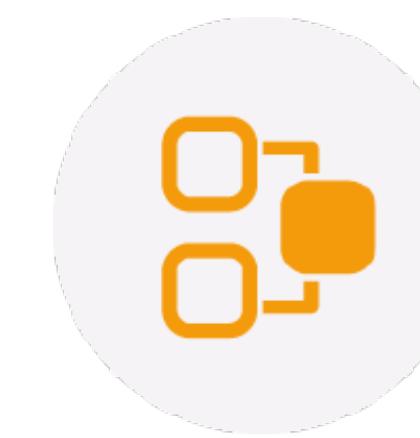


```
from yellowbrick.features.radviz import RadViz
...
visualizer = RadViz(classes=<target classes>, features = <features>)
visualizer.fit(features, target)
visualizer.transform(features)
visualizer.poof()
```

# WHAT IF I TOLD YOU...



## FEATURE ENGINEERING CAN BE AUTOMATED



# Feature Tools

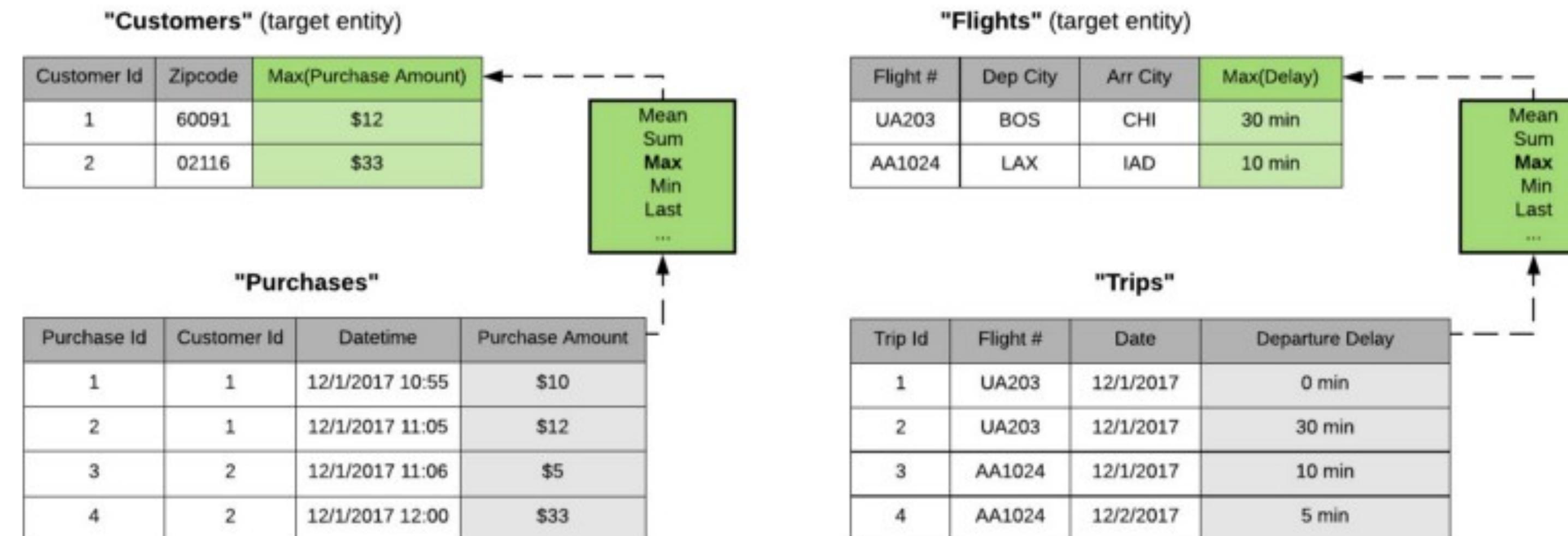
- Open Source
- Automated Feature Engineering



<https://www.featuretools.com/>

# Feature Tools

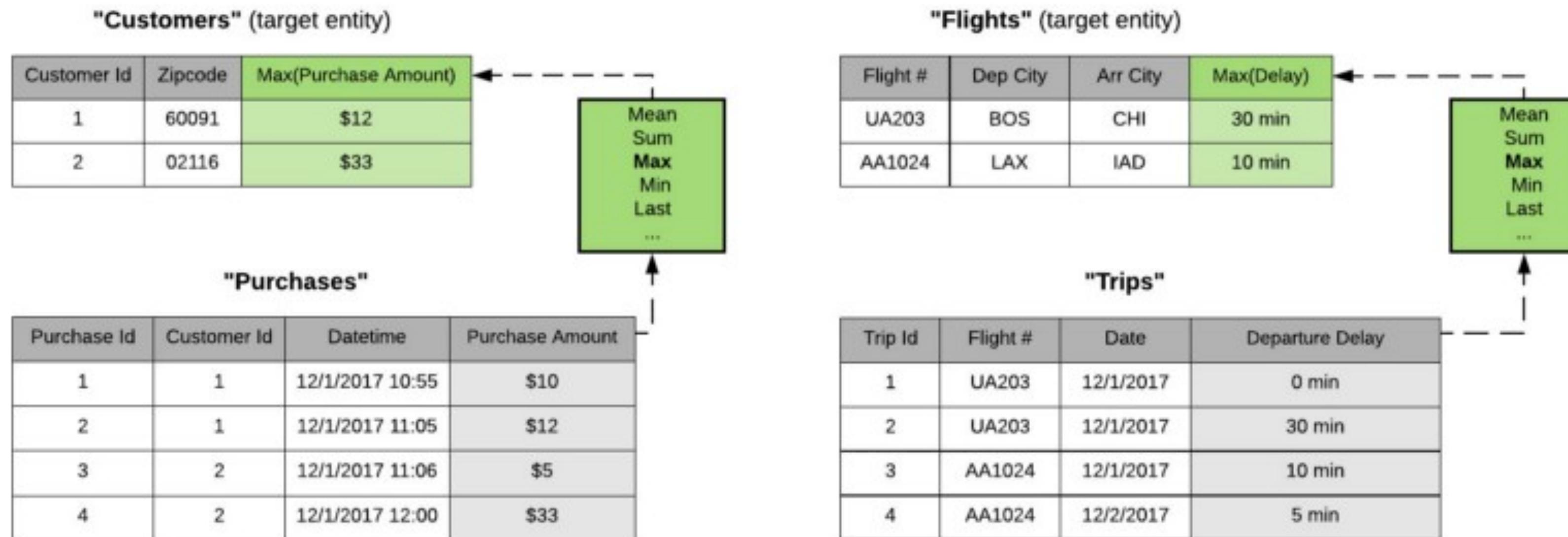
1. Features are derived from relationships between the data points in a dataset.



To calculate a customer's most expensive purchase, we apply the **Max** primitive to the purchase amount field in all related purchases. When we perform the same steps to a dataset of airplane flights, we calculate "the longest flight delay".

# Feature Tools

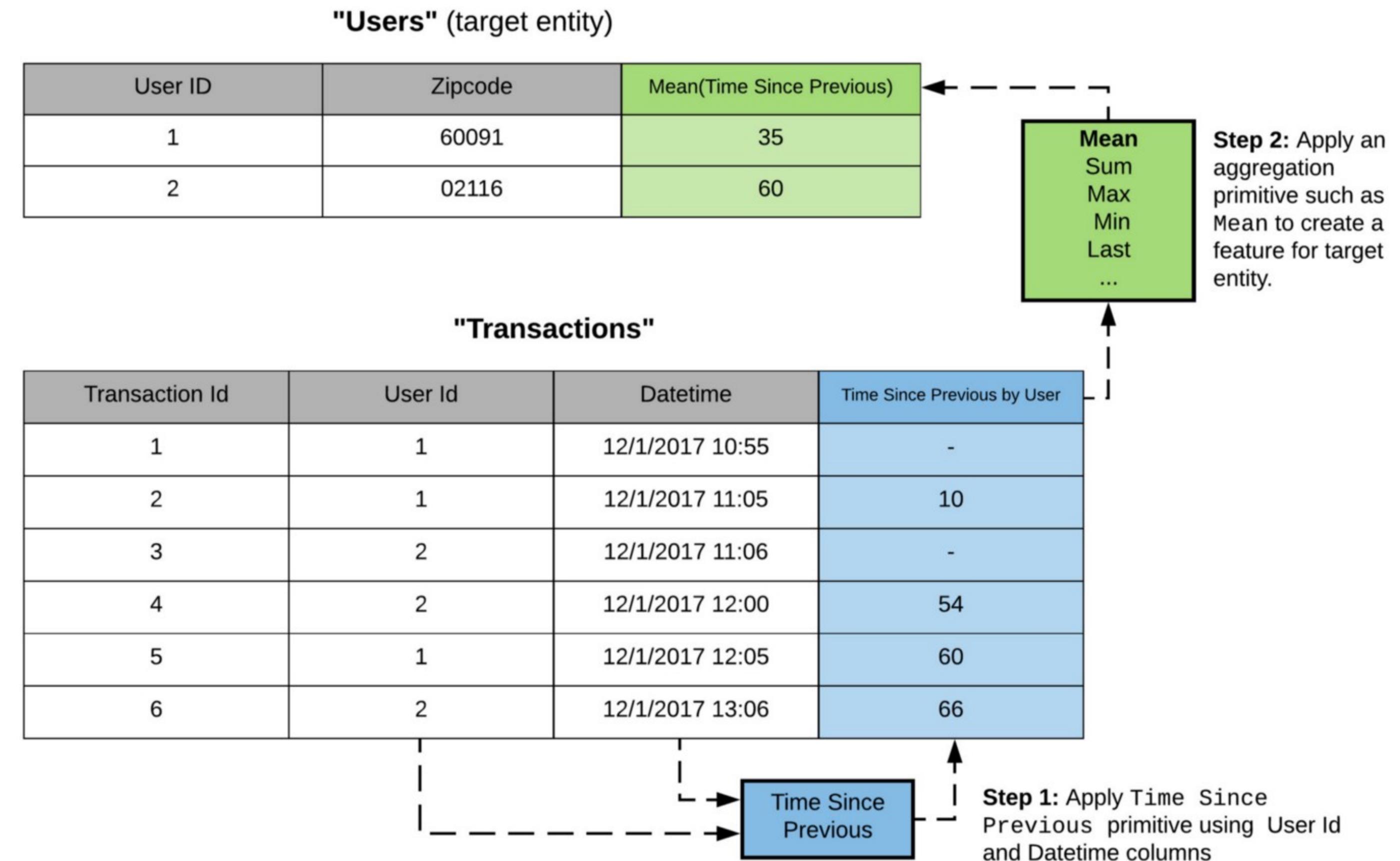
2. Across datasets, many features are derived by using similar mathematical operations.



To calculate a customer's most expensive purchase, we apply the **Max** primitive to the purchase amount field in all related purchases. When we perform the same steps to a dataset of airplane flights, we calculate "the longest flight delay".

# Feature Tools

3. New features are often composed from utilizing previously derived features.



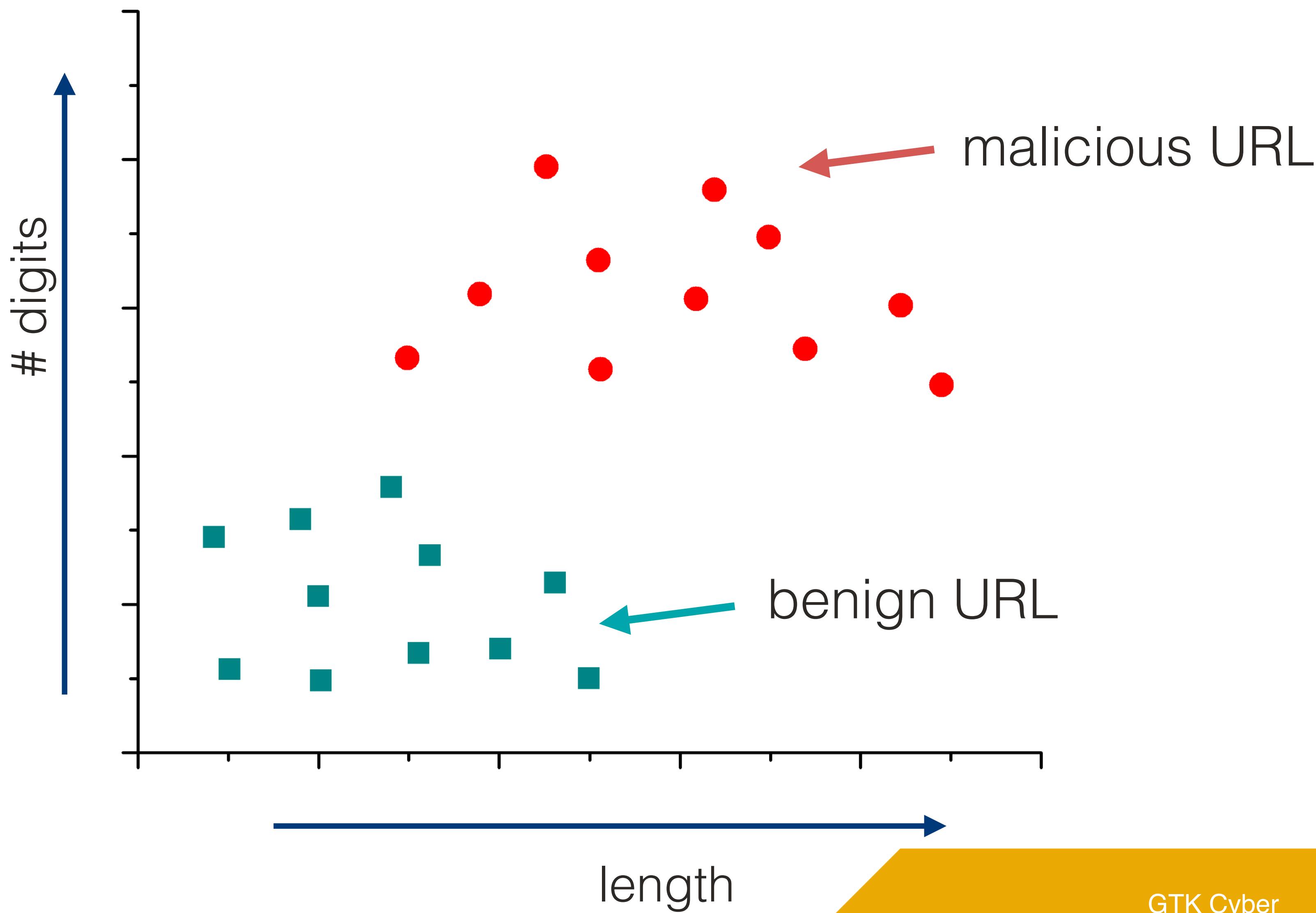
# In-Class Exercise

Please complete Worksheet 2: Feature Engineering

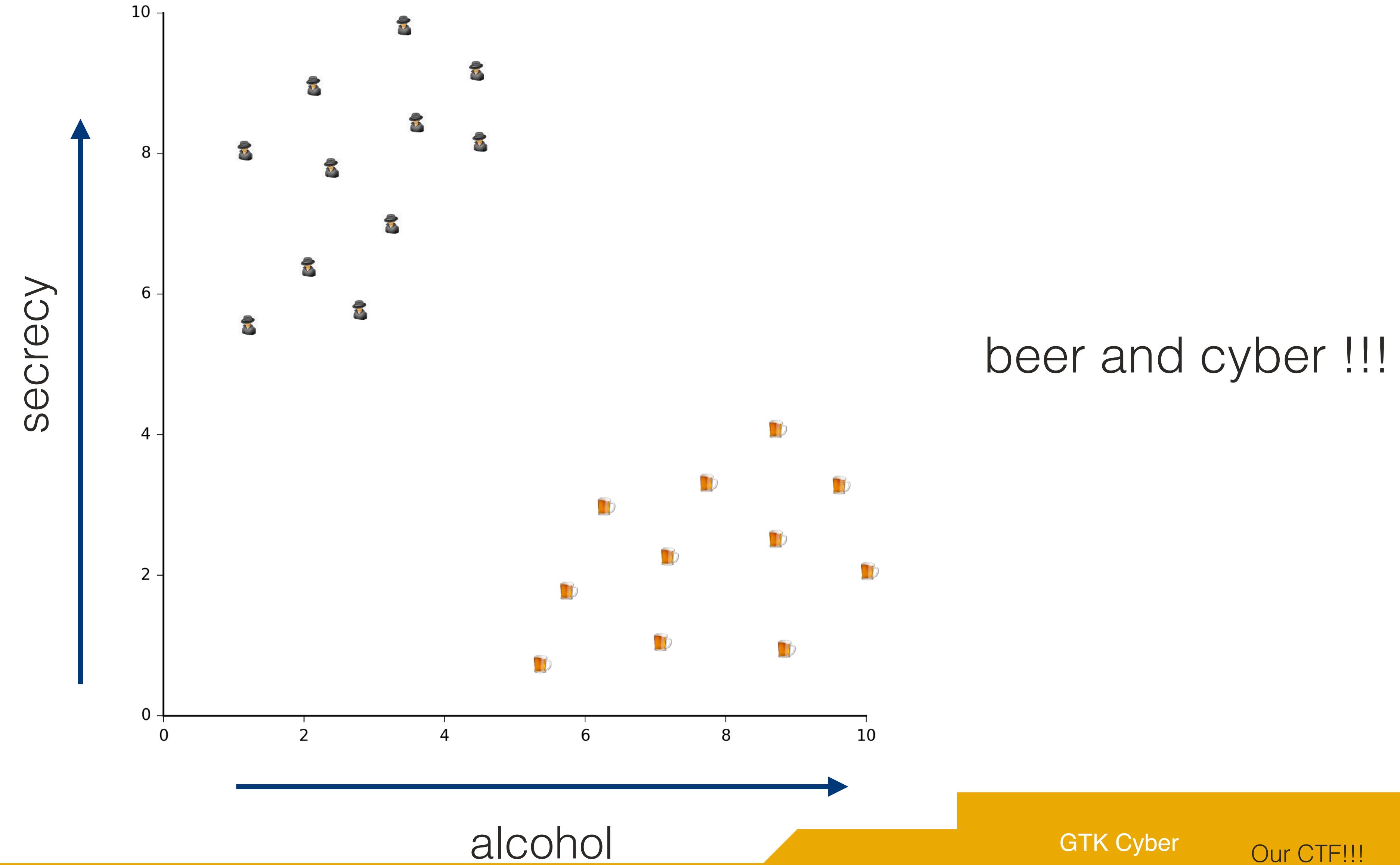
# **Module 4: Machine Learning**

Part 2: Modeling

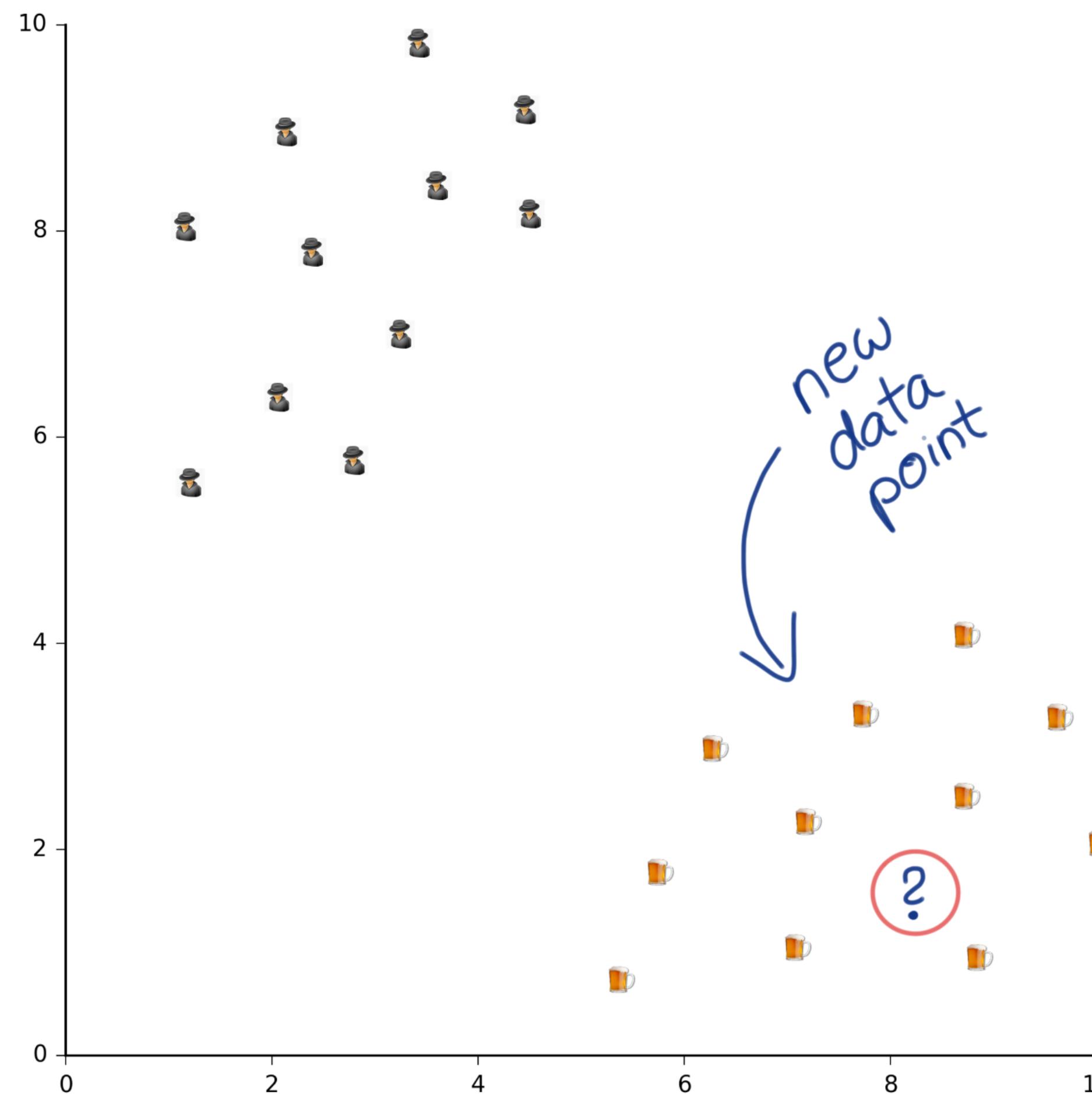
# Features and Labels



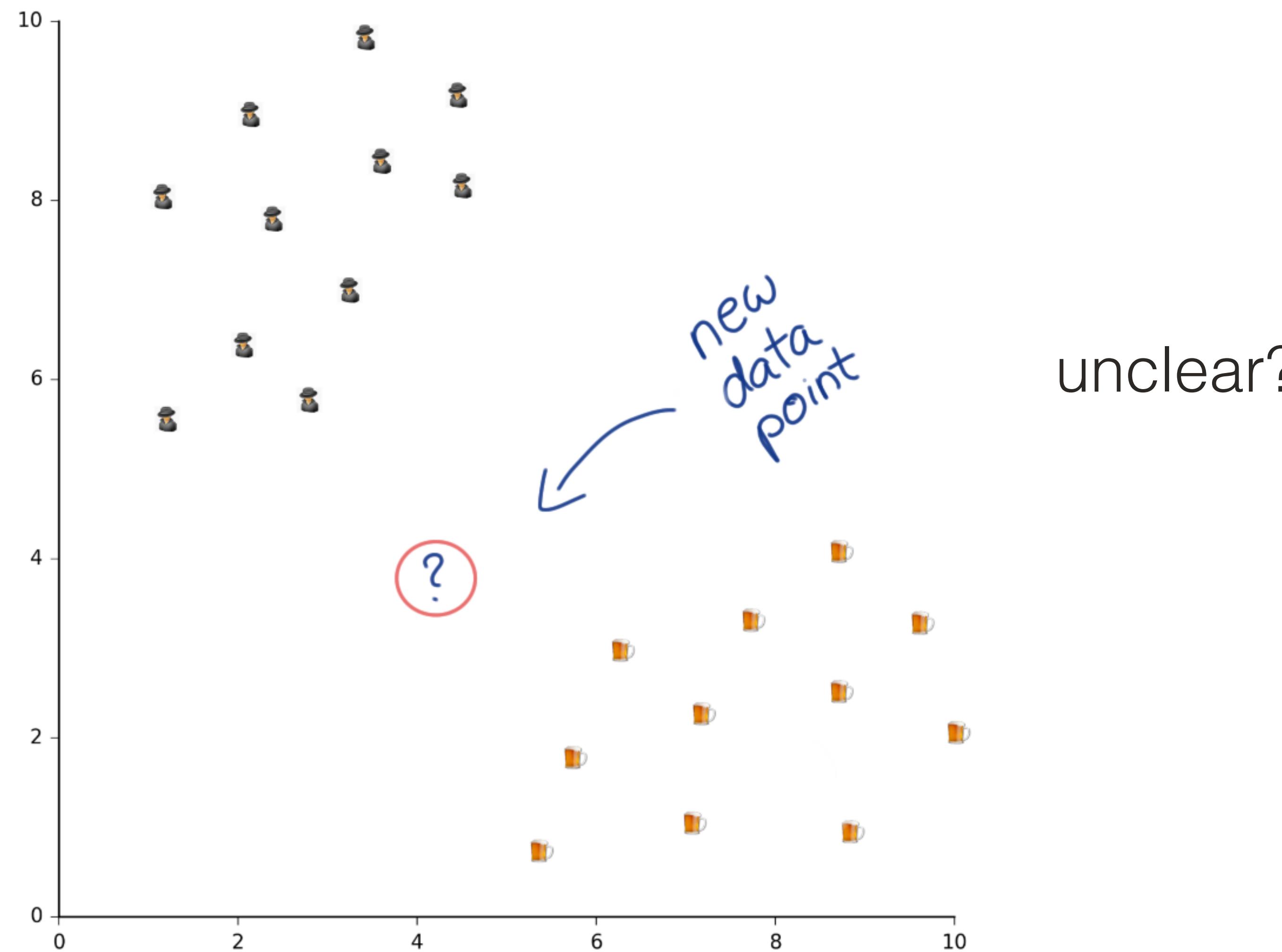
# FUN Features and Labels



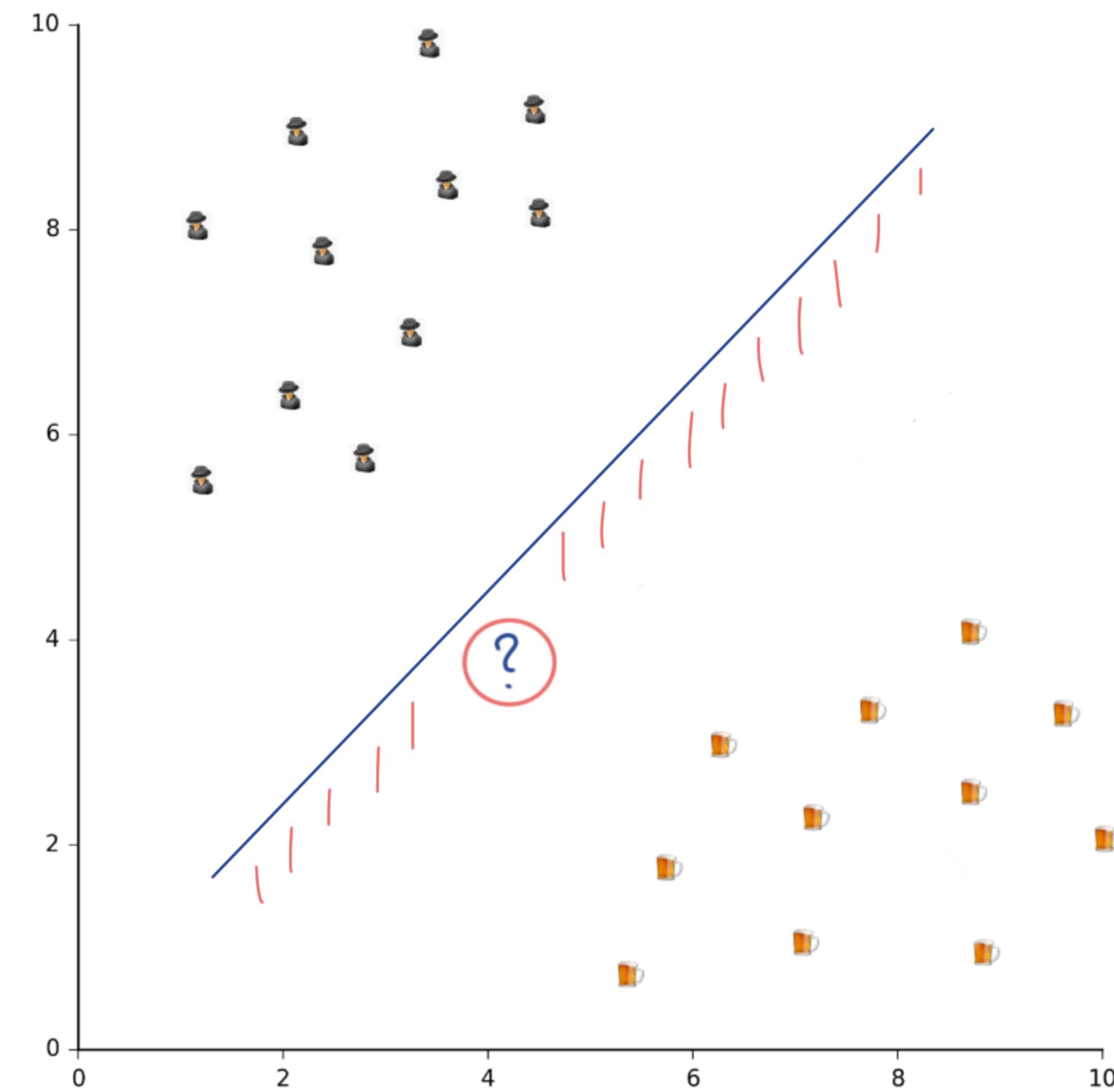
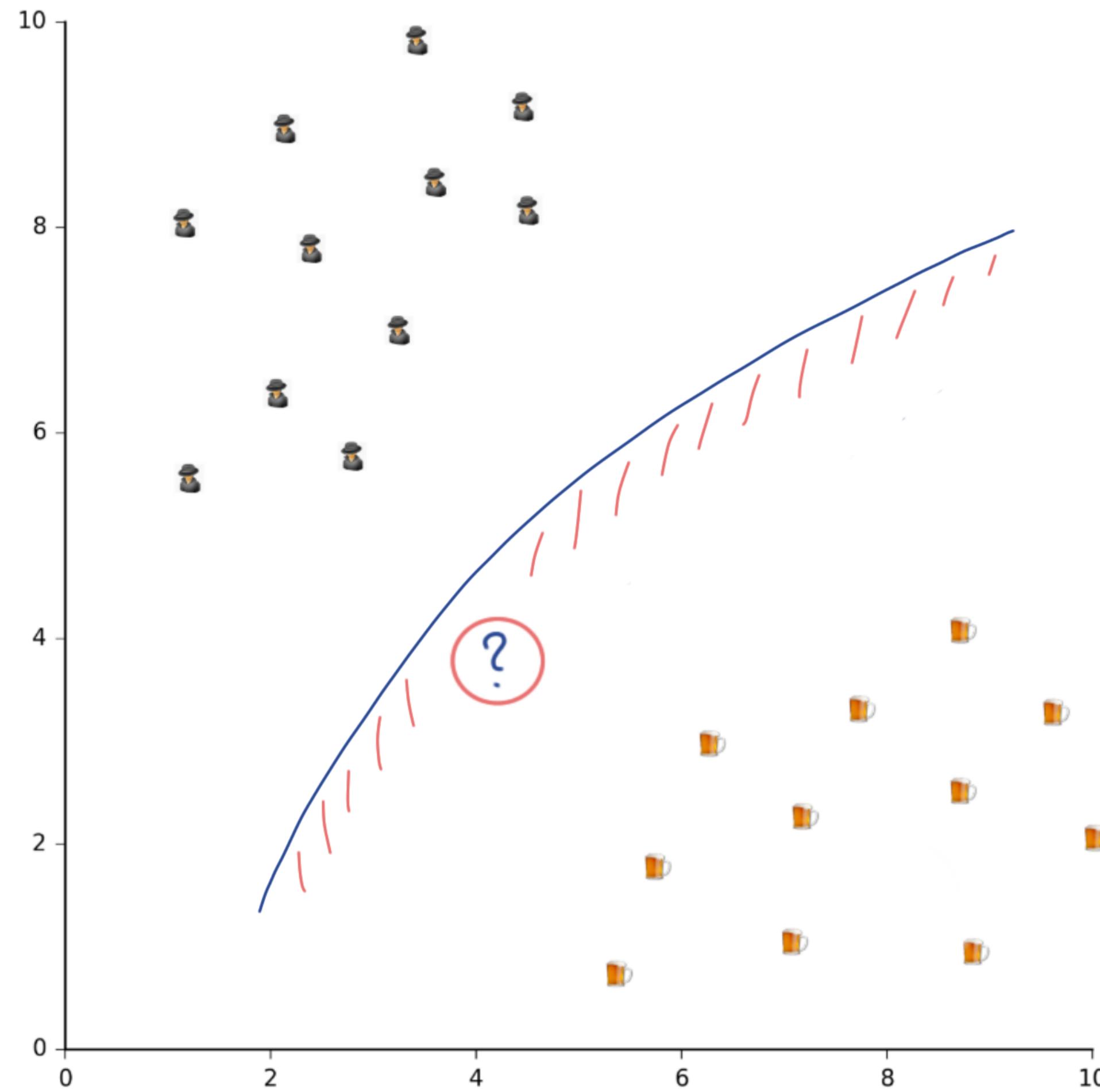
# Making a new decision



# Making a new decision



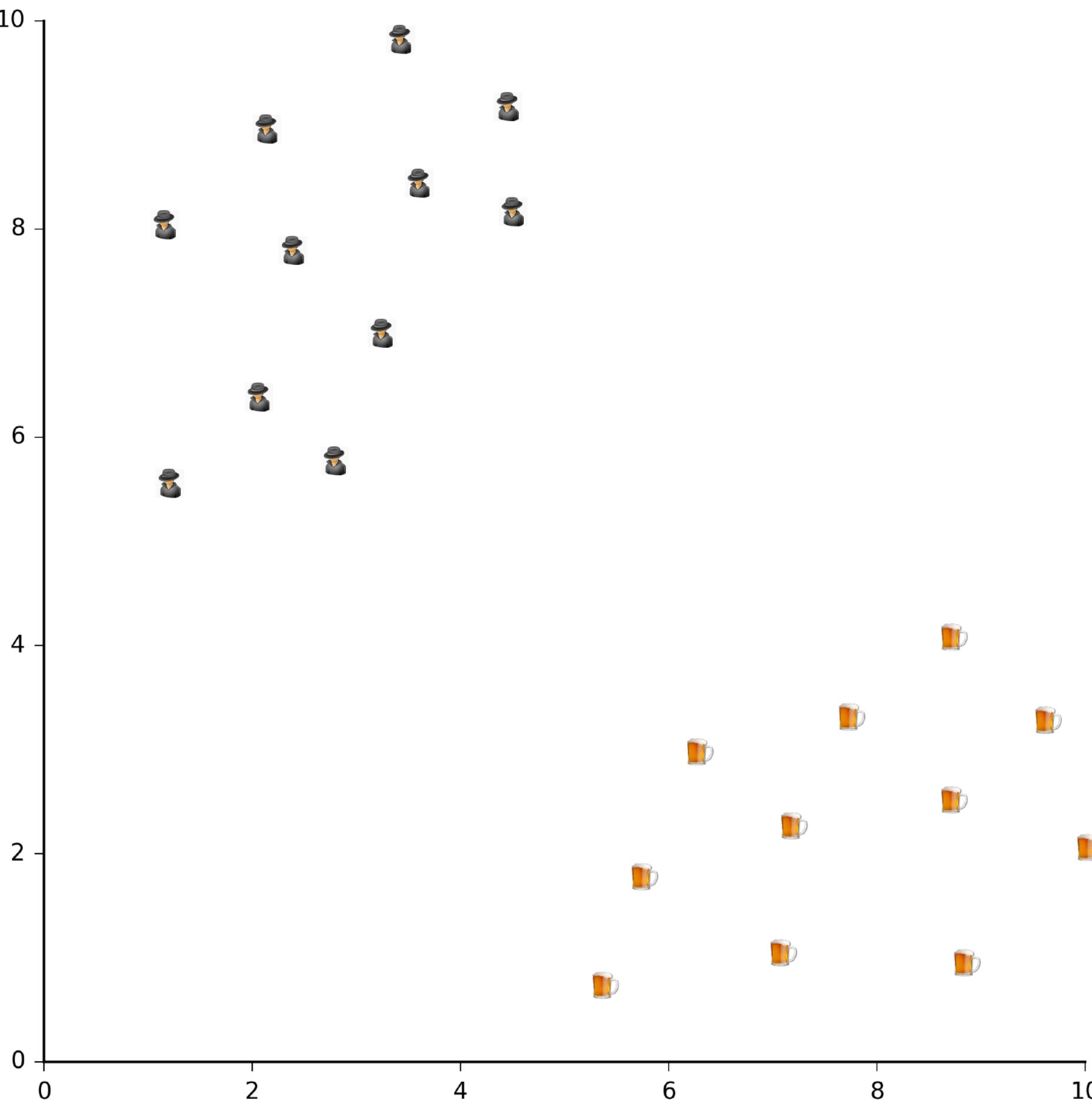
# Decision surface concept



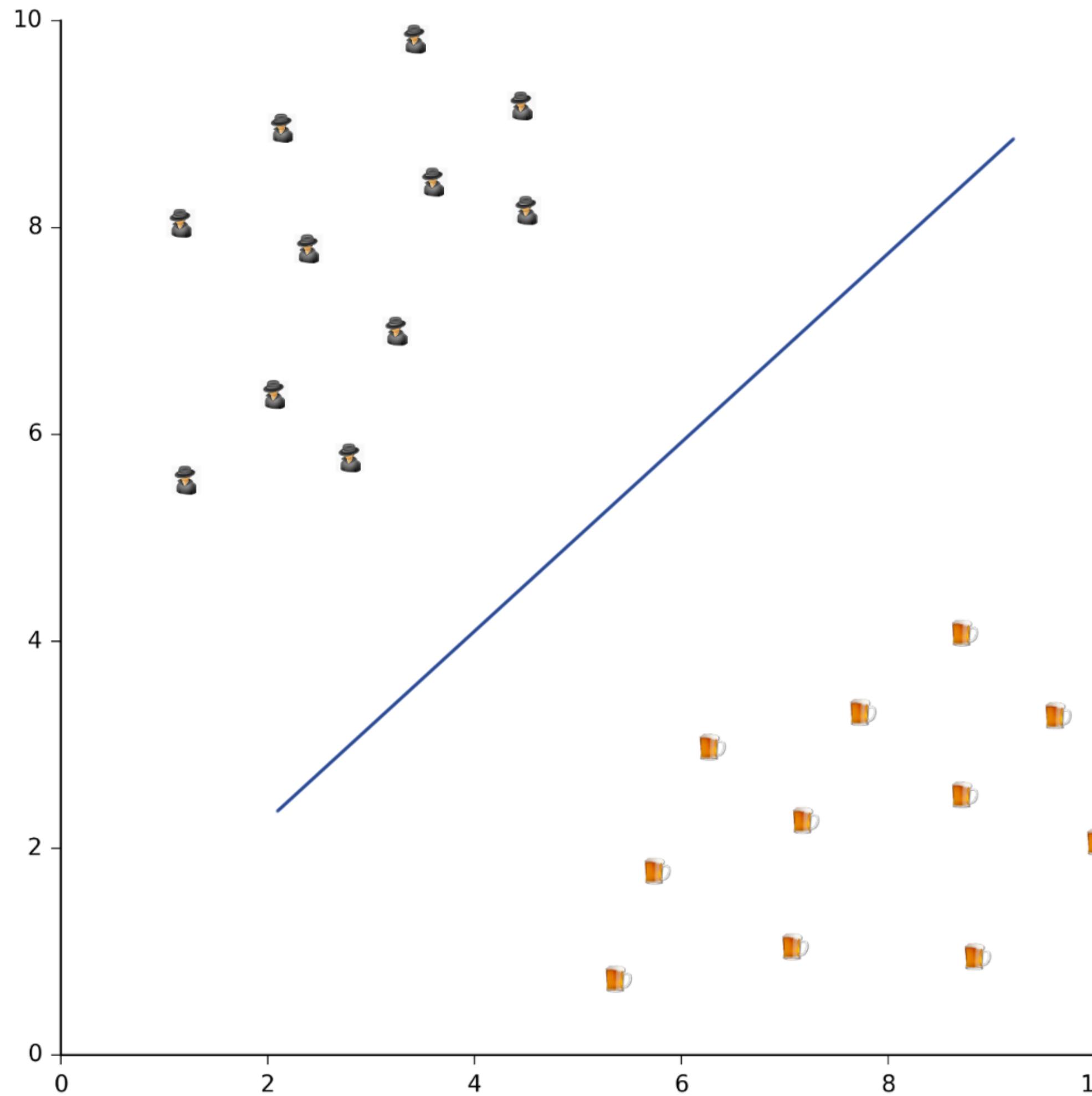
# **Support Vector Machines (SVM)**



# How can we linearly separate beer and cyber?

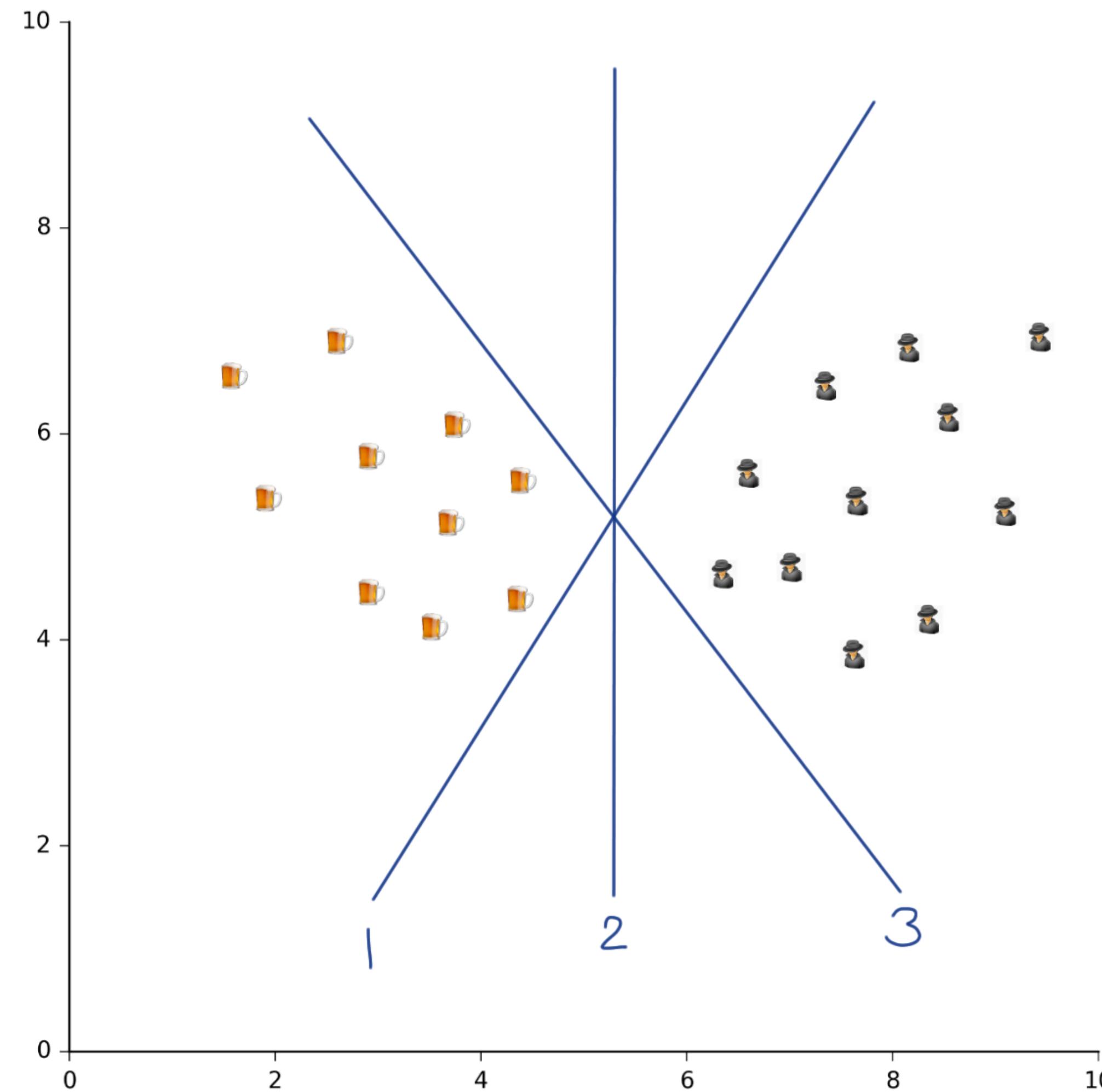


# How can we linearly separate beer and cyber?

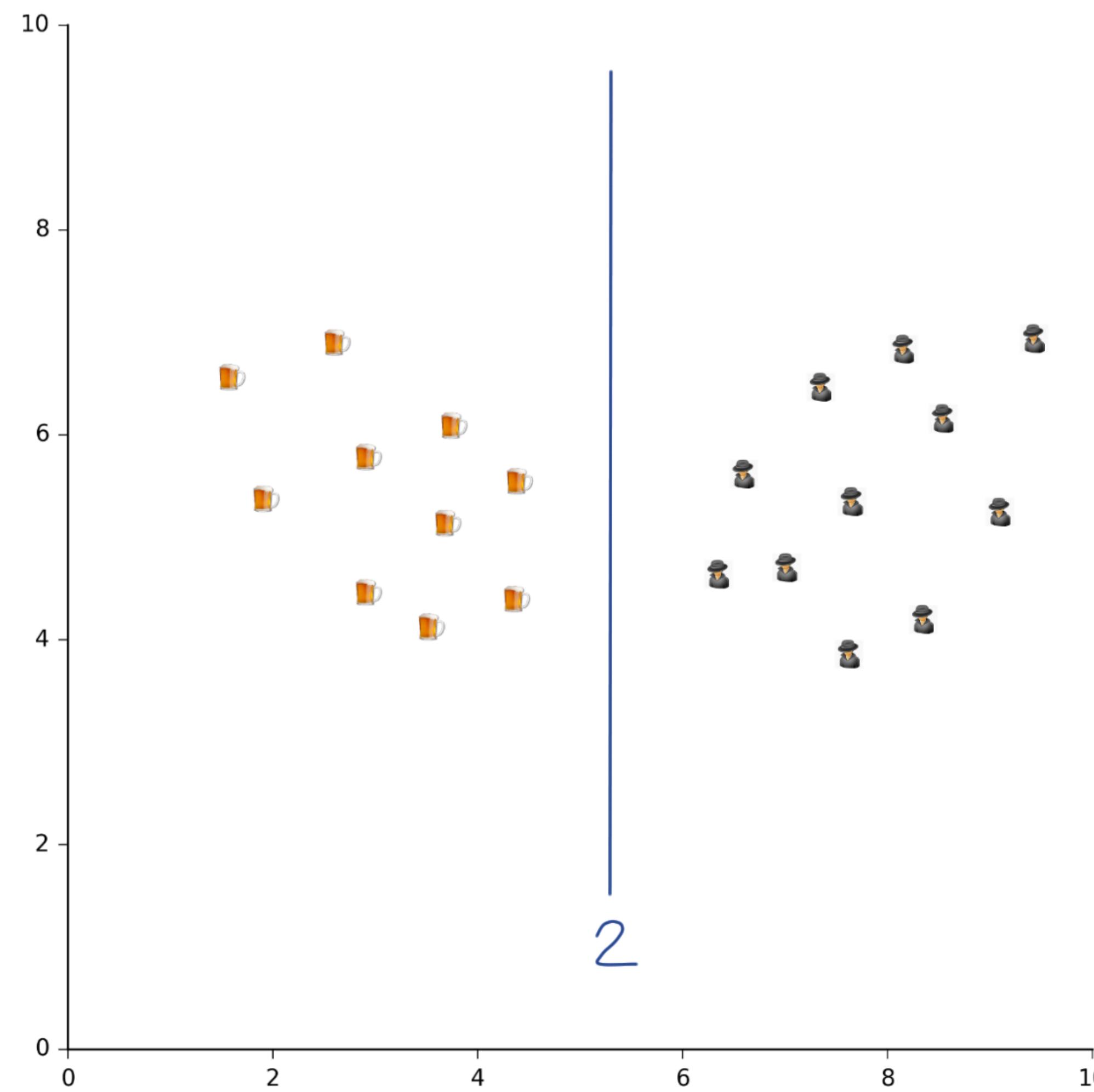


Answer: straight line

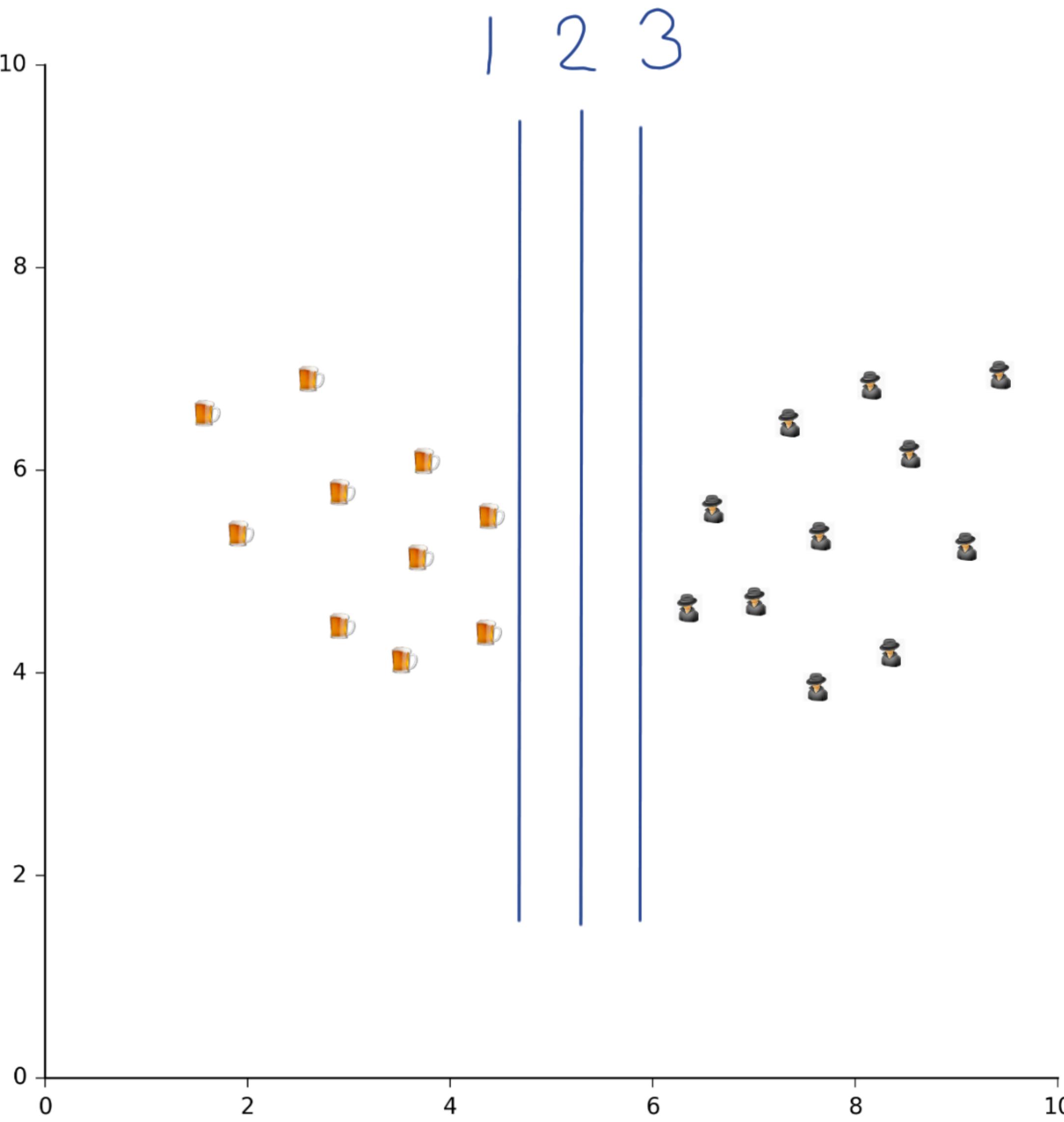
# Same question, choose the best line!

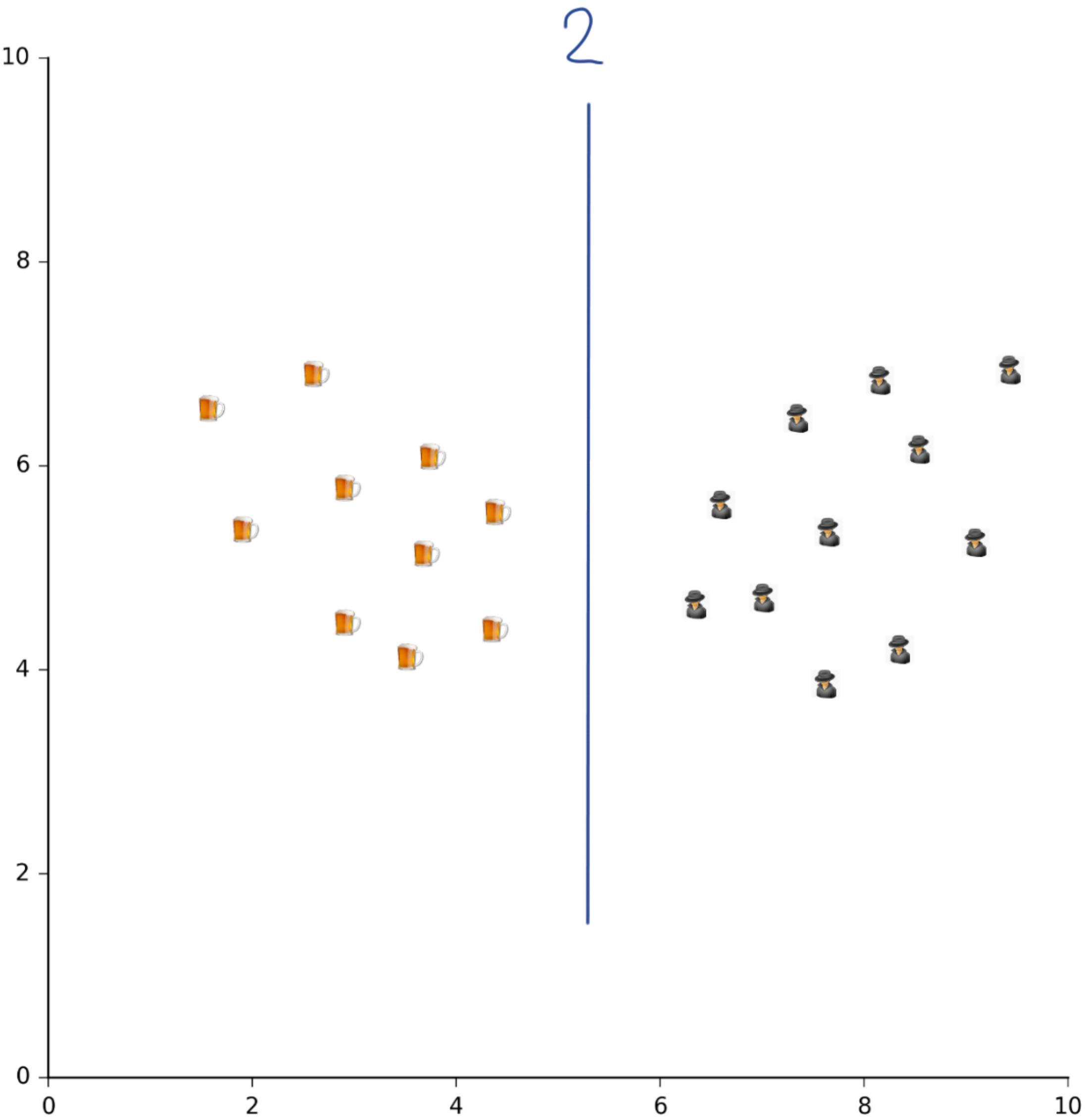


# Same question, choose the best line!



Answer: 2

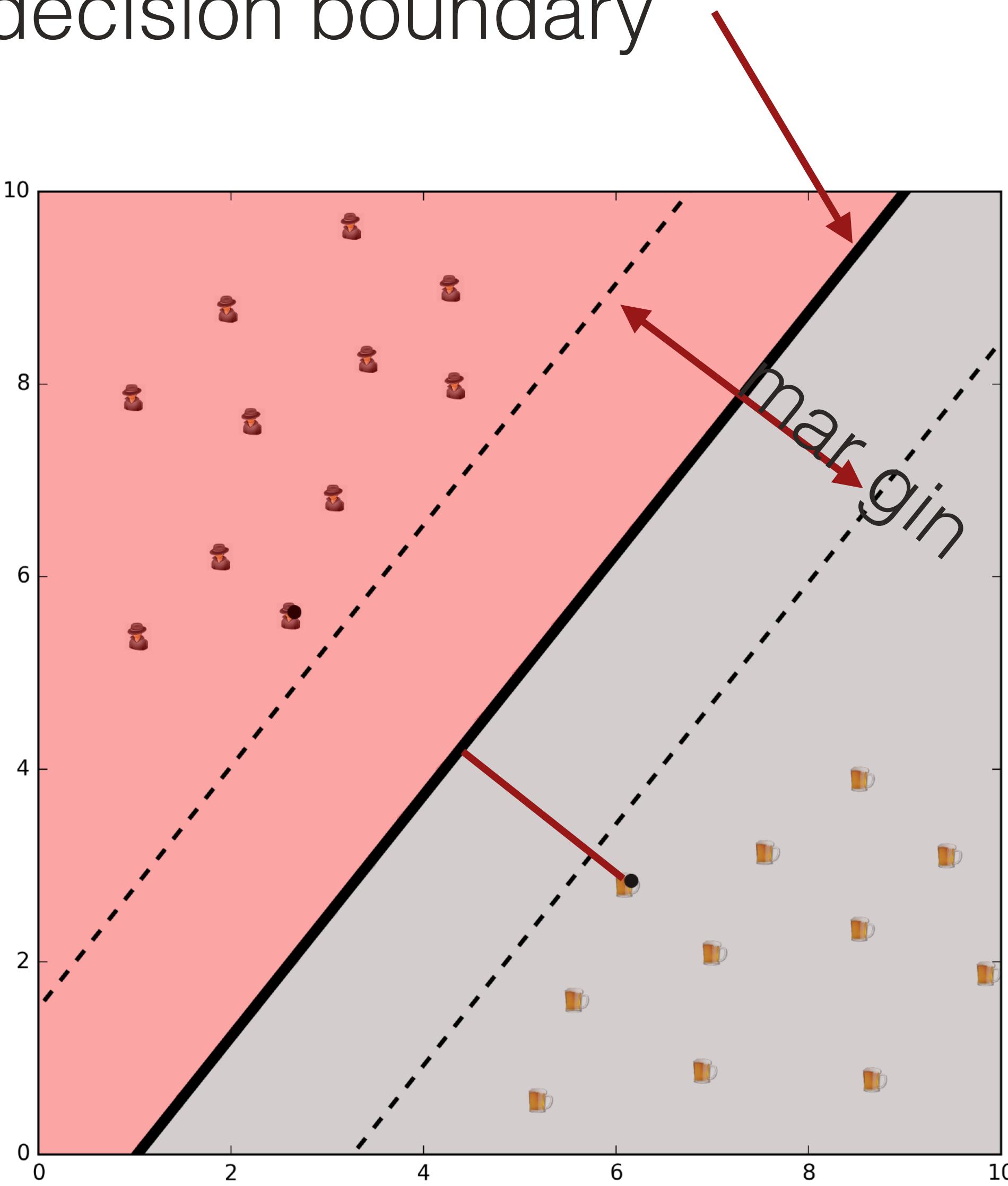




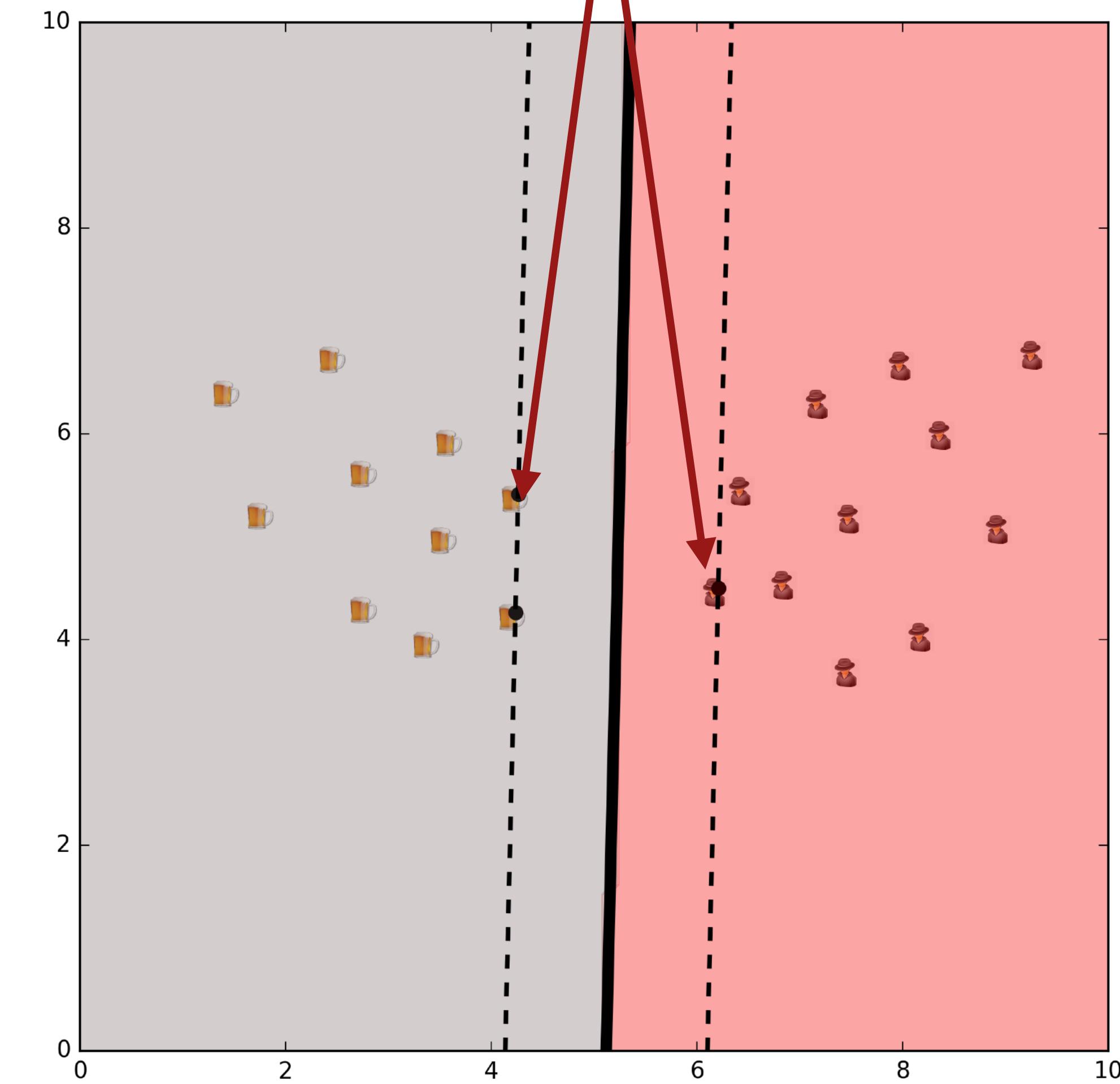
Answer: 2

# Open BlackBox...

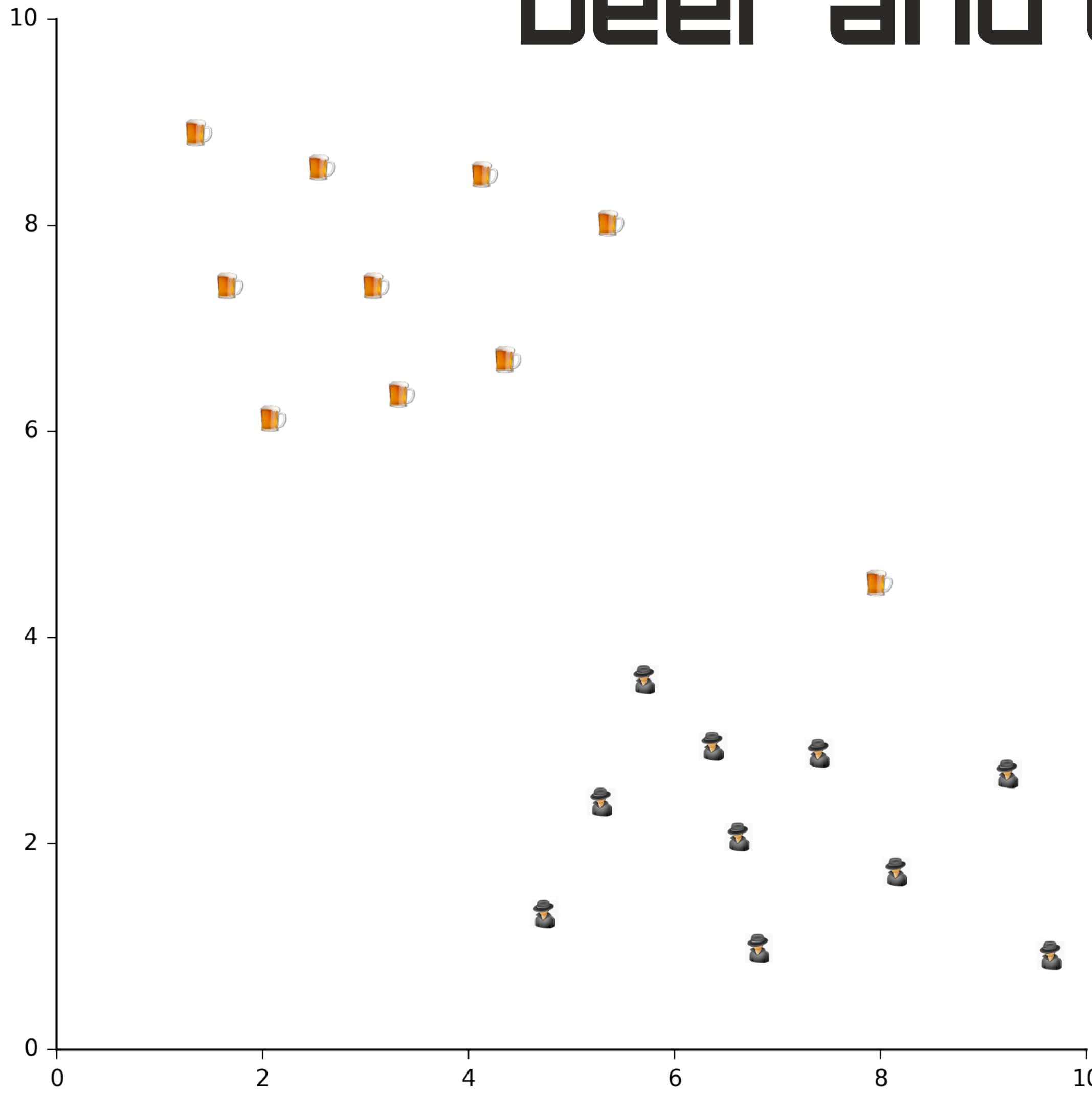
decision boundary



support vectors

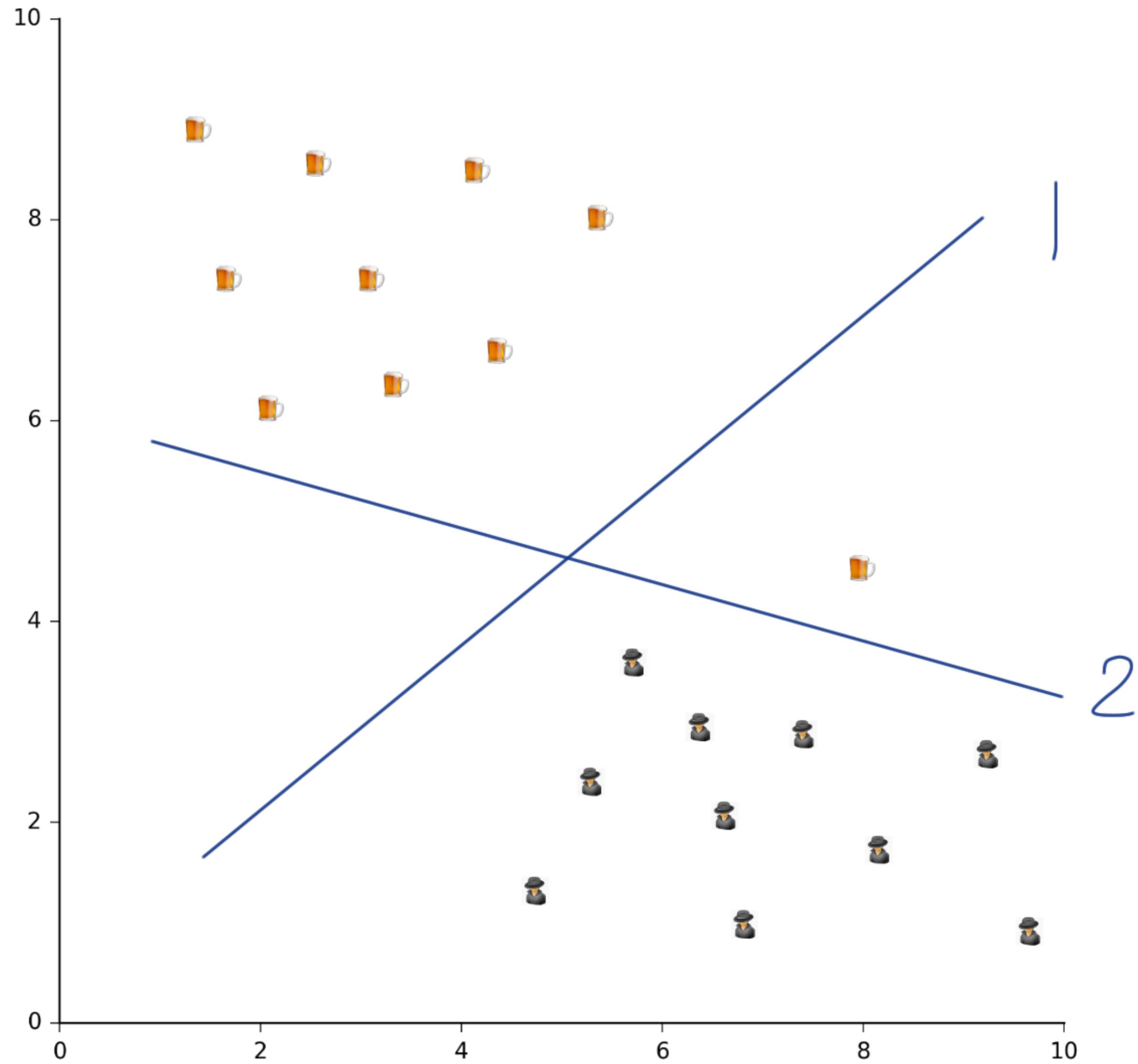


# Another test, how do we best separate beer and cyber linearly?

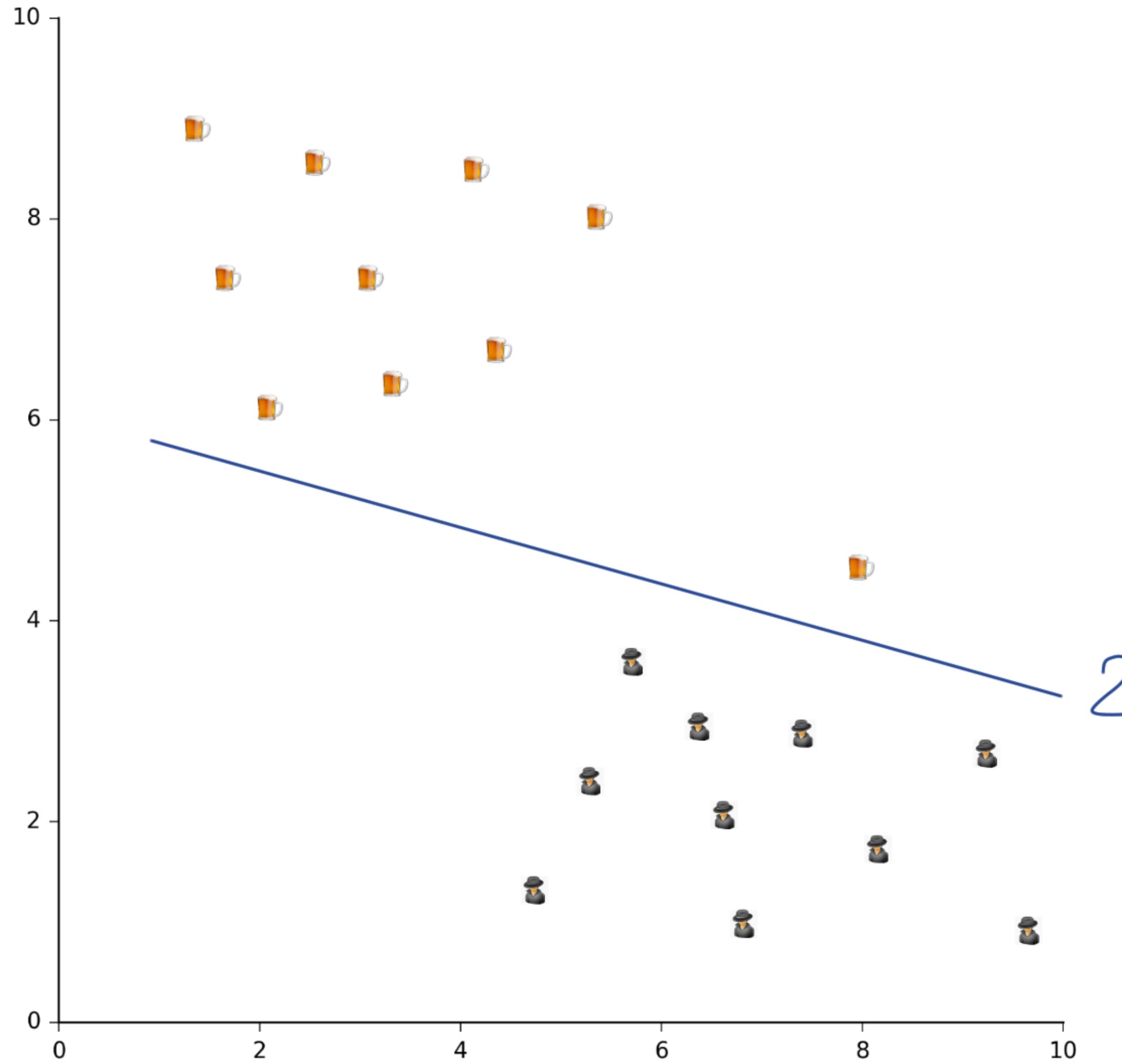


GTK Cyber

# Choose the best line?

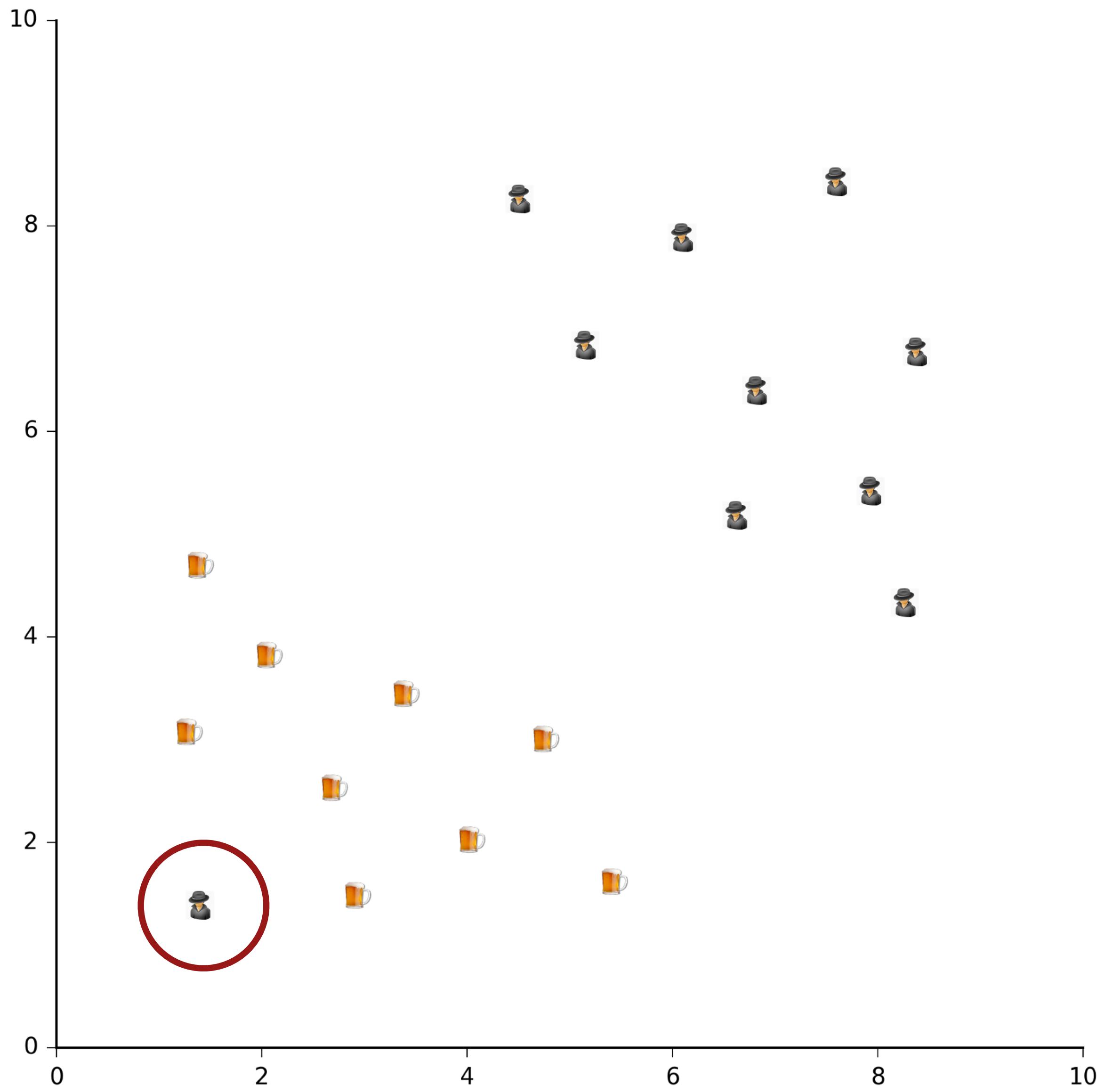


# Choose the best line!

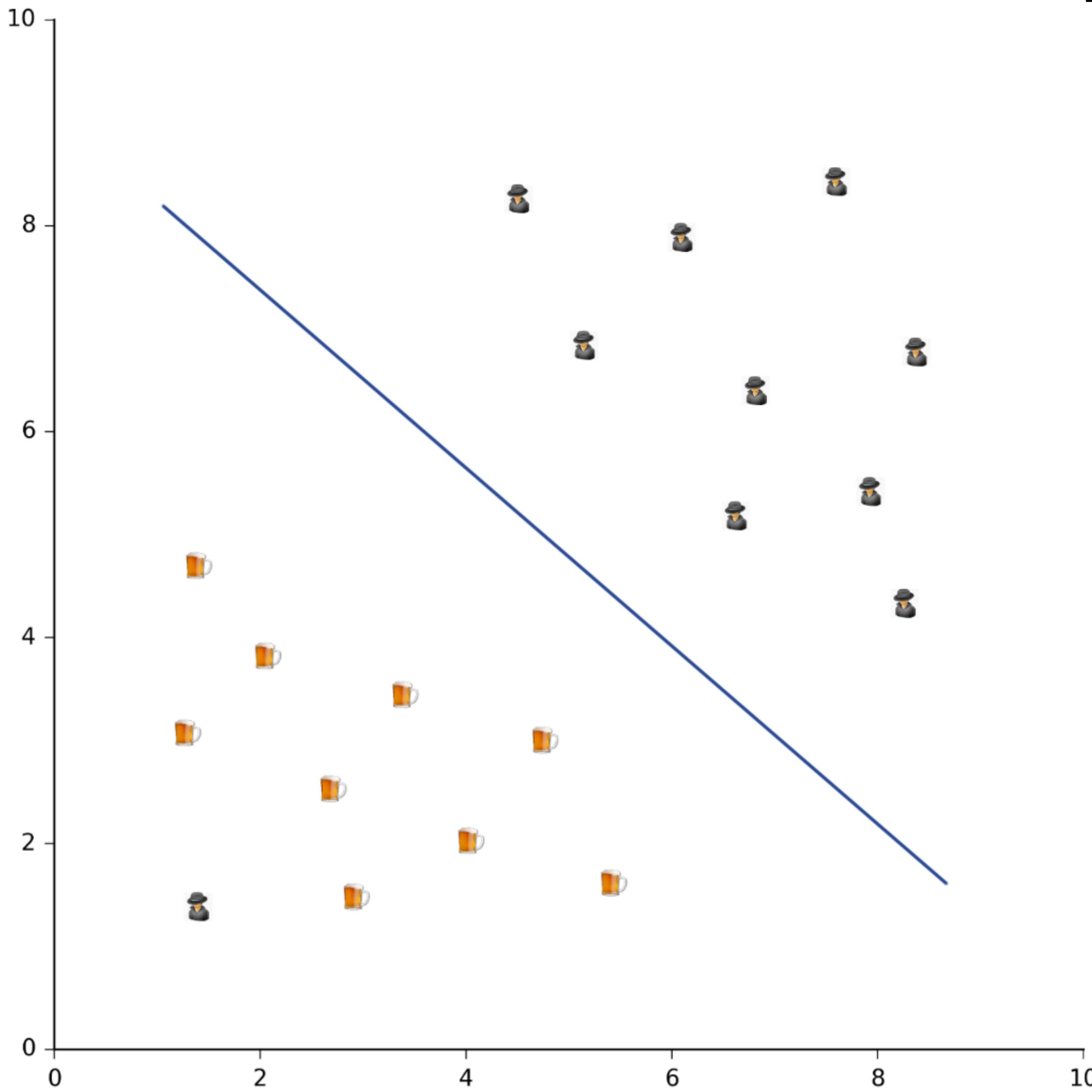


Answer: 2

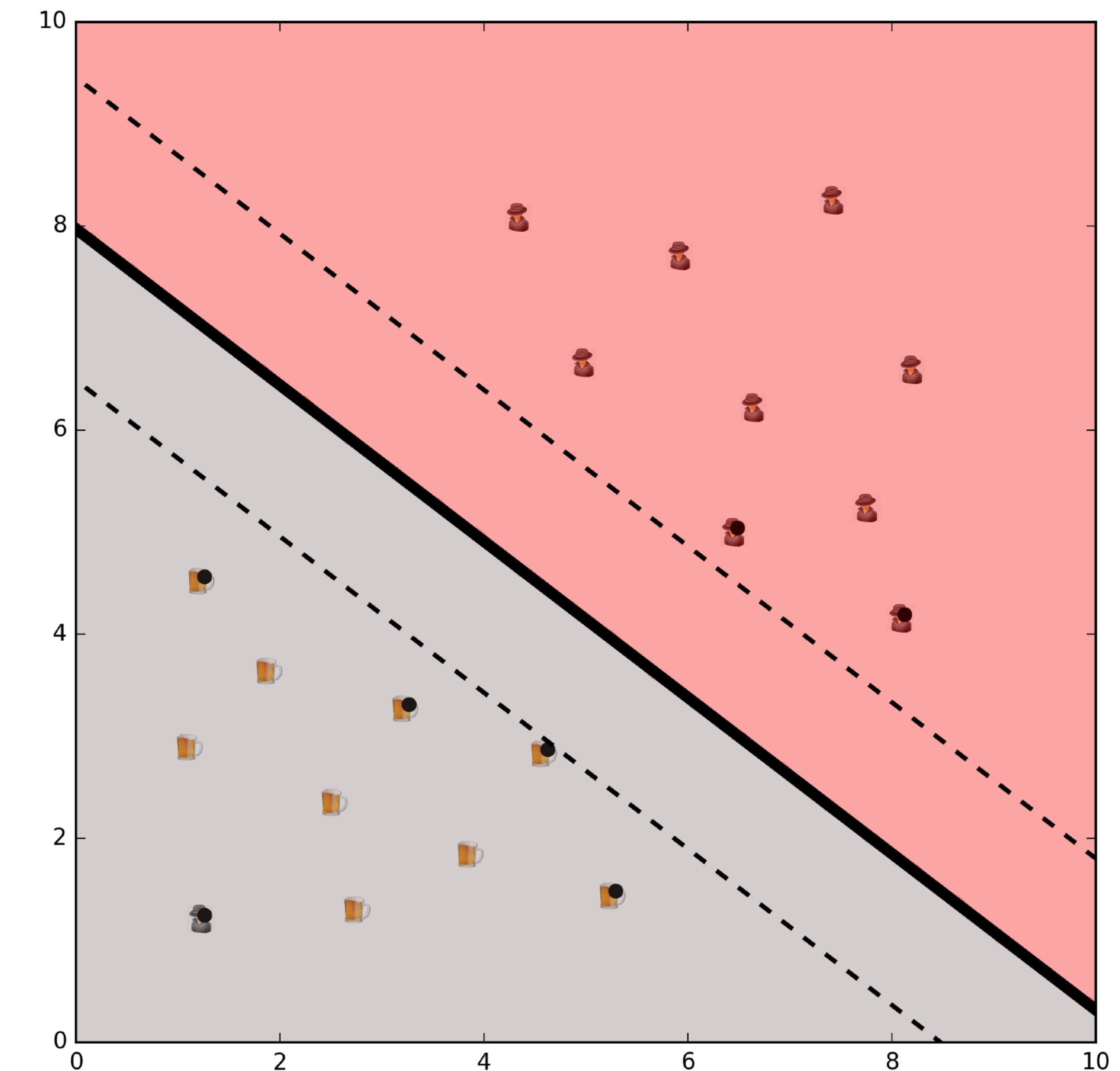
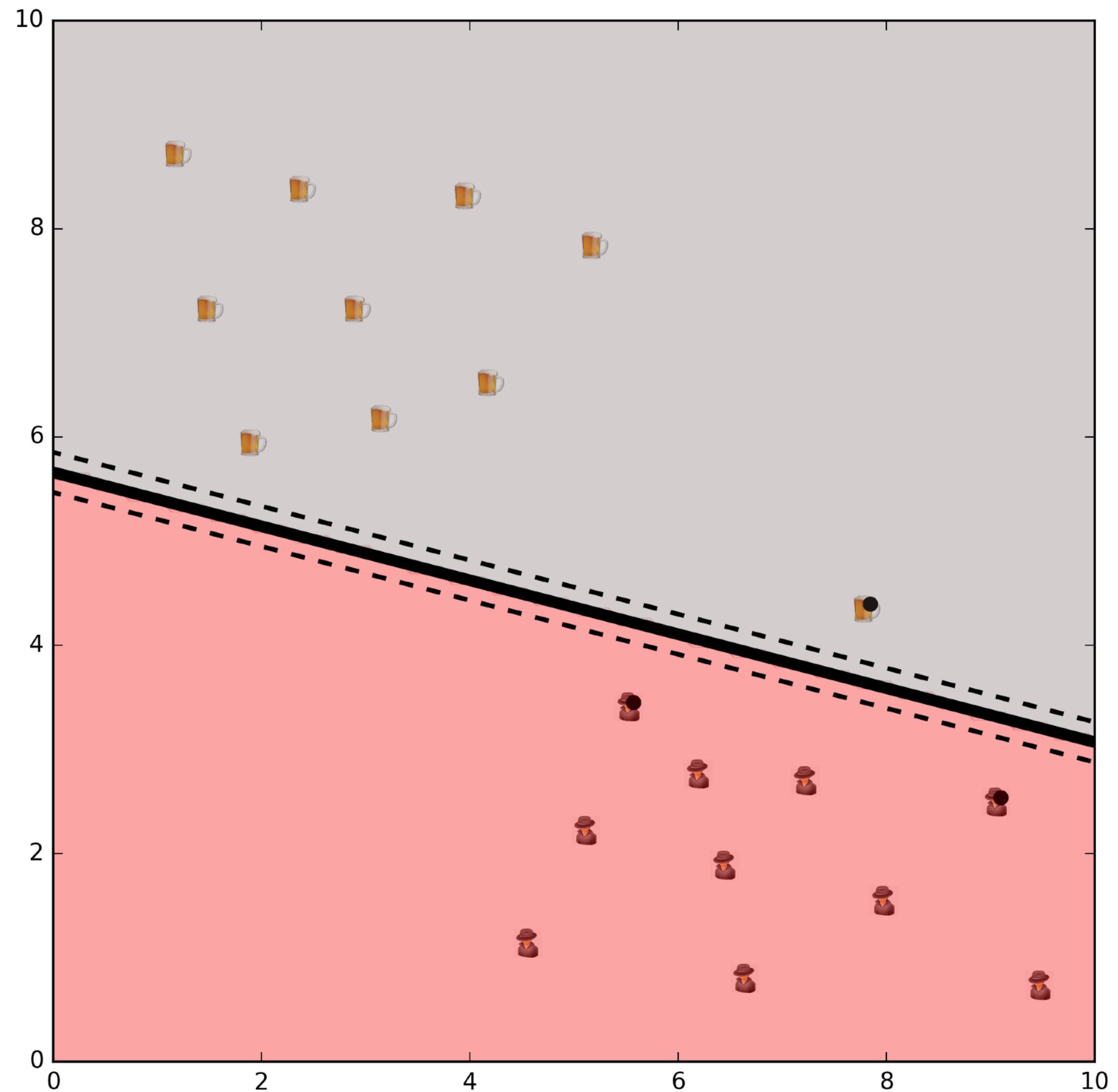
# Wait and now?



# Just Ignore?



# Open BlackBox...

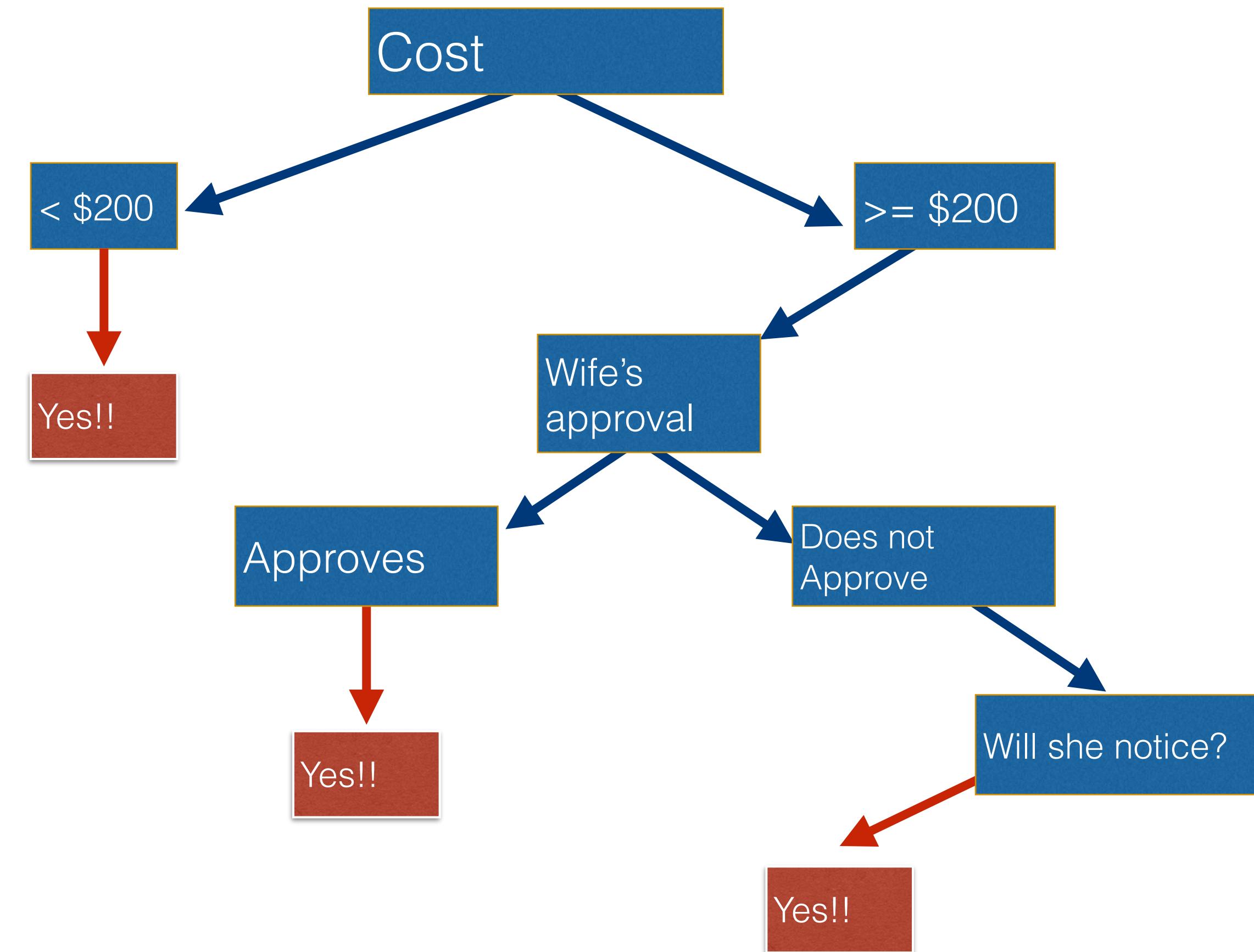


# **Simple Decision Tree**

## **(DT)**



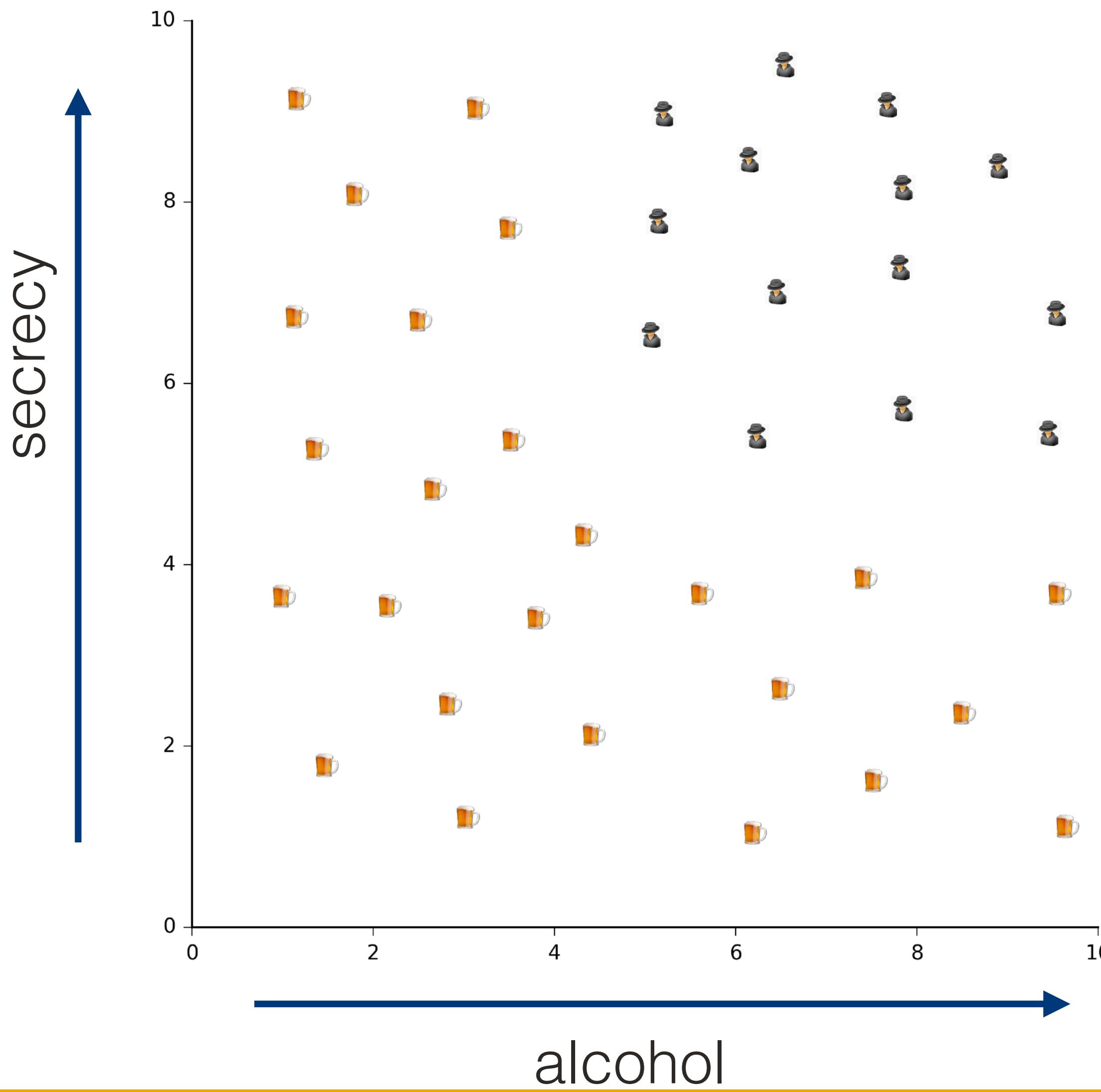
# Should I buy a new tech gadget?



This is Charles'  
example!

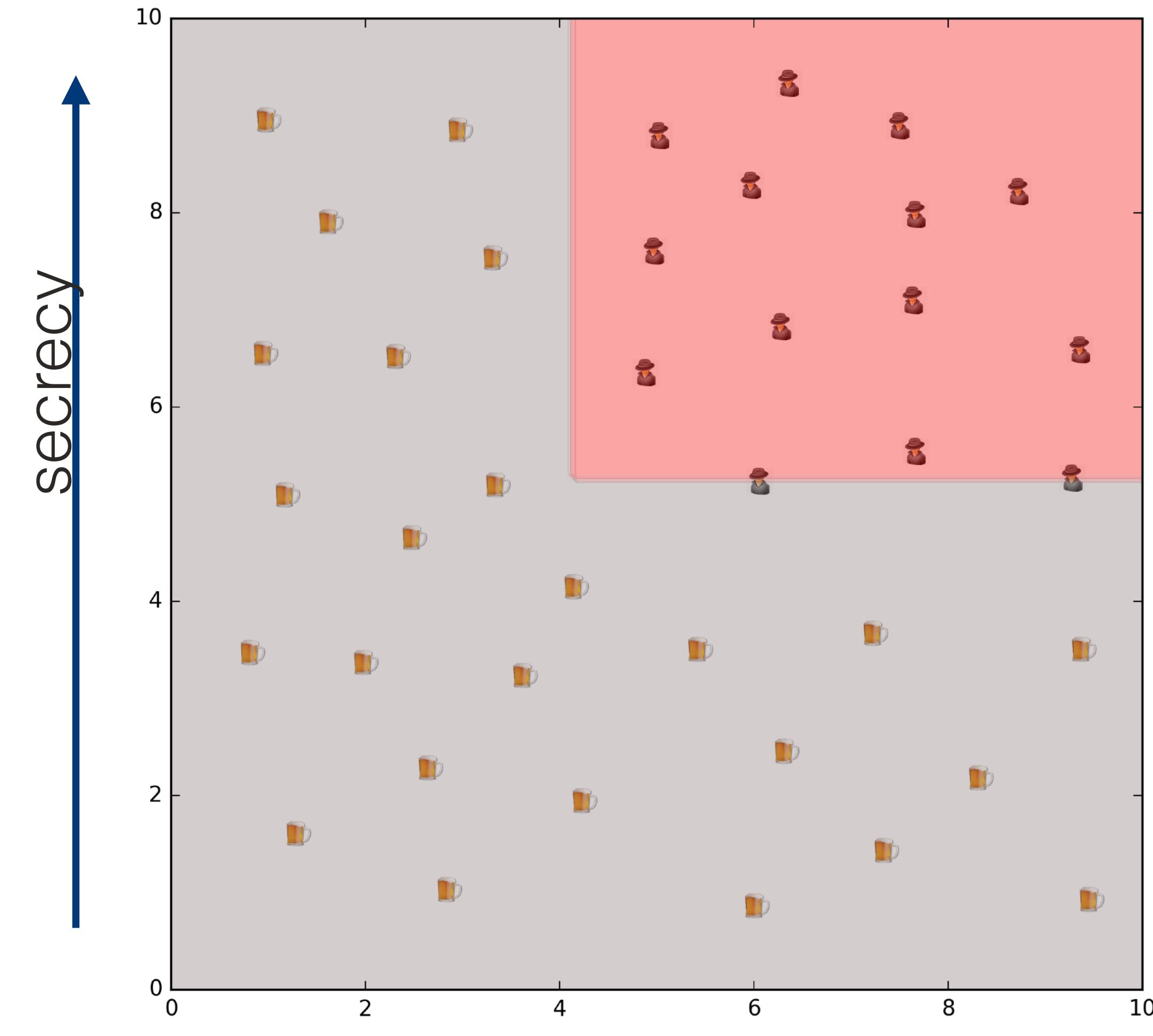
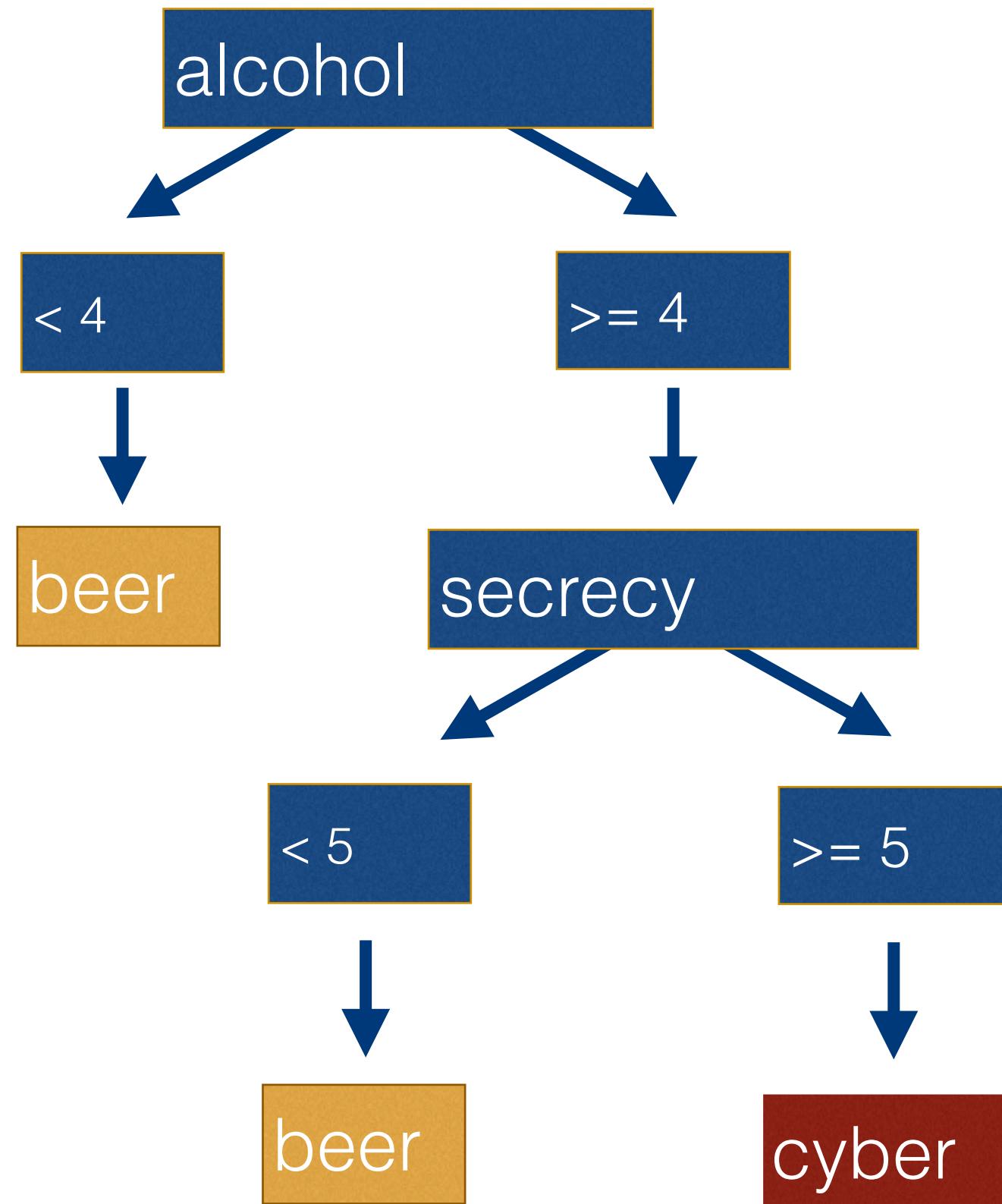


# How can we separate beer and cyber?

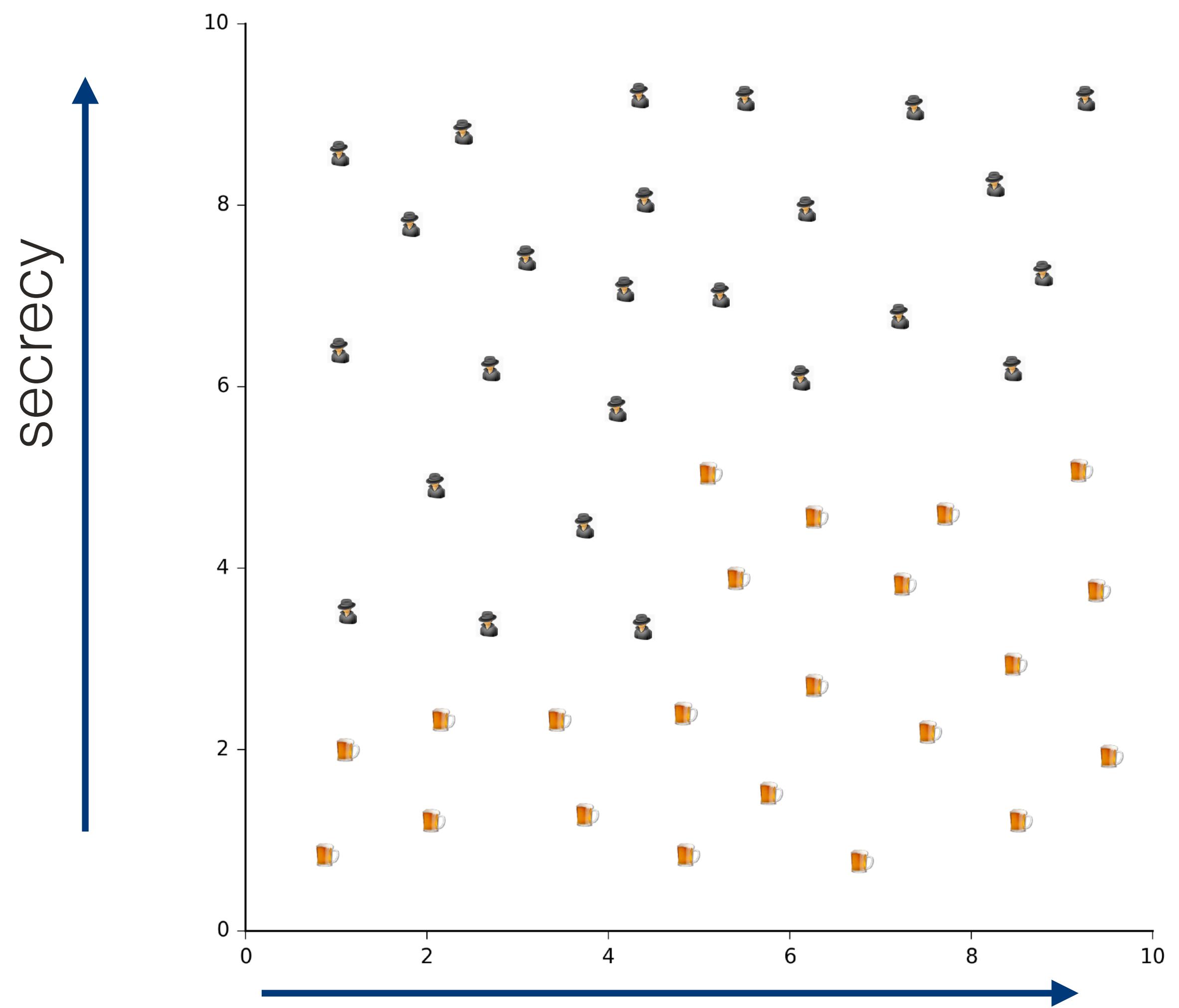


GTK Cyber

# How can we separate beer and cyber?

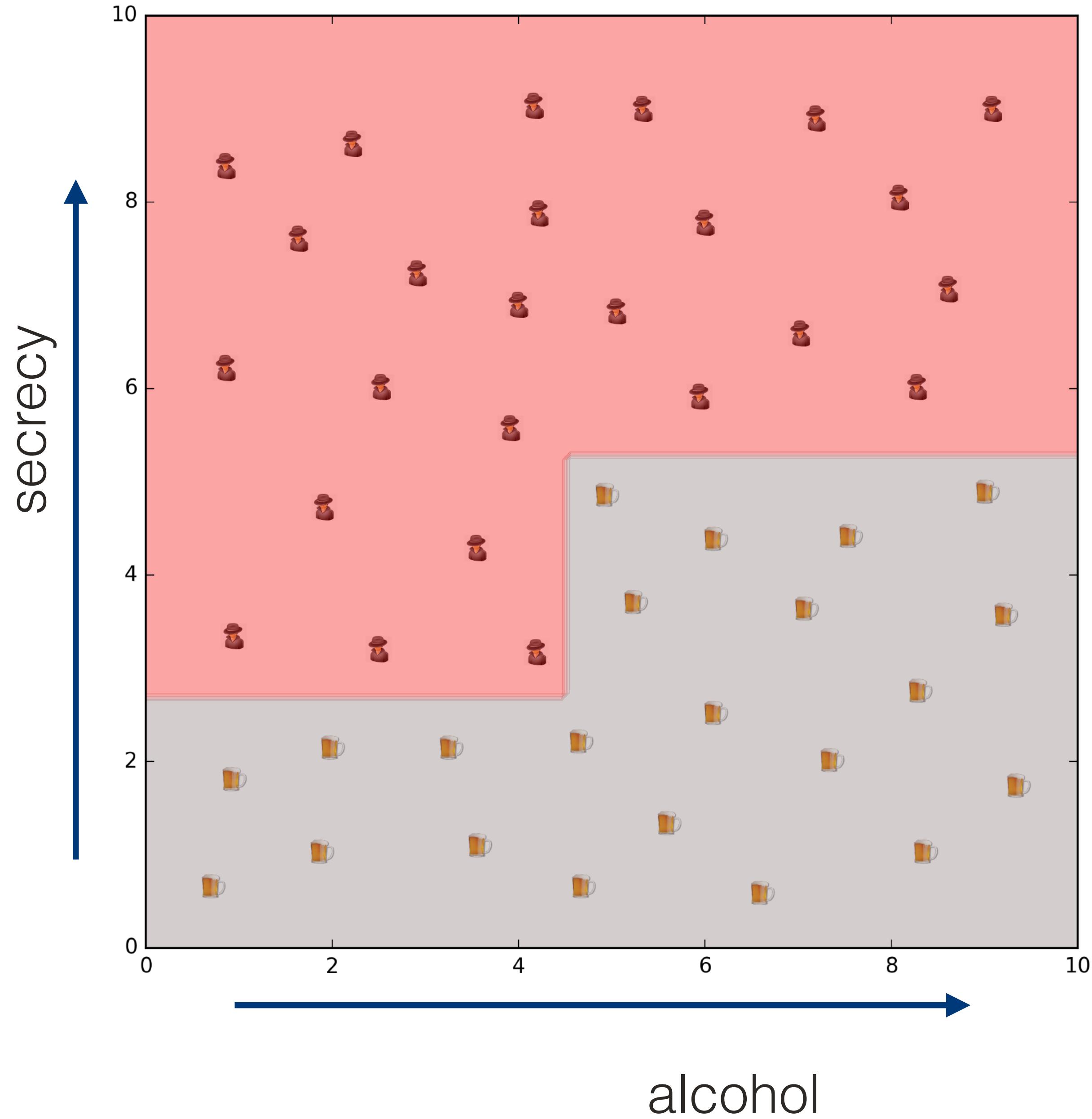


alcohol



alcohol

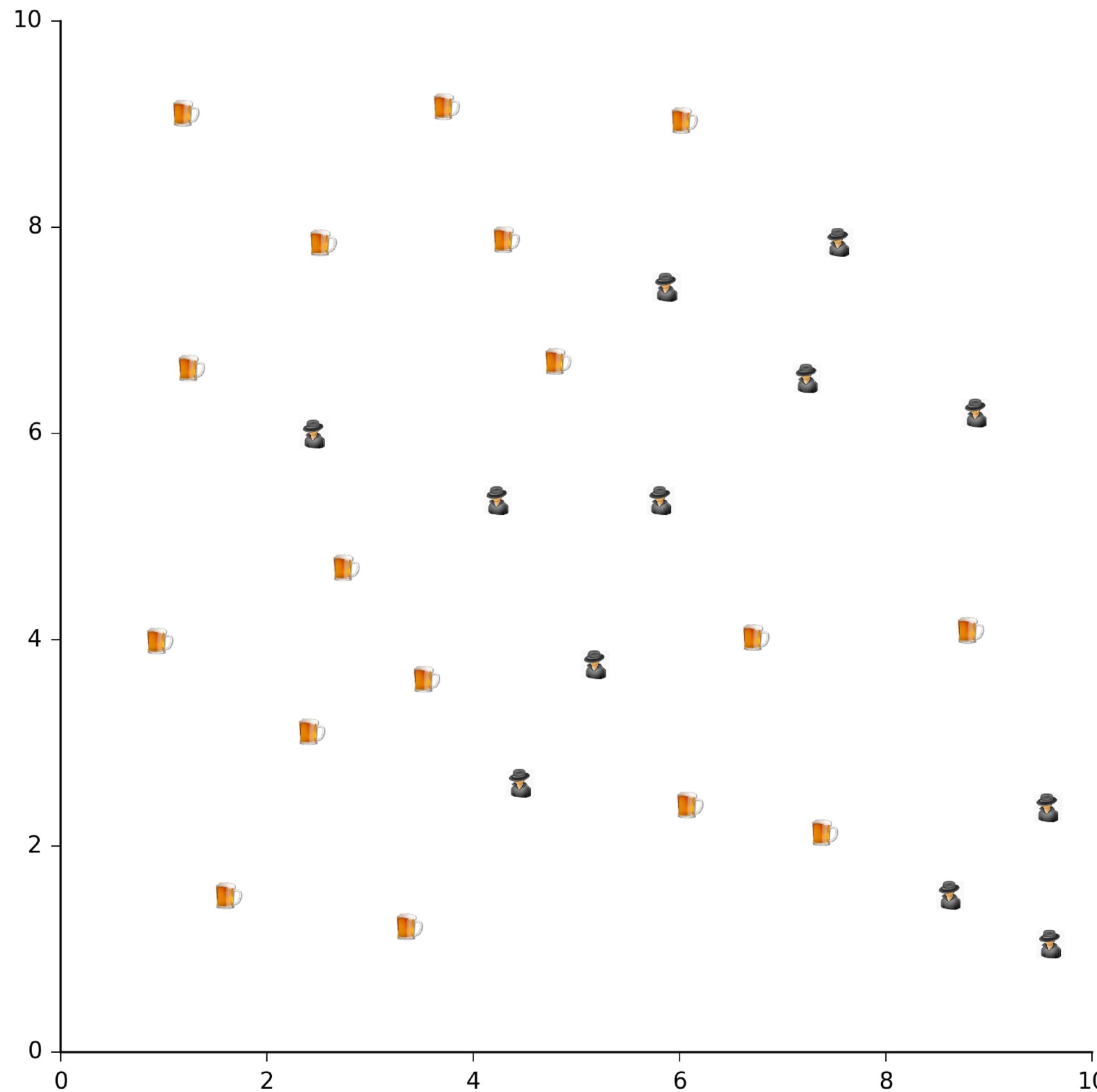
GTK Cyber



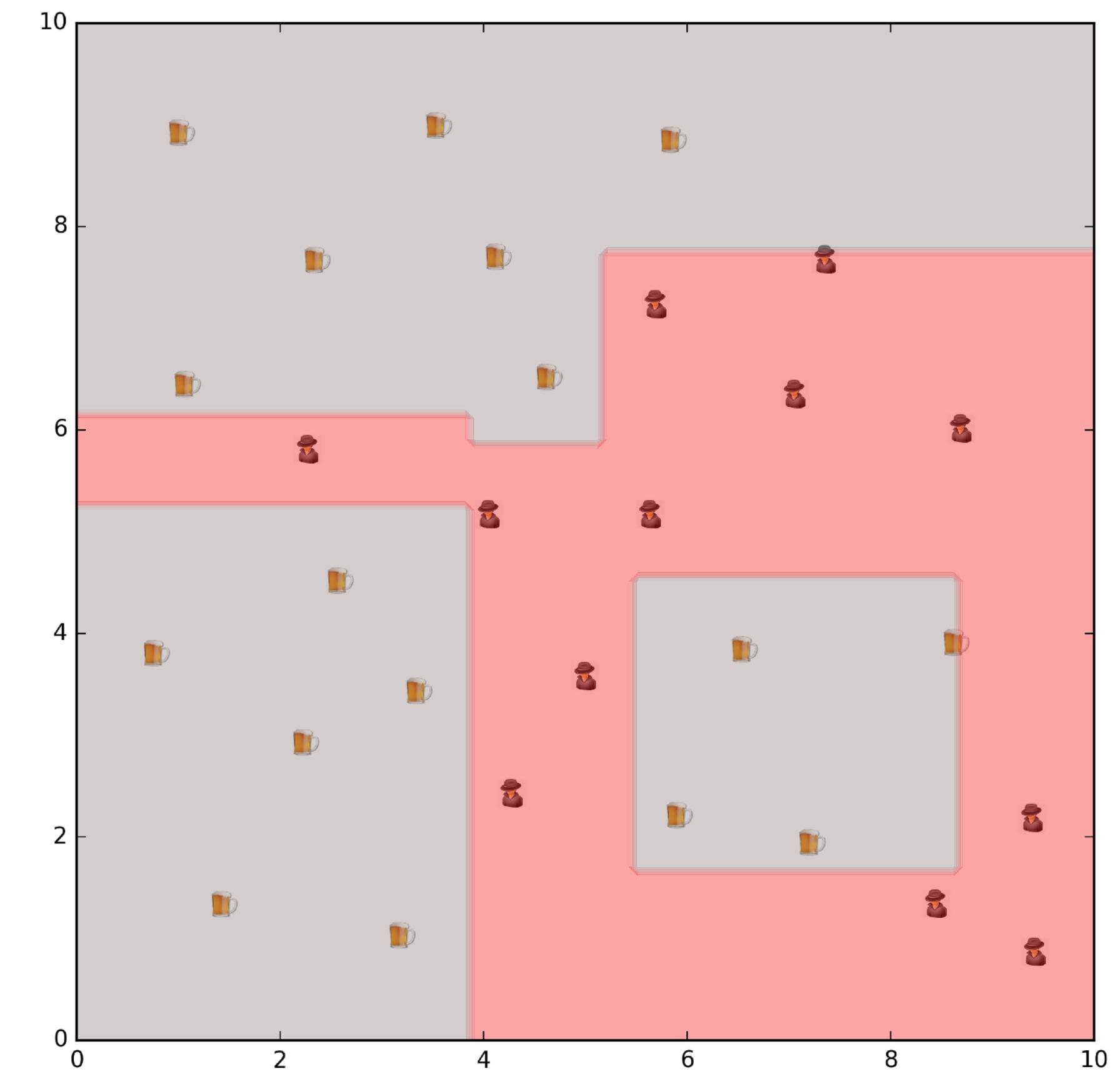
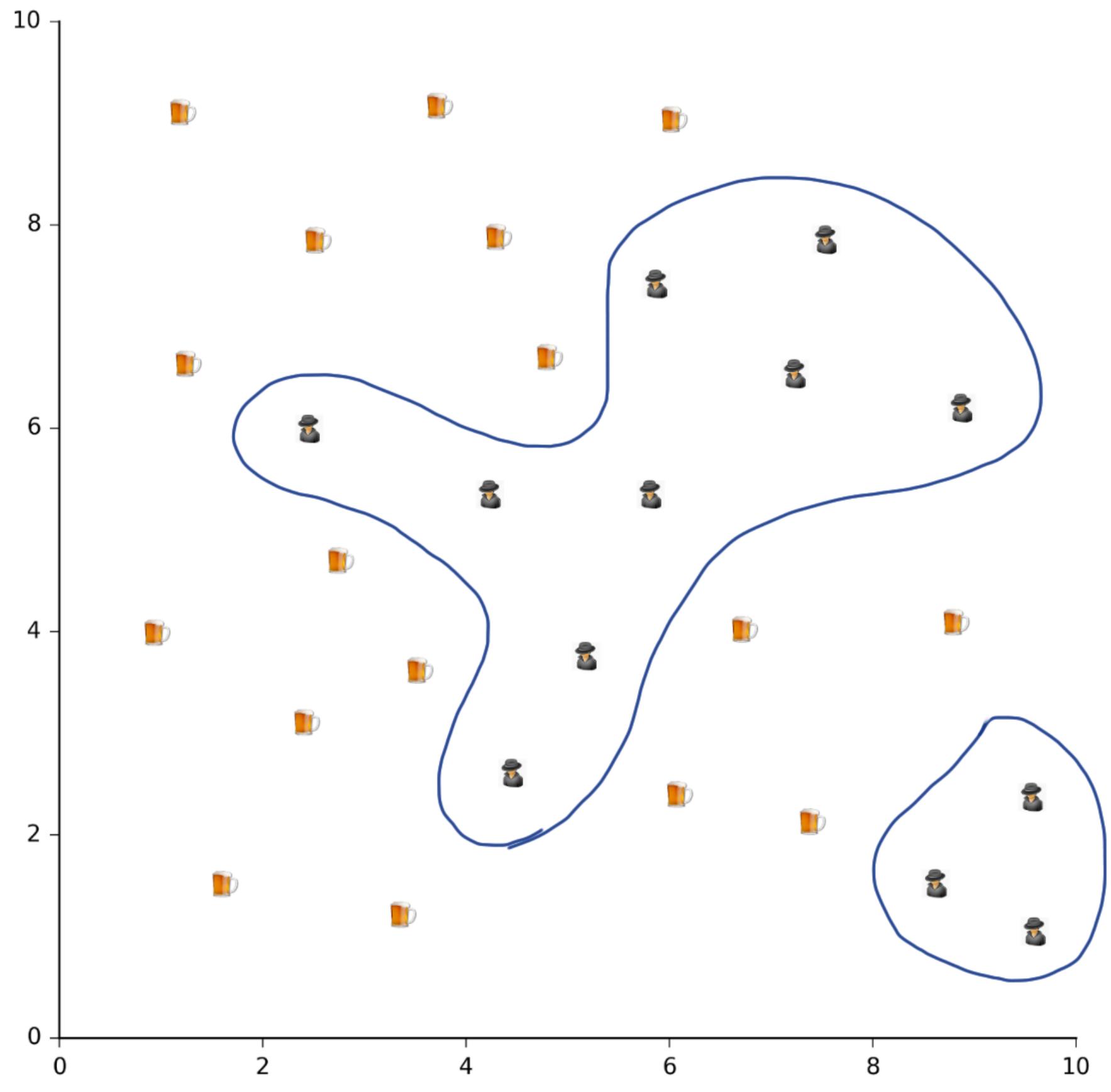
Answer:

1. secrecy threshold 5.27
2. alcohol threshold 4.47
3. secrecy threshold 2.70

# Ultimate last challenge! Who can solve it?

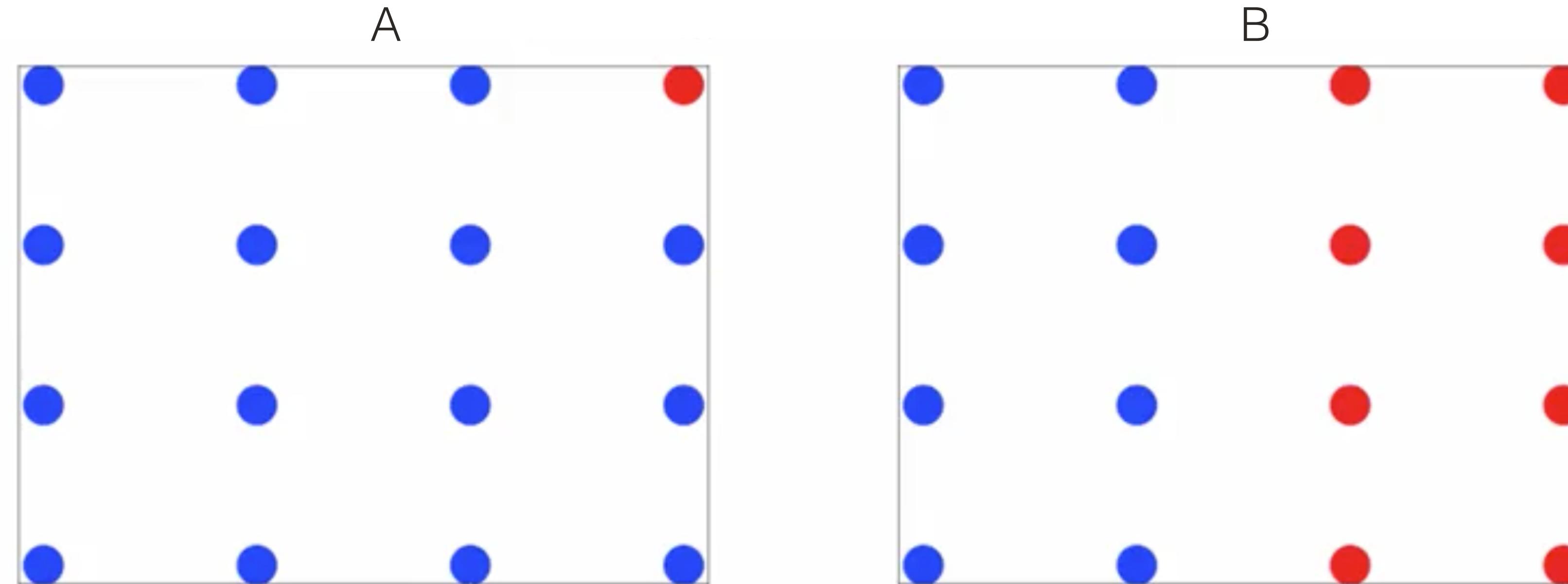


# Such boundaries can be a sign of over-fitting!



# Lastly: How does a Decision Tree exactly decide to split?

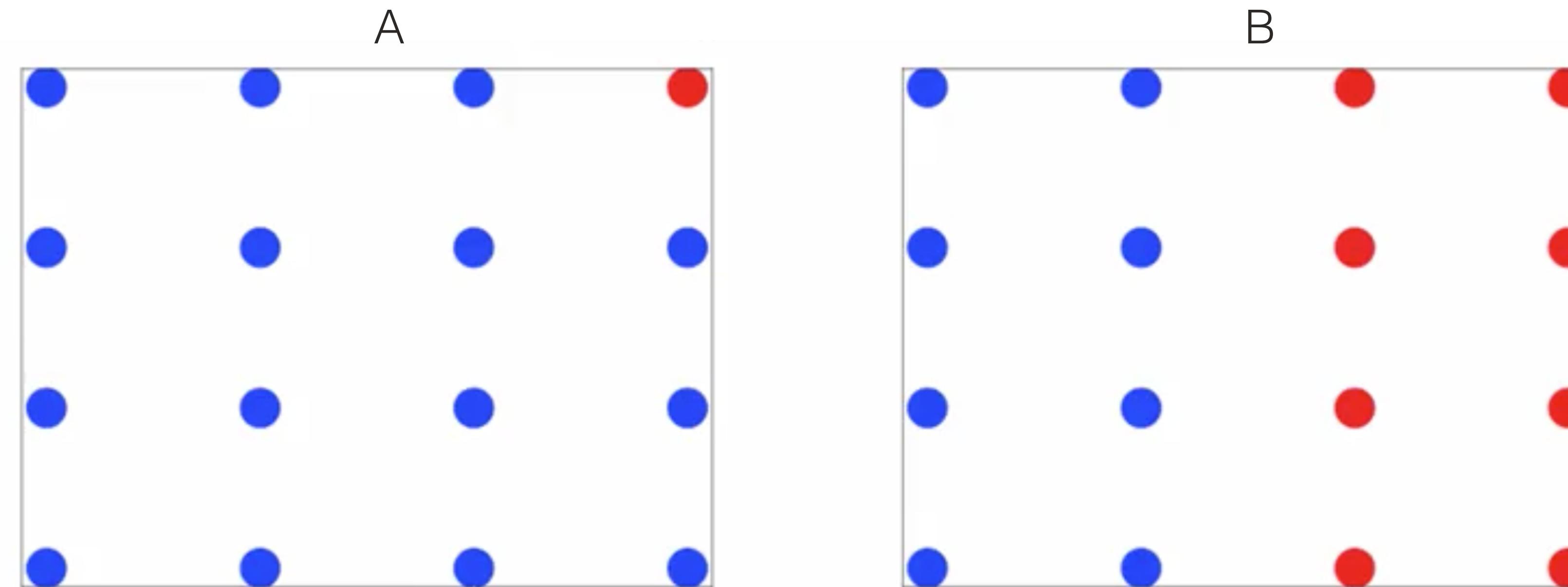
Criterion for best splits: **Impurity**



Quiz: By intuition which example appears to be more pure, that is, separates the two classes better?

# Lastly: How does a Decision Tree exactly decide to split?

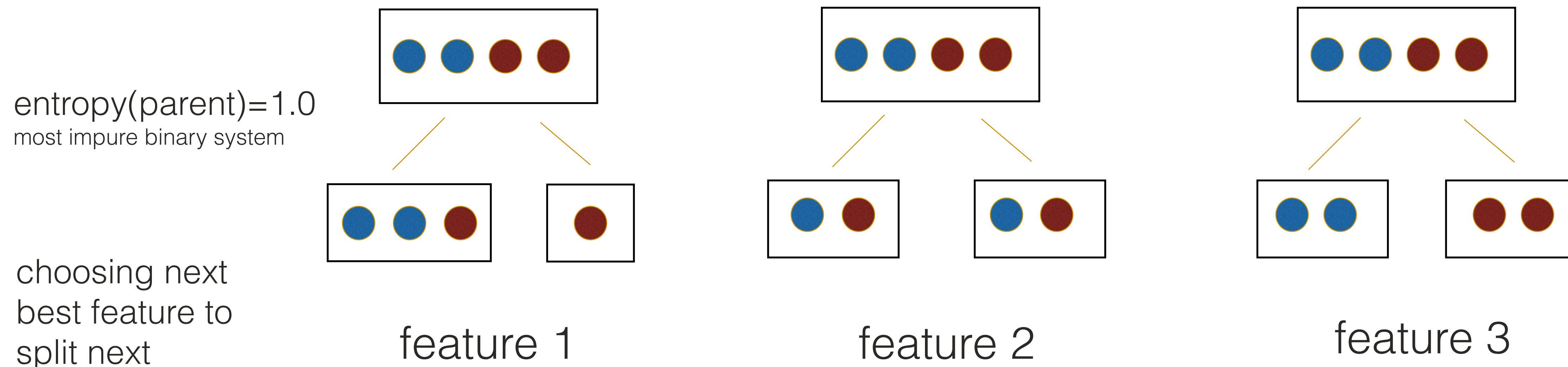
Criterion for best splits: **Information Gain**



Answer: A

# Which feature below gives the best split?

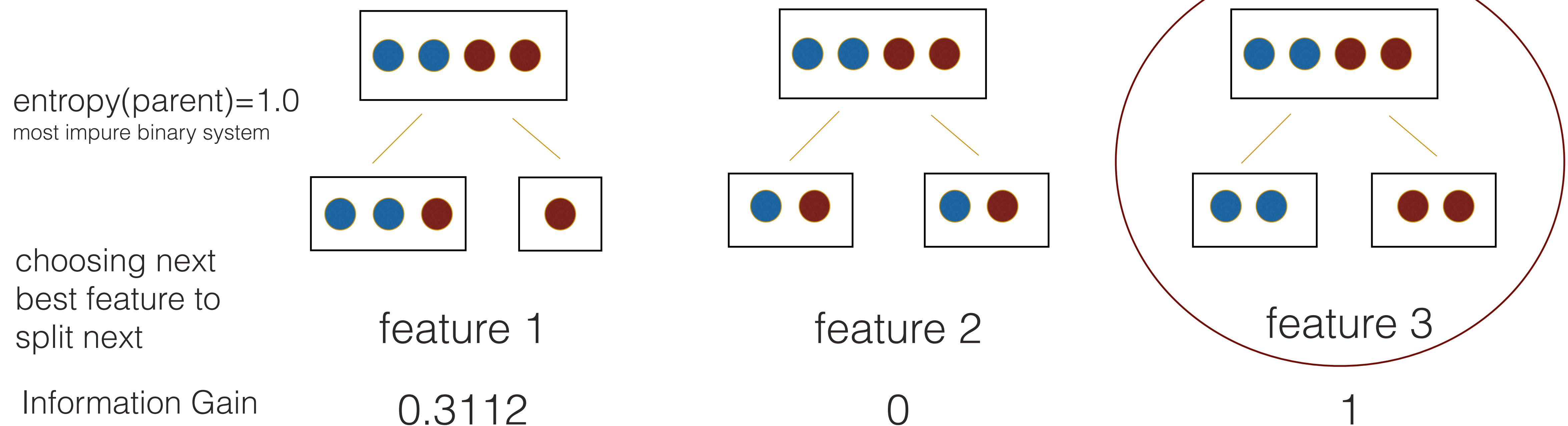
Information Gain =  $\text{entropy}(\text{parent}) - [\text{weighted average}] \text{entropy}(\text{children})$



?

# which feature below gives the best split?

Information Gain =  $\text{entropy}(\text{parent}) - [\text{weighted average}] \text{entropy}(\text{children})$



# Advantages of Decision Trees

- Can be used for regression or classification
- Can be displayed graphically
- Highly interpretable
- Can be specified as a series of rules, and more closely approximate human decision-making than other models
- Prediction is fast
- Features don't need scaling
- Automatically learns feature interactions (they are non-linear models)
- Tend to ignore irrelevant features (especially when there are lots of features)
- Because decision trees are non-linear models they will outperform linear models if the relationship between features and response is highly non-linear

# Disadvantages of Decision Trees

- Performance is (generally) not competitive with the best supervised learning methods
- Can easily overfit the training data (tuning is required)
- Small variations in the data can result in a completely different tree (they are high variance models)
- Recursive binary splitting makes "locally optimal" decisions that may not result in a globally optimal tree
- Don't tend to work well if the classes are highly unbalanced
- Don't tend to work well with very small datasets

# K-Nearest Neighbors Classifier

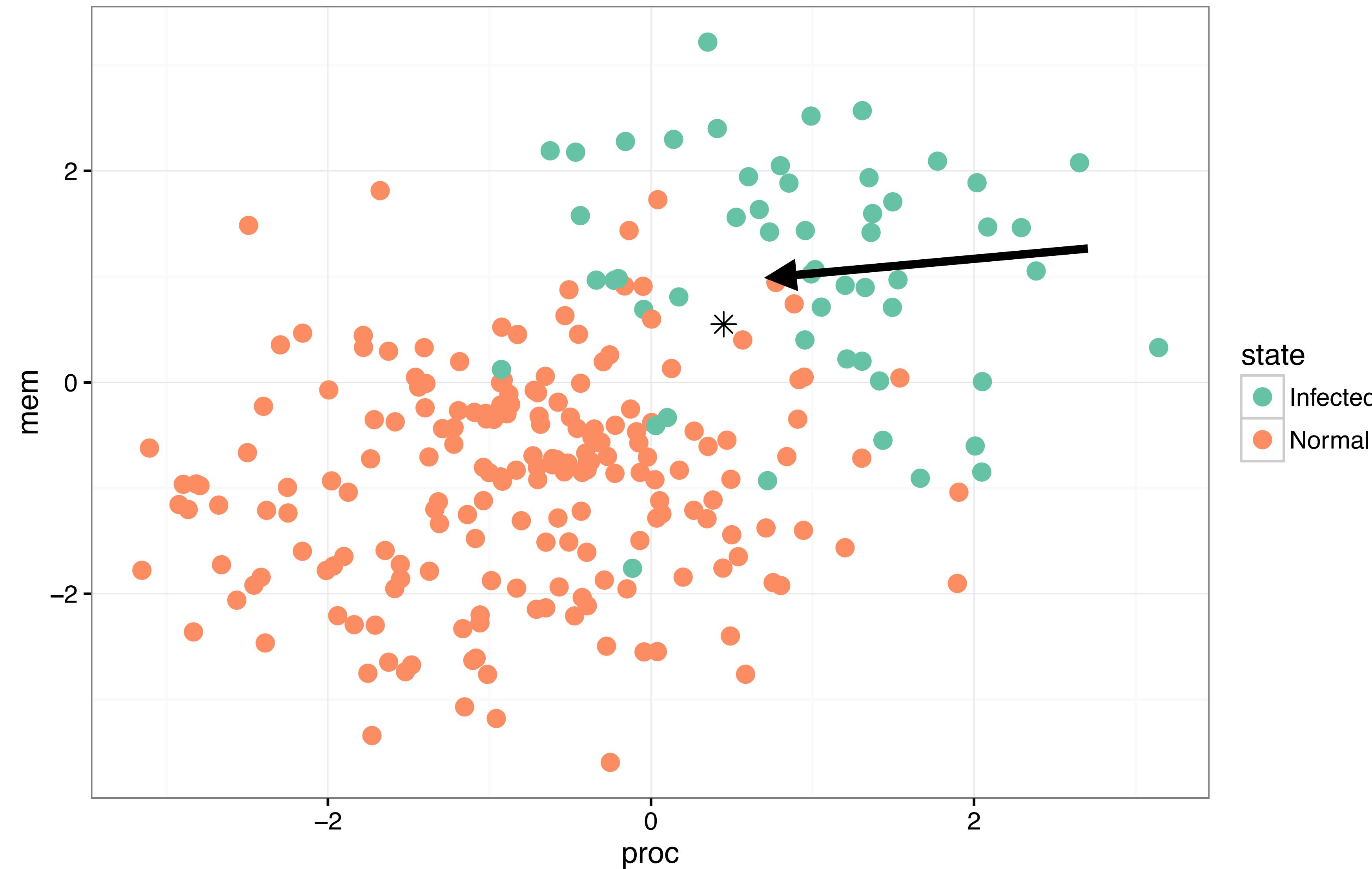


# K-nearest neighbors

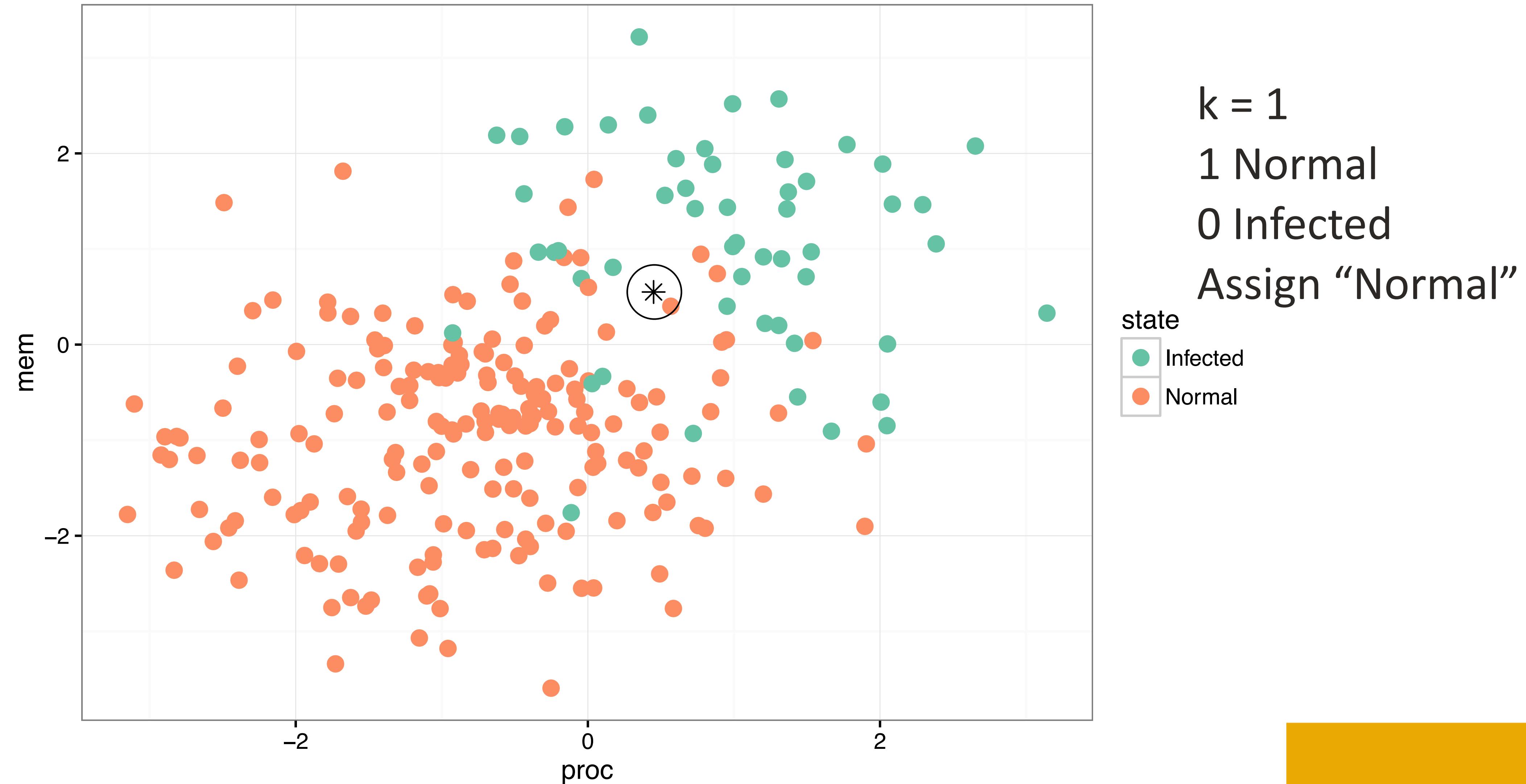
Define K, and for every unknown observation:

1. Find k-nearest neighbors (by *distance*)
2. Assign class based on dominate class
3. Optional weight by distance

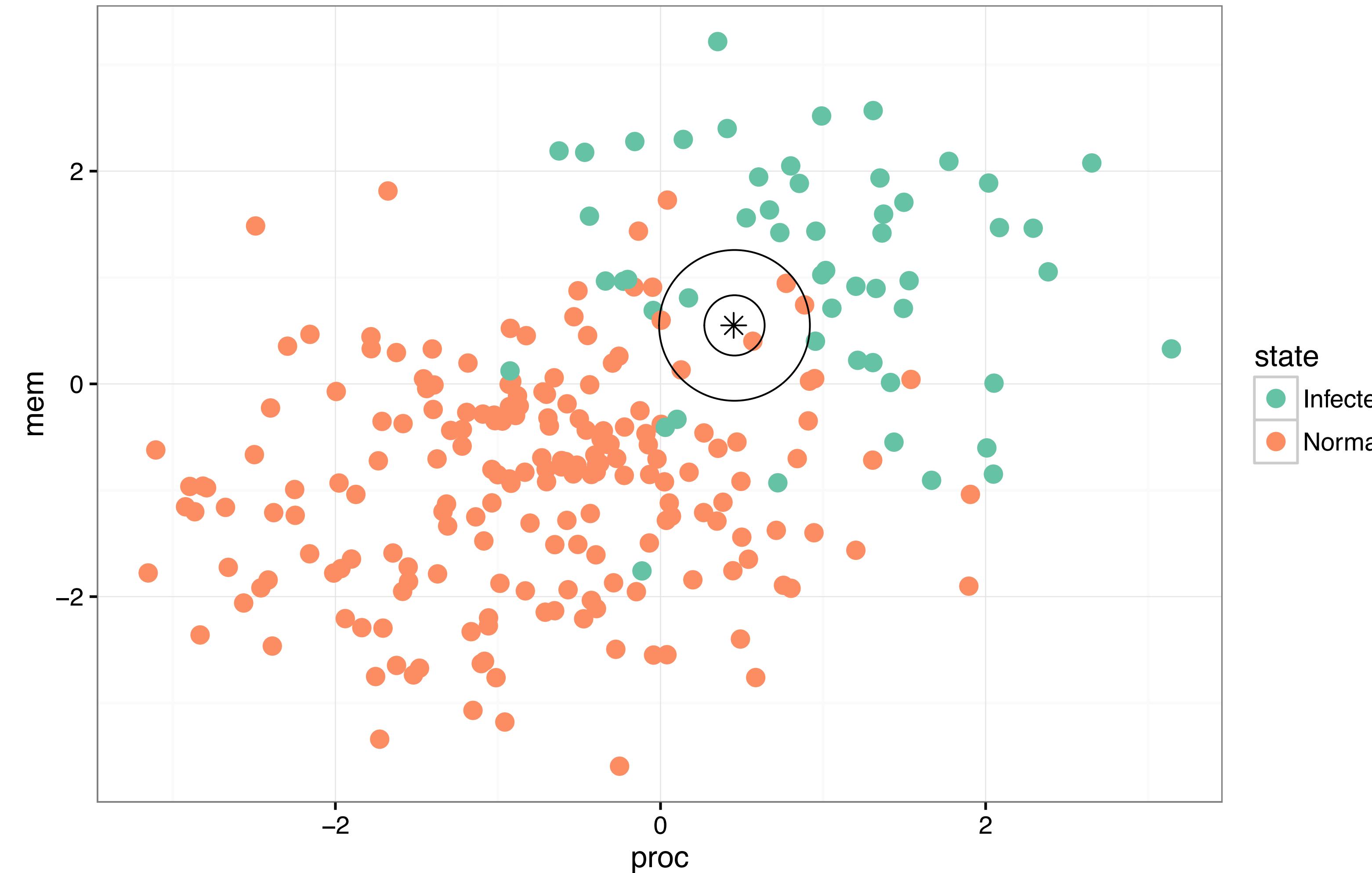
# K-nearest neighbors



# K-nearest neighbors



# K-nearest neighbors



$k = 5$   
4 Normal  
1 Infected  
Assign "Normal"

# K-nearest neighbors

Further reading: <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/>

# K-NN Classifiers

## Advantages of K-NN Classifiers

- Robust to noisy, non-linear training data
- Flexible to feature / distance choices
- Works with large datasets
- Naturally handles multi-class cases

## Disadvantages of K-NN Classifiers:

- Need to determine the value of K
- Training is fast but predictions are slow. Indexing can help
- Must have meaningful distance function

# Ensembles



# Ensembles

**Ensemble learning** is the process of combining several predictive models in order to produce a combined model that is more accurate than any individual model.

- **Regression:** take the average of the predictions
- **Classification:** take a vote and use the most common prediction, or take the average of the predicted probabilities

# Ensembles

For ensembling to work well, the models must have the following characteristics:

- **Accurate:** they outperform random guessing
- **Independent:** their predictions are generated using different processes

**The big idea:** If you have a collection of individually imperfect (and independent) models, the "one-off" mistakes made by each model are probably not going to be made by the rest of the models, and thus the mistakes will be discarded when averaging the models.

**Note:** As you add more models to the voting process, the probability of error decreases, which is known as [Condorcet's Jury Theorem](#).

# Bagging

1. Grow  $n$  trees using  $n$  bootstrap samples from the training data.
2. Train each tree on its bootstrap sample and make predictions.
3. Combine the predictions:
  - Average the predictions for **regression trees**
  - Take a majority vote for **classification trees**

# Random Forest

Random Forests are a **slight variation of bagged trees** that have even better performance:

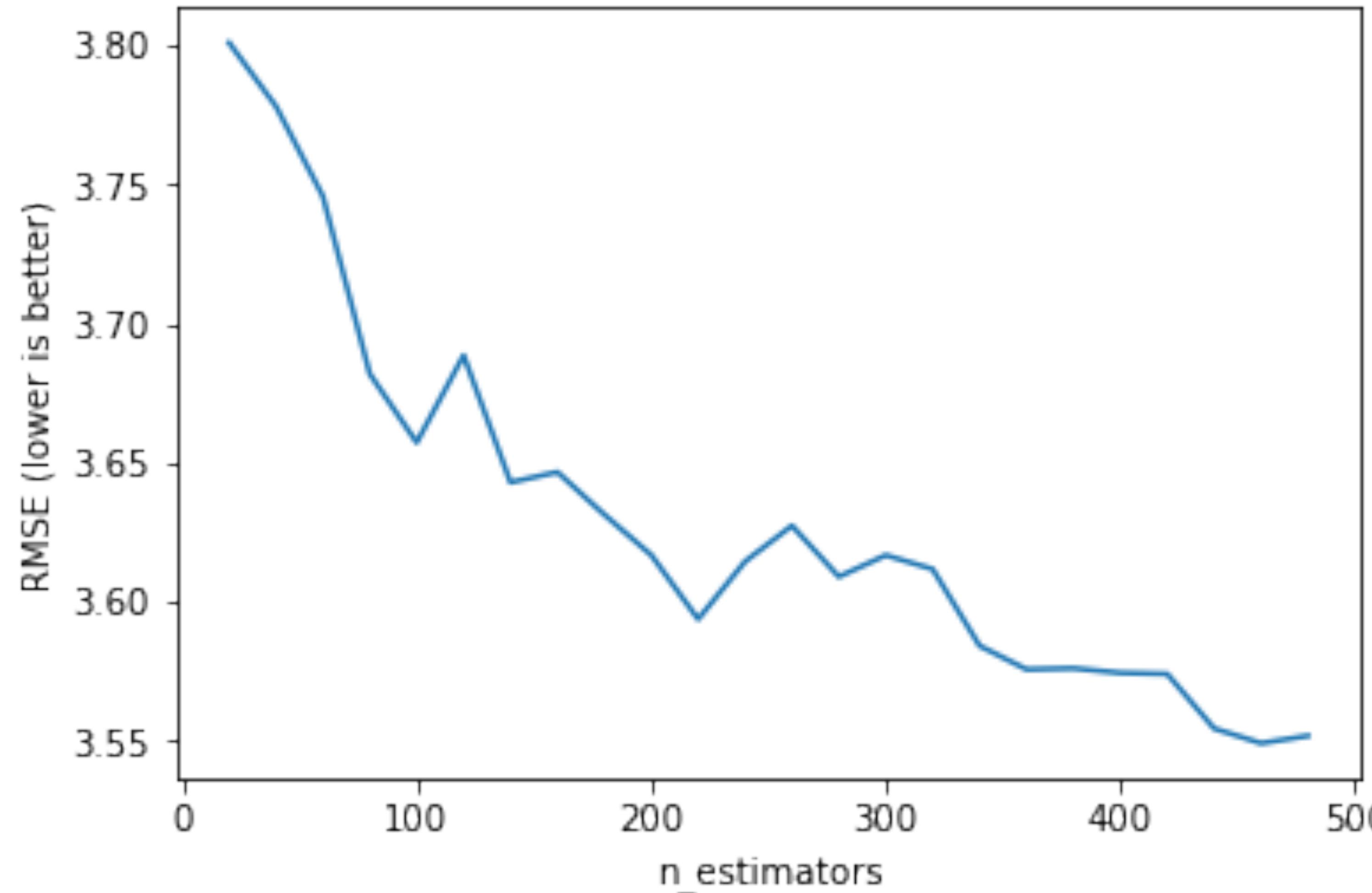
- Just like bagging, we create an ensemble of decision trees using bootstrapped samples of the training set.
- However, when building each tree, each time a split is considered, a **random sample of  $m$  features** is chosen as split candidates from the **full set of  $p$  features**. The split is only allowed to use **one of those  $m$  features**.
  - A new random sample of features is chosen for **every single tree at every single split**.
  - For **classification**,  $m$  is typically chosen to be the square root of  $p$  (the total number of features).
  - For **regression**,  $m$  is typically chosen to be somewhere between  $p/3$  and  $p$ .

# Tuning a Random Forest

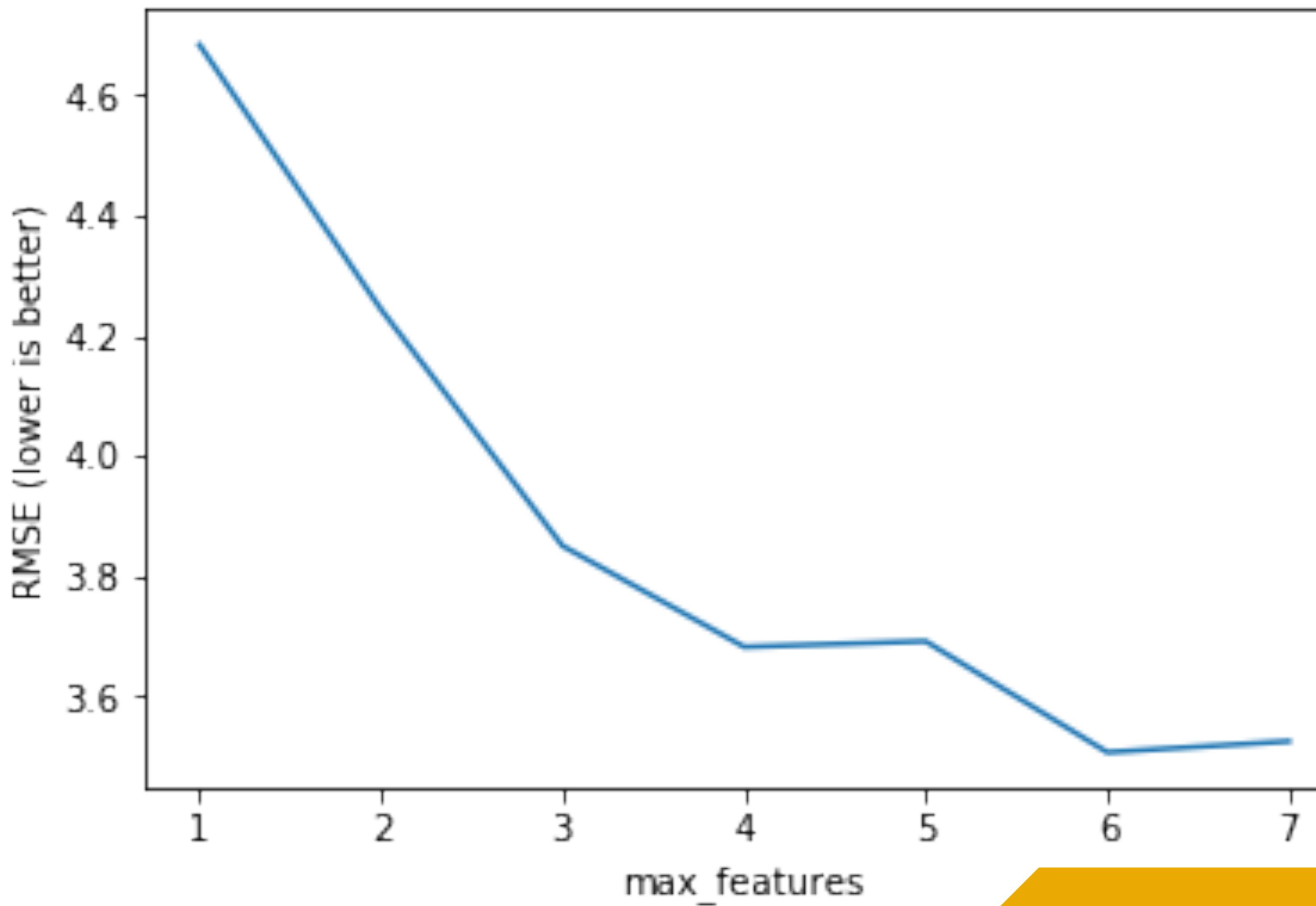
2 important parameters that should be tuned when creating a random forest model are:

- The number of trees to grow (called **n\_estimators** in scikit-learn)
- The number of features that should be considered at each split (called **max\_features** in scikit-learn)

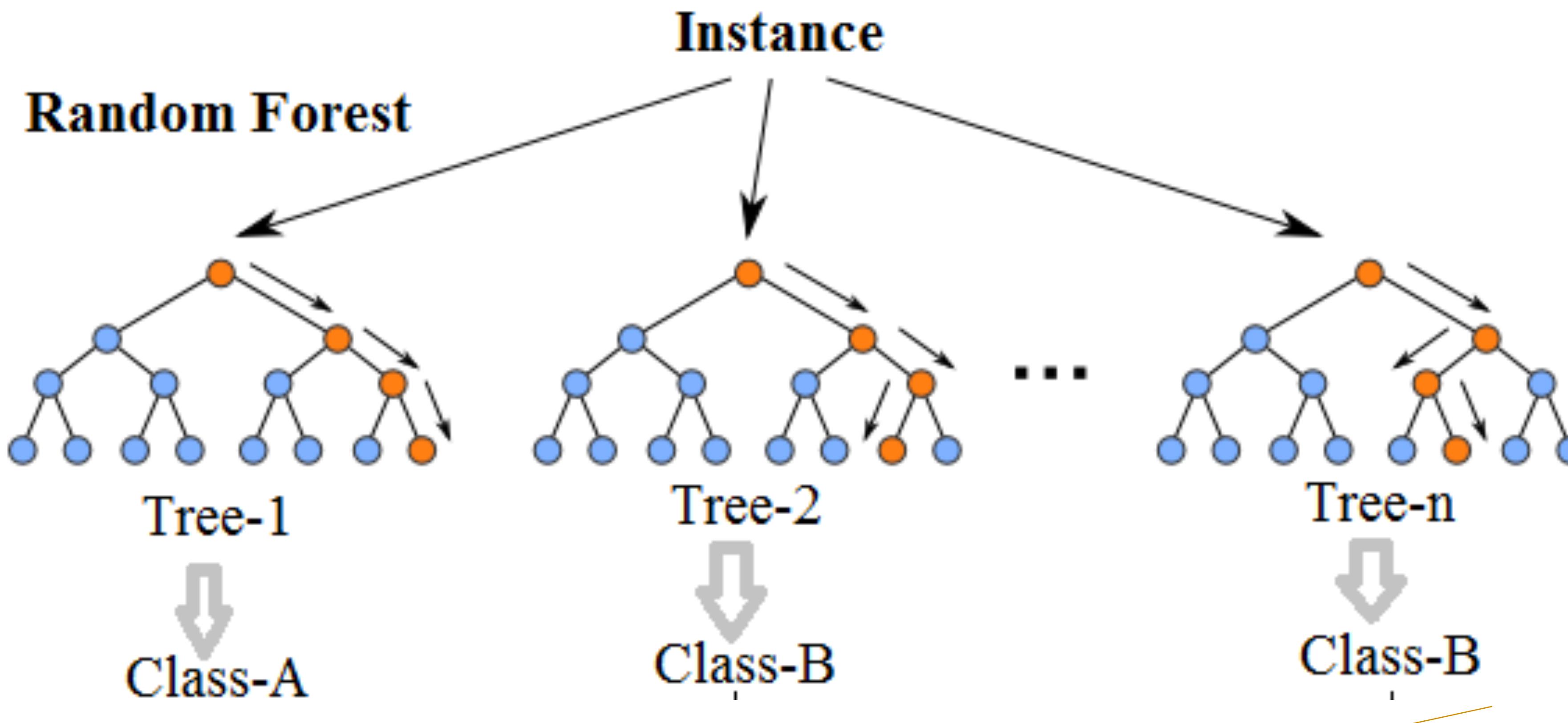
# Tuning a Random Forest



# Tuning a Random Forest



# Tuning a Random Forest



Class A:  $p\%$  Class B:  $1-p\%$

Majority Win

GTK Cyber

# Random Forests

## Advantages of Random Forests:

- Performance is competitive with the best supervised learning methods
- Provides a more reliable estimate of feature importance
- Allows you to estimate out-of-sample error without using train/test split or cross-validation

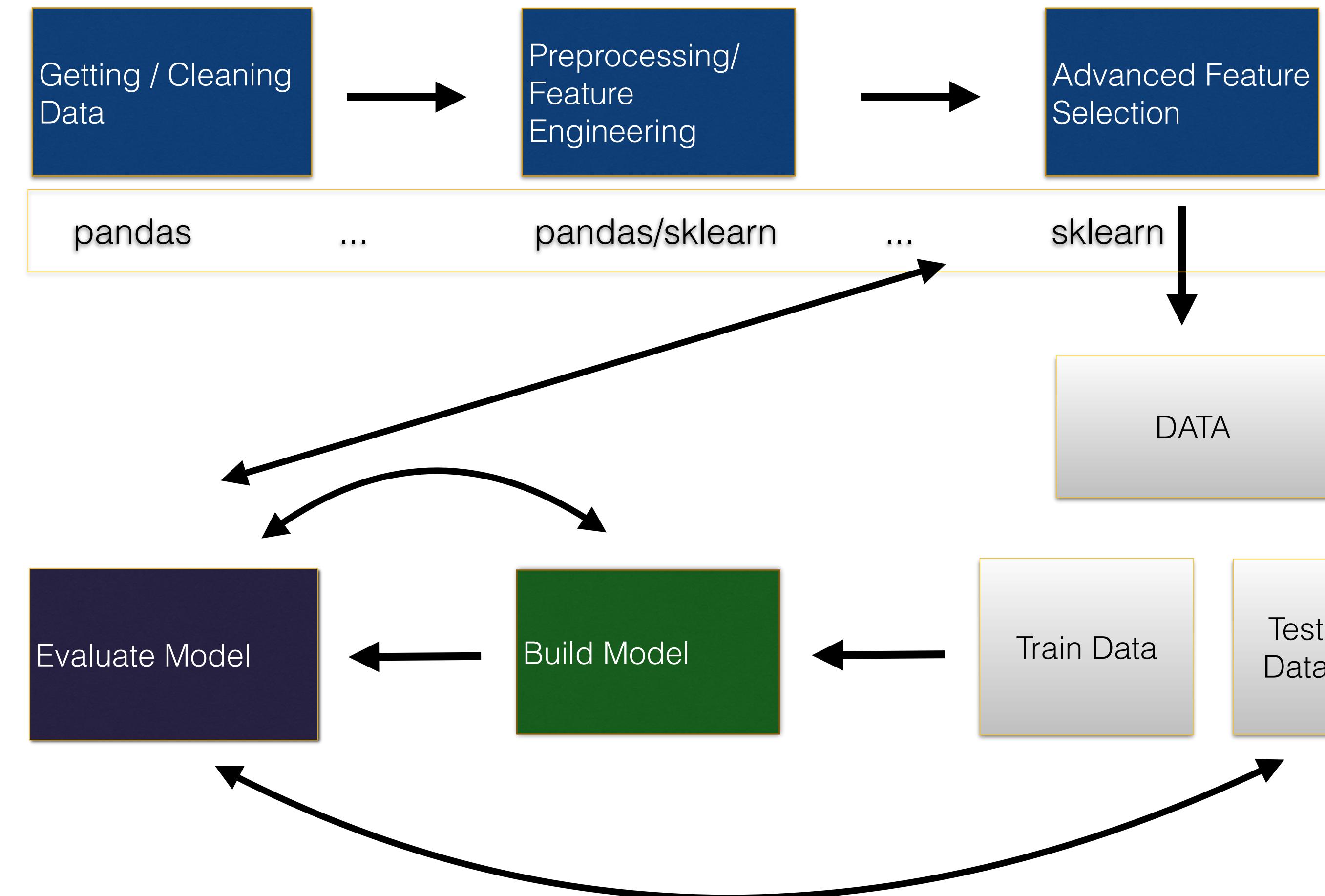
## Disadvantages of Random Forests:

- Less interpretable
- Slower to train
- Slower to predict

# **Supervised Machine Learning**

Implementation

# Machine Learning Process



# Scikit-Learn Process

The basic process for any supervised learning is the same for any supervised learning in Python.

1. Create the Classifier or Regressor object.

```
clf = RandomForestClassifier()
```

2. Call the `.fit()` method using the **training feature matrix (X)** and the **training target vector (y)**.

```
clf.fit( training_features, training_target )
```

3. Call the `.predict()` method using a feature matrix

```
#Returns vector of predictions
```

```
clf.predict( testing_features )
```

# Train-Predict in sklearn

```
## Support Vector Machine Classification  
  
clf = svm.SVC()  
clf = clf.fit(X, target)  
target_pred = clf.predict(X)
```

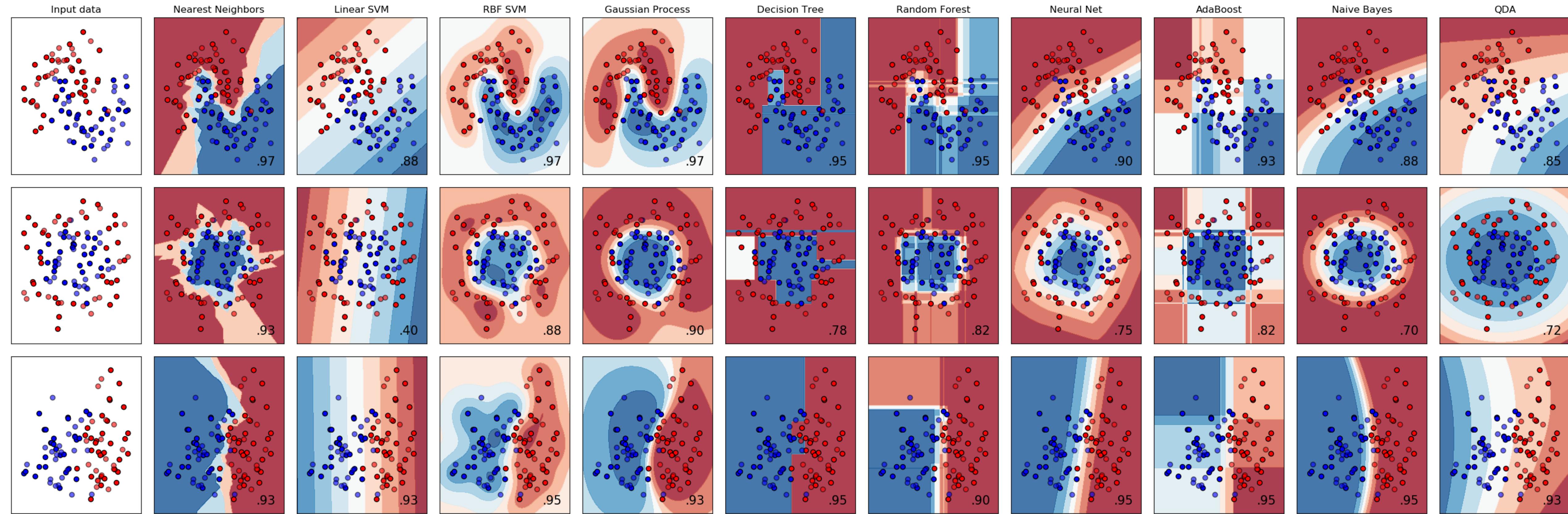
# Train-Predict in sklearn

```
## Decision Tree Classification  
  
clf = tree.DecisionTreeClassifier()  
clf = clf.fit(X, target)  
target_pred = clf.predict(X)
```

# Classification Algorithms in Scikit-Learn

- Generalized Linear Models
- Kernel Ridge
- Support Vector Machines
- Nearest Neighbors
- Gaussian Processes
- Naive Bayes
- Trees
- Neural Networks
- AdaBoost
- Gradient Tree Boosting
- Ensemble methods

# Classification Algorithms in Scikit-Learn



# Python Sklearn Overview

<http://scikit-learn.org/stable/>



## Classification

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ...

[— Examples](#)

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, ridge regression, Lasso, ...

[— Examples](#)

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, ...

[— Examples](#)

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-negative matrix factorization.

[— Examples](#)

## Model selection

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning

**Modules:** grid search, cross validation, metrics.

[— Examples](#)

## Preprocessing

Feature extraction and normalization.

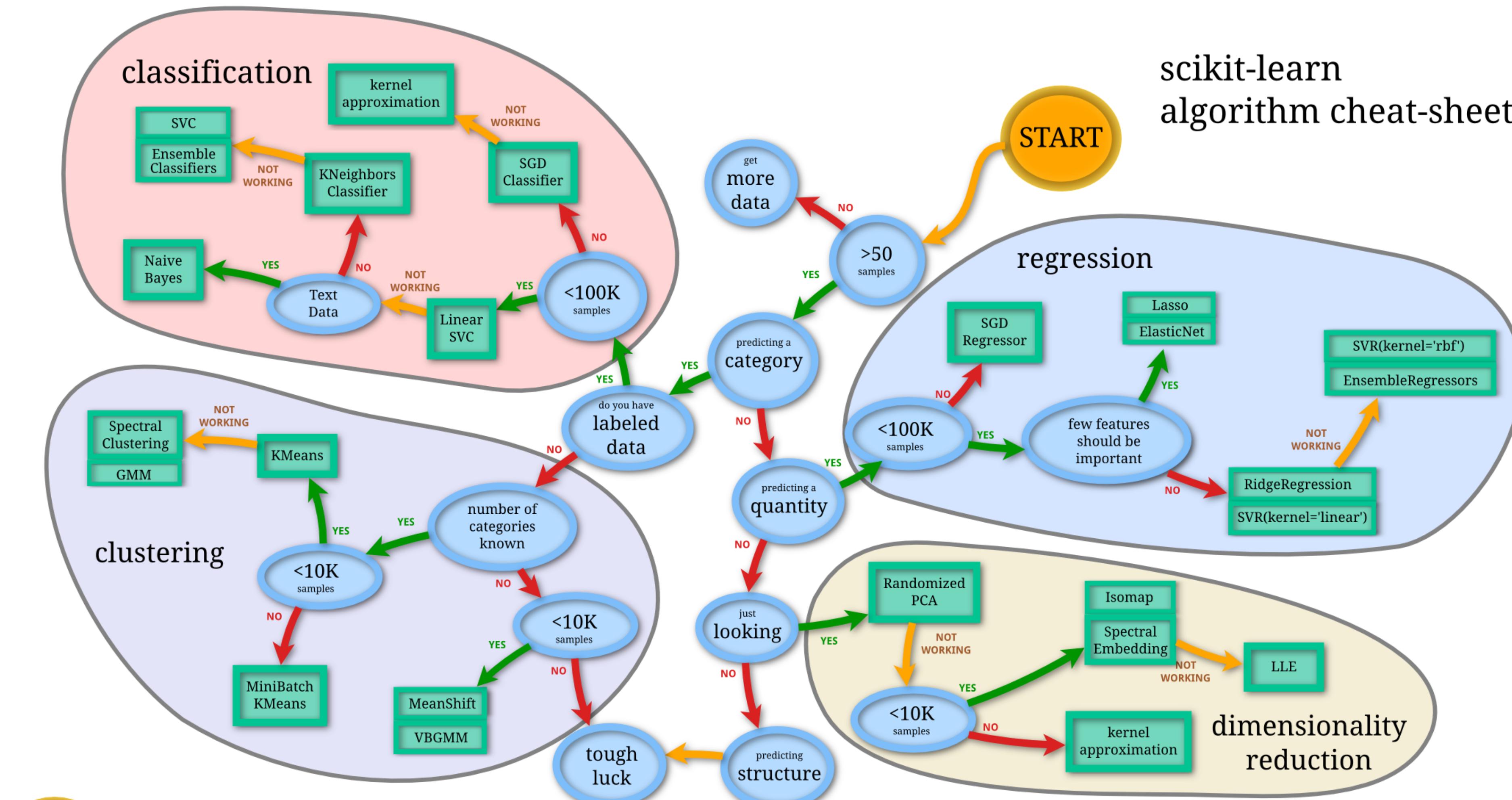
**Application:** Transforming input data such as text for use with machine learning algorithms.

**Modules:** preprocessing, feature extraction.

[— Examples](#)

# Sklearn Cheat-Sheet

scikit-learn  
algorithm cheat-sheet



# Sklearn Classifiers Navigation

## sklearn.tree.DecisionTreeClassifier

```
class sklearn.tree. DecisionTreeClassifier (criterion='gini', splitter='best', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None,
max_leaf_nodes=None, min_impurity_split=1e-07, class_weight=None, presort=False) [source]
```

A decision tree classifier.

Read more in the [User Guide](#).

**Parameters:** **criterion** : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

**splitter** : string, optional (default="best")

The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

**max\_features** : int, float, string or None, optional (default=None)

The number of features to consider when looking for the best split:

Check parameters for each classifier!

# Sklearn Classifiers Navigation

**fit (X, y, sample\_weight=None, check\_input=True, X\_idx\_sorted=None)** [source]

Build a decision tree classifier from the training set (X, y).

**Parameters:** **X** : array-like or sparse matrix, shape = [n\_samples, n\_features]

The training input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csc_matrix`.

**y** : array-like, shape = [n\_samples] or [n\_samples, n\_outputs]

The target values (class labels) as integers or strings.

**sample\_weight** : array-like, shape = [n\_samples] or None

Sample weights. If None, then samples are equally weighted. Splits that would create child nodes with net zero or negative weight are ignored while searching for a split in each node. Splits are also ignored if they would result in any single class carrying a negative weight in either child node.

**predict (X, check\_input=True)** [source]

Predict class or regression value for X.

For a classification model, the predicted class for each sample in X is returned. For a regression model, the predicted value based on X is returned.

**Parameters:** **X** : array-like or sparse matrix of shape = [n\_samples, n\_features]

The input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csr_matrix`.

**check\_input** : boolean, (default=True)

Allow to bypass several input checking. Don't use this parameter unless you know what you do.

**Returns:** **y** : array of shape = [n\_samples] or [n\_samples, n\_outputs]

The predicted classes, or the predict values.

Check input and output dimensions of fit and predict methods for feature matrix X and target vector y!

# Sklearn Classifiers Navigation

**Attributes:** `classes_` : array of shape = [n\_classes] or a list of such arrays  
The classes labels (single output problem), or a list of arrays of class labels (multi-output problem).

`feature_importances_` : array of shape = [n\_features]  
The feature importances. The higher, the more important the feature. The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance [R245].

`max_features_` : int,  
The inferred value of max\_features.

`n_classes_` : int or list  
The number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).

`n_features_` : int  
The number of features when `fit` is performed.

After calling `fit` method you can retrieve certain attributes of your `clf` object!

# Sklearn Classifiers Navigation

**fit (X, y=None)** [source]

Compute k-means clustering.

**Parameters:** X : array-like or sparse matrix, shape=(n\_samples, n\_features)

Training instances to cluster.

**fit\_predict (X, y=None)** [source]

Compute cluster centers and predict cluster index for each sample.

Convenience method; equivalent to calling fit(X) followed by predict(X).

**fit\_transform (X, y=None)** [source]

Compute clustering and transform X to cluster-distance space.

Equivalent to fit(X).transform(X), but more efficiently implemented.

**transform (X, y=None)** [source]

Transform X to a cluster-distance space.

In the new space, each dimension is the distance to the cluster centers. Note that even if X is sparse, the array returned by transform will typically be dense.

**Parameters:** X : {array-like, sparse matrix}, shape = [n\_samples, n\_features]

New data to transform.

**Returns:** X\_new : array, shape [n\_samples, k]

X transformed in the new space.

Clustering and dimensionality reduction methods like PCA have more fit and transform rather than predict and often combine fit\_transform in one method!

"Why should I trust you?"  
Explaining the predictions of any classifier

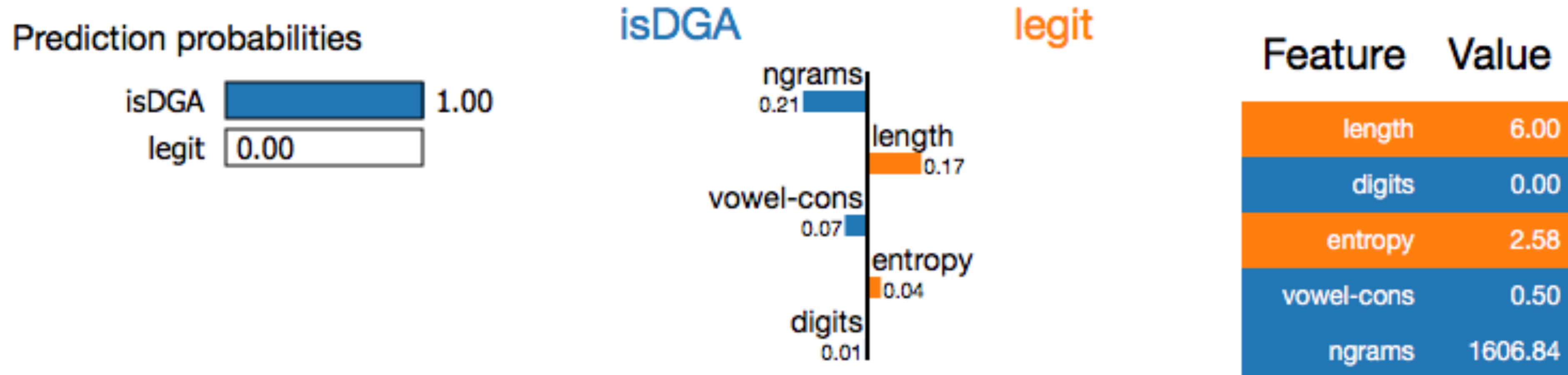
```

import lime
import lime.lime_tabular
explainer = lime.lime_tabular.LimeTabularExplainer(<training_data>,
    feature_names=<feature names>,
    class_names=<class names>,
    discretize_continuous=False)

exp = explainer.explain_instance( <test_row>,
    <model>.predict_proba,
    num_features=5,
    top_labels=1)

exp.show_in_notebook(show_table=True, show_all=False)

```

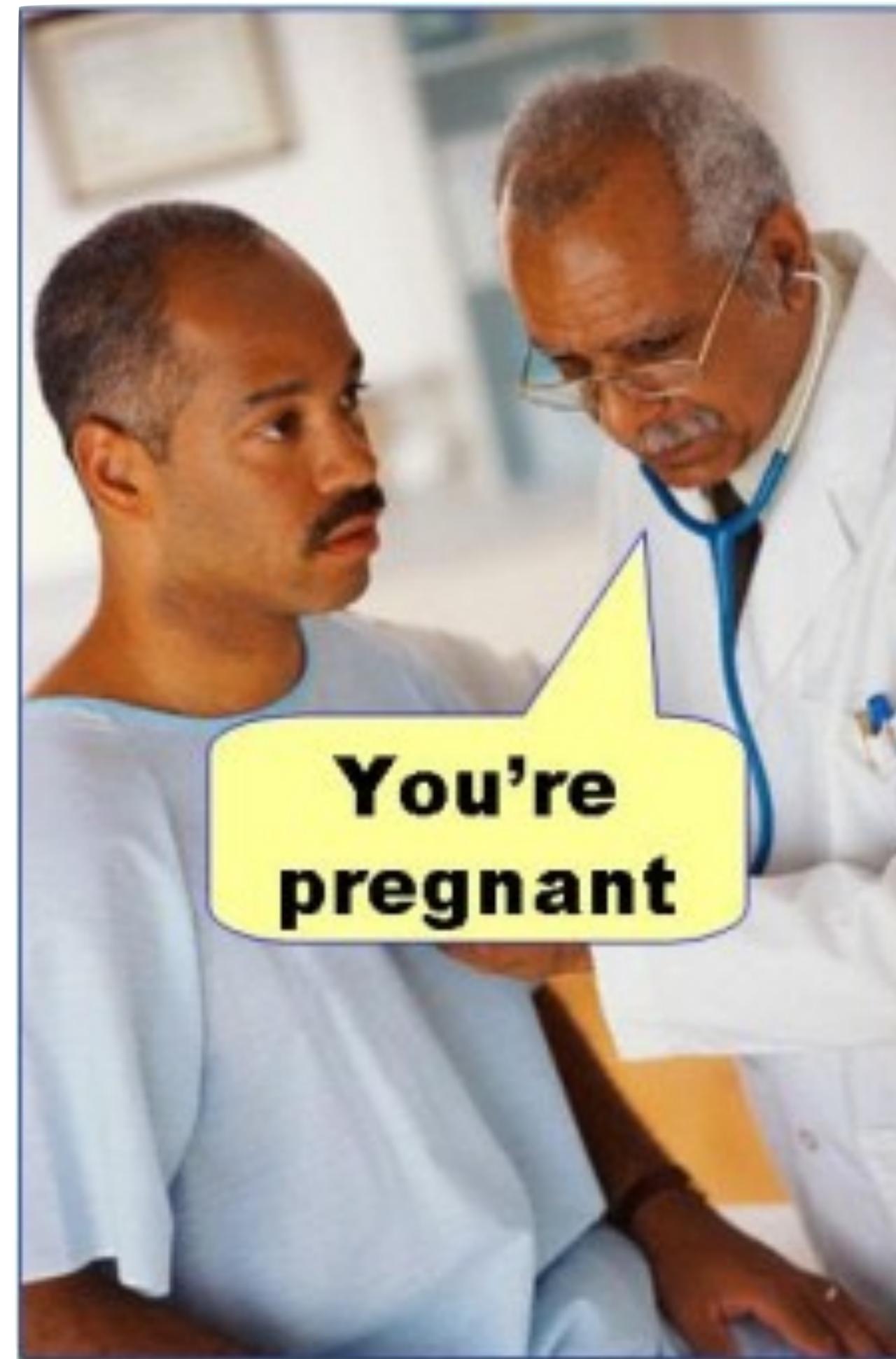


# **Supervised Machine Learning**

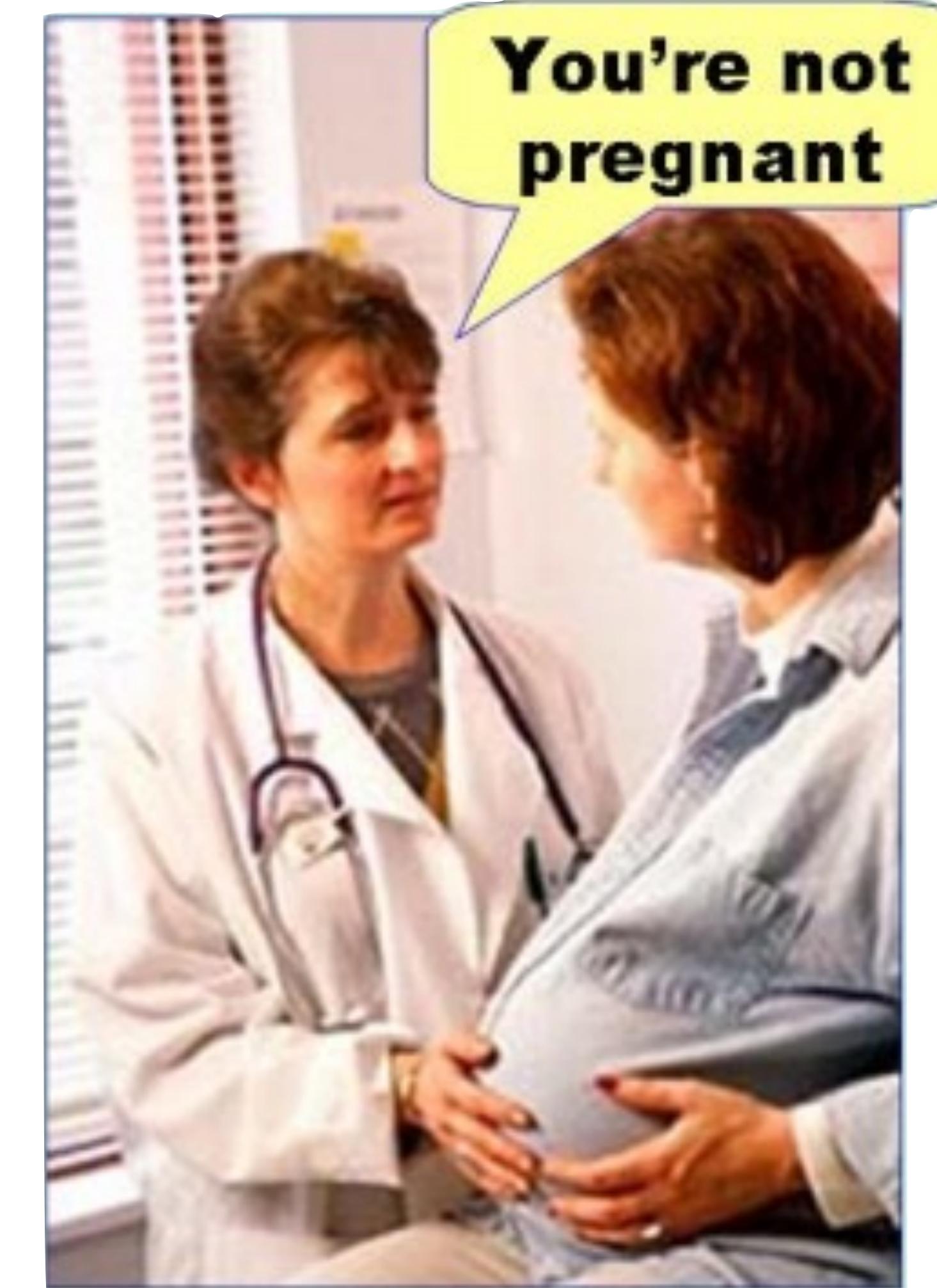
Metrics and cross-validation

# **Performance Metrics for Classifiers**

## Type I Error (False Positive)



## Type II Error (False Negative)



# Confusion Matrix

```
target = [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0]
print(confusion_matrix(target, target_pred))
```

True target	0	[ [ 7 2 ] ]
	1	[ 3 4 ] ]
	0	1

Predicted target

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

# Accuracy

```
accuracy = accuracy_score(target, target_pred)
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

Accuracy = (TP + TN)/ N  
Quiz: Calculate by hand!

True target	0	[ [ 7    2 ]
	1	[ 3    4 ] ]
	0    1	

Predicted target

# Accuracy

```
accuracy = accuracy_score(target, target_pred)
```

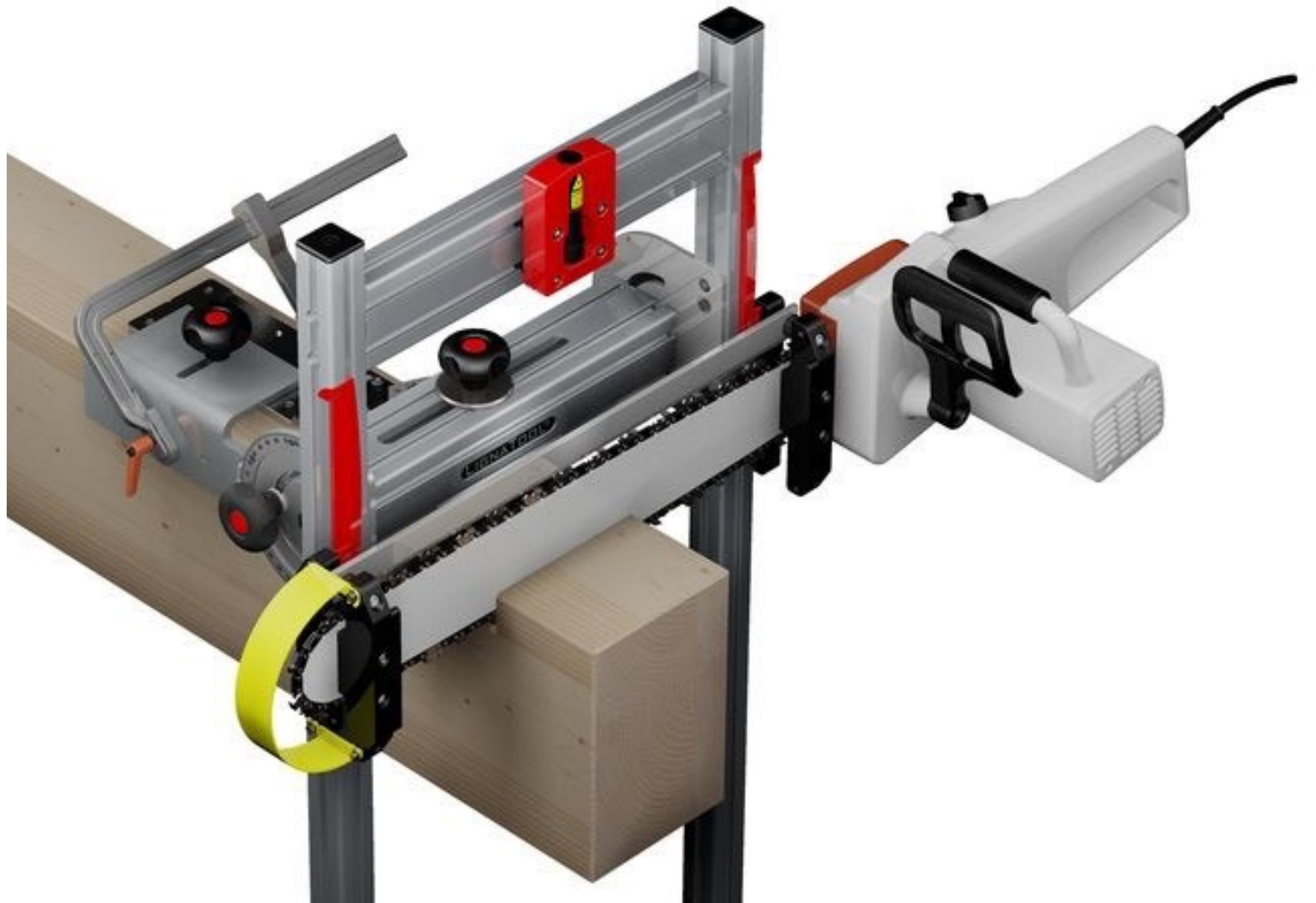
True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

$$\begin{aligned}\text{Accuracy} &= (\text{TP} + \text{TN}) / N \\ &= (7+4) / 16 \\ &= 0.6875\end{aligned}$$

True target	0	[ [ 7 2 ] ]
	1	[ 3 4 ] ]
	0 1	

Predicted target

# Precision



Column-wise!



GTK Cyber

# Precision

```
precision_class = precision_score(target, target_pred, average = None)
precision_avg = precision_score(target, target_pred, average = 'binary')
```



$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Quiz: Calculate precision by hand for both classes!

True target	0	[ [ 7 2 ] ]
	1	[ 3 4 ] ]
	0 1	

Predicted target



# Precision

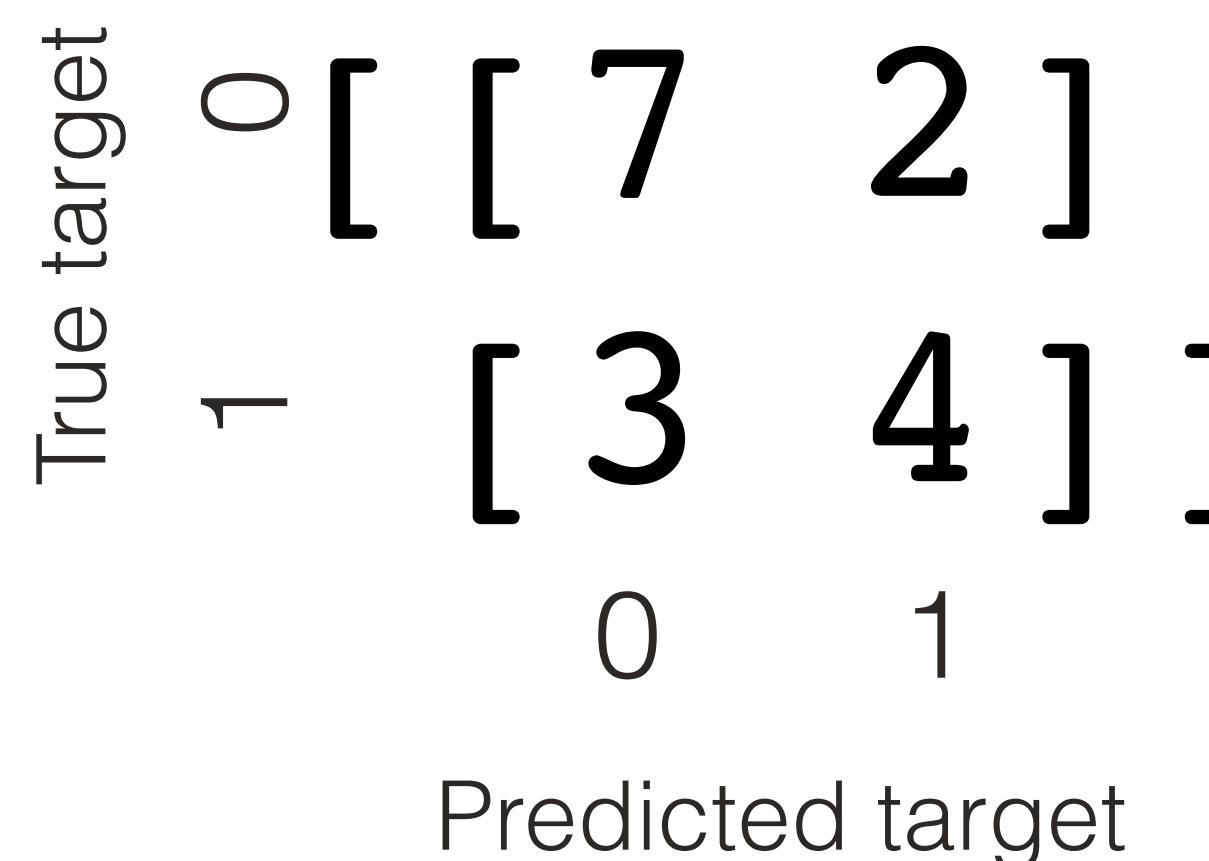
```
precision_class = precision_score(target, target_pred, average = None)
precision_avg = precision_score(target, target_pred, average = 'binary' )
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Prec\_Class0} = 7 / (7 + 3) = 0.7$$

$$\text{Prec\_Class1} = 4 / (4 + 2) = 0.666$$



**Accurate**  
**Precise**



**Not Accurate**  
**Precise**



**Accurate**  
**Not Precise**



**Not Accurate**  
**Not Precise**





Row-wise!



GTK Cyber

# Recall

```
recall_class = recall_score(target, target_pred, average = None)  
recall_avg = recall_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

Recall = TP / (TP + FN)  
Quiz: Calculate recall by hand for both classes!

True target	0	[ [ 7 2 ]
	1	[ 3 4 ] ]
	0	1

Predicted target



# Recall

```
recall_class = recall_score(target, target_pred, average = None)  
recall_avg = recall_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

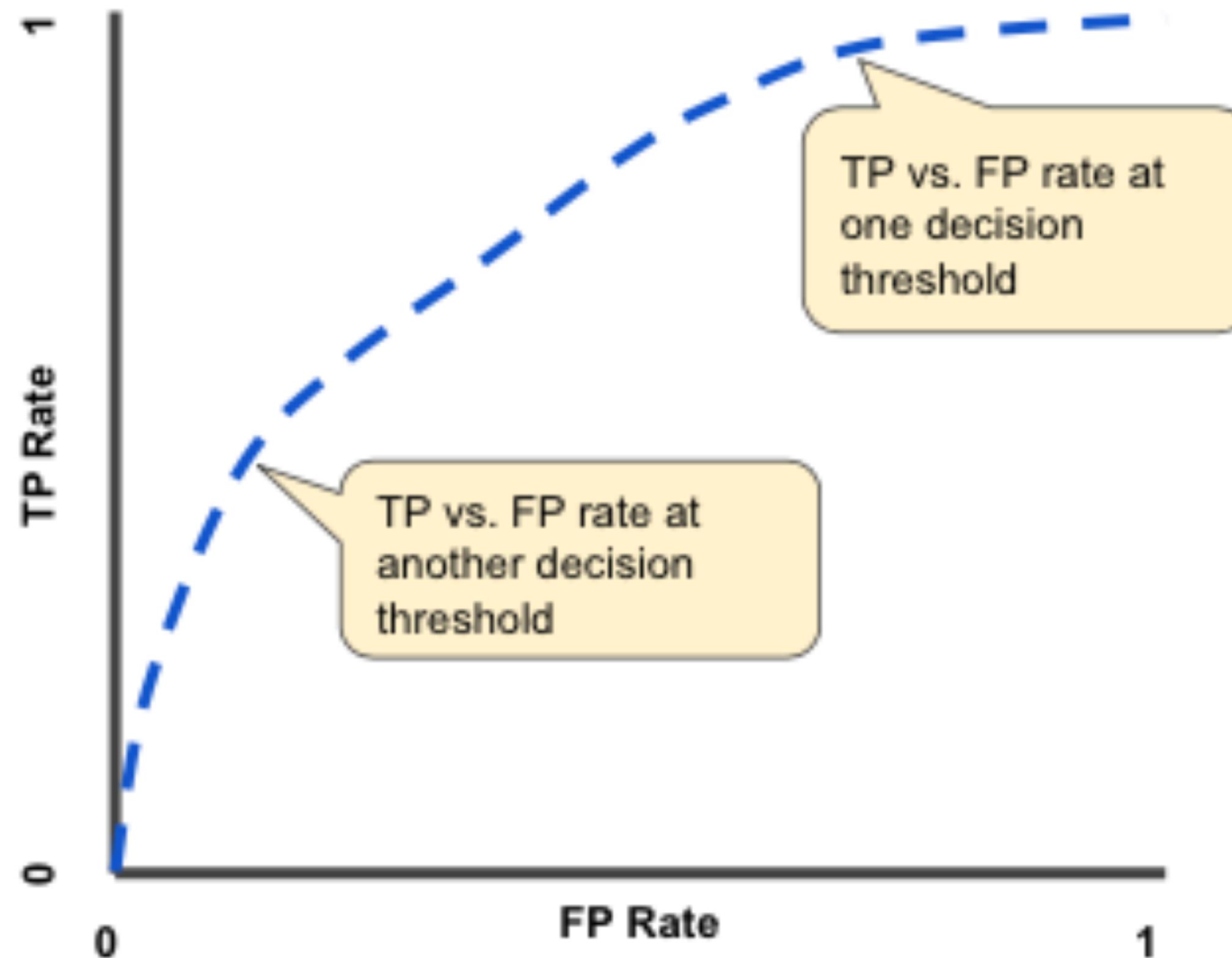
True target	0	[ [ 7    2 ]
	1	[ 3    4 ] ]
	0	1

Predicted target

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$
$$\text{Rec_Class0} = 7 / (7 + 2) = 0.777$$
$$\text{Rec_Class1} = 4 / (4 + 3) = 0.571$$



# Receiver Operating Characteristic (ROC)



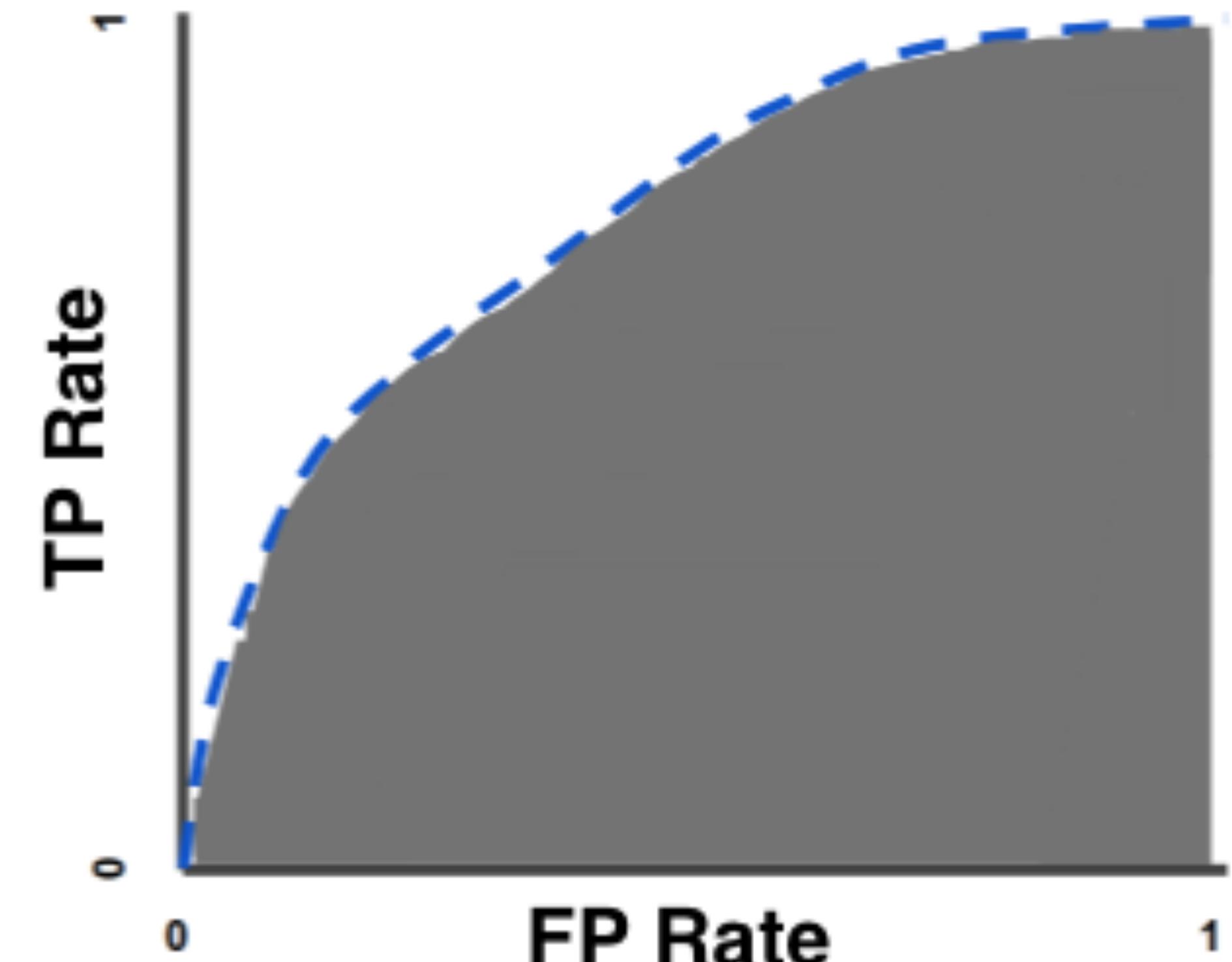
- Compares the True Positive Rate (TPR) on Y-axis, to the False Positive Rate (FPR) on the X- axis

```
import matplotlib.pyplot as plt  
import scikitplot as skplt  
  
skplt.metrics.plot_roc(target_test,  
predicted_probs)  
  
plt.show()
```

# Area Under Curve (AUC)

- AUC is an aggregate measure of performance across all possible classification thresholds.
- AUC ranges from 0 to 1 and a model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

```
from sklearn.metrics import roc_auc_score  
  
roc_auc_score(target_test, target_preds)
```



# Quiz: compute metrics for each class of 3-class confusion matrix

```
target = [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(confusion_matrix(target, target_pred))
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target [ [ 7 2 5 ]  
[ 3 4 2 ]  
[ 0 1 4 ] ]  
Predicted target

Accuracy = (Sum Diagonal)/ N

Precision = TP/ (TP + FP)

Recall = TP/ (TP + FN)

# Quiz: compute metrics for each class of 3-class confusion matrix

```
target = [1,0,0,1,1,1,1,1,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2]
print(confusion_matrix(target, target_pred))
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

$$\begin{aligned} \text{Accuracy} &= (7+4+4)/28 = 0.5357 \\ \text{Prec\_Class2} &= 4/(4 + 5 + 2) = 0.3636 \\ \text{Rec\_Class2} &= 4/(4 + 1 + 0) = 0.8 \\ &\dots \end{aligned}$$

True target  
[ [ 7 2 5 ]  
[ 3 4 2 ]  
[ 0 1 4 ] ]  
Predicted target

Precision: [ 0.7 0.57142857 0.36363636 ]  
Recall: [ 0.5 0.44444444 0.8 ]

# Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!  
How about precision and recall?



$$\begin{bmatrix} [10 & 0] \\ [5 & 1] \end{bmatrix}$$

A

$$\begin{bmatrix} [7 & 2] \\ [3 & 4] \end{bmatrix}$$

B

$$\begin{bmatrix} [0 & 4] \\ [1 & 11] \end{bmatrix}$$

C

# Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!  
How about precision and recall?



Precision: [ 0.7 1]  
Recall: [ 1 0.2]

Precision: [ 0.7 0.7]  
Recall: [ 0.8 0.6]

Precision: [ 0 0.7]  
Recall: [ 0 0.9]

[ [ 10 0 ]

[ [ 7 2 ]

[ [ 0 4 ]

[ 5 1 ] ]

[ 3 4 ] ]

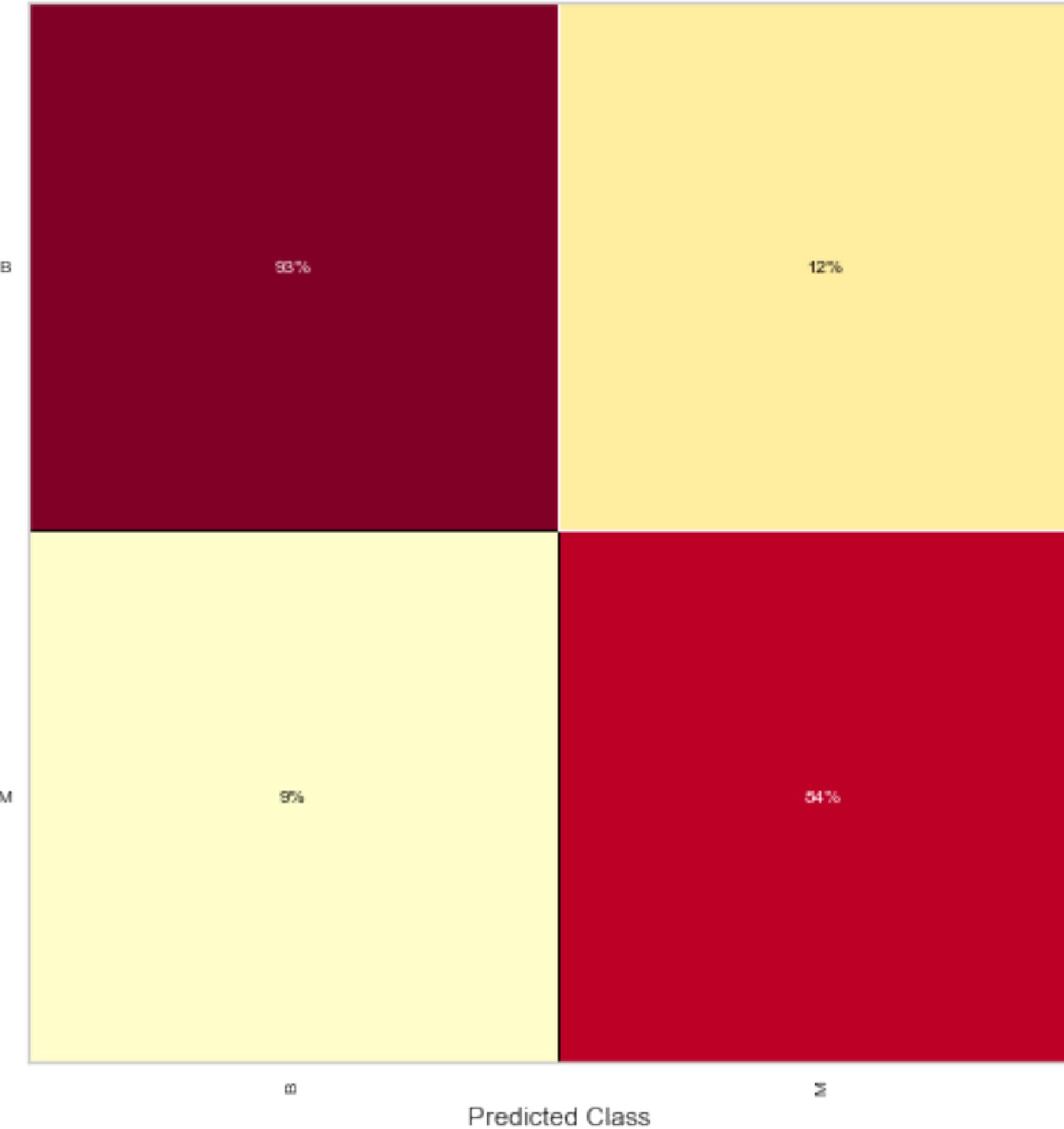
[ 1 11 ] ]

A

B

# Visualizing the Confusion Matrix

RandomForestClassifier Confusion Matrix

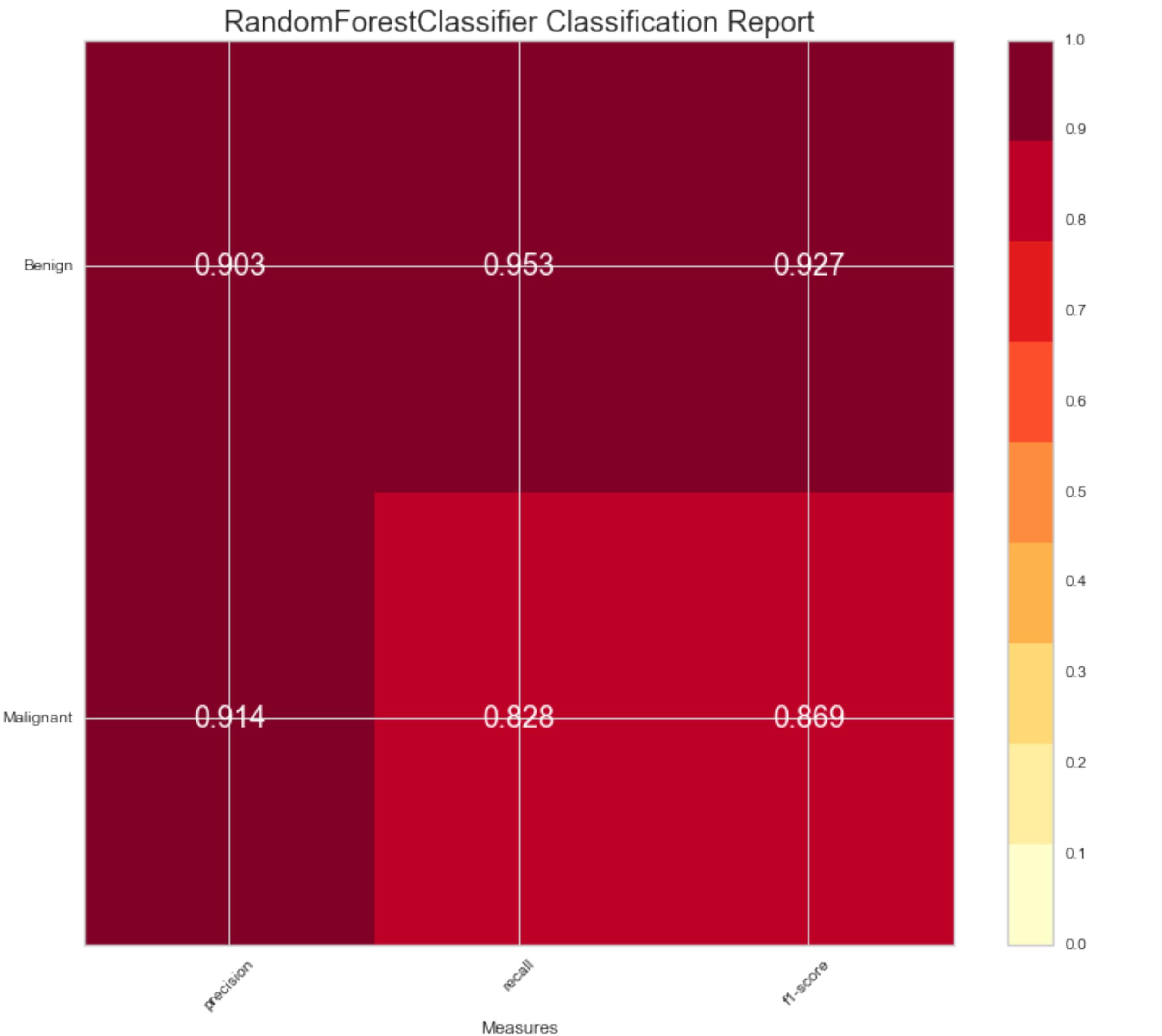


```
from yellowbrick.classifier import ConfusionMatrix

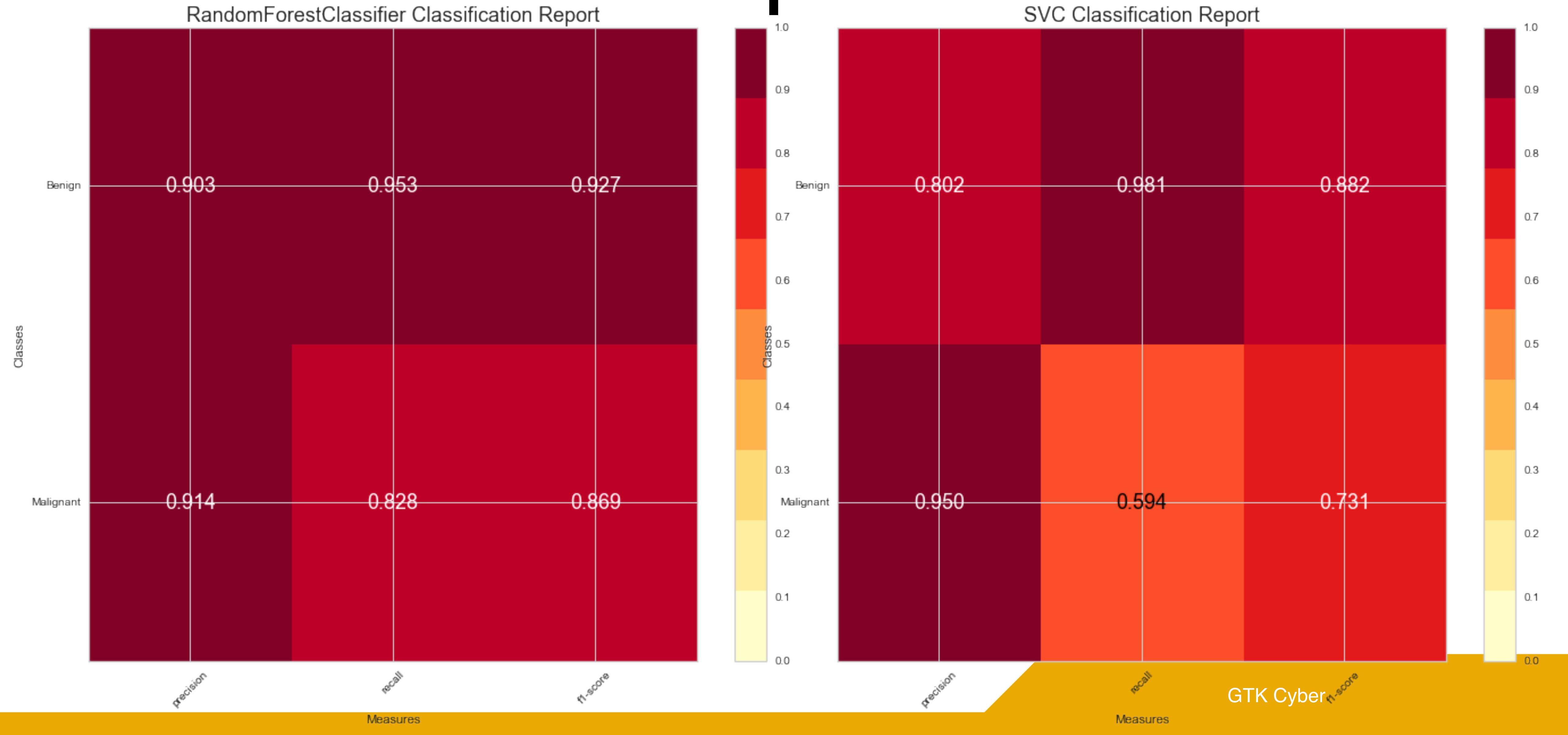
#Create the models for both the Random Forest
random_forest_model = RandomForestClassifier()

random_forest_conf_matrix = ConfusionMatrix(
                                         random_forest_model )
random_forest_conf_matrix.fit( x_train, y_train )
random_forest_conf_matrix.score( x_test, y_test )
random_forest_conf_matrix.poof()
```

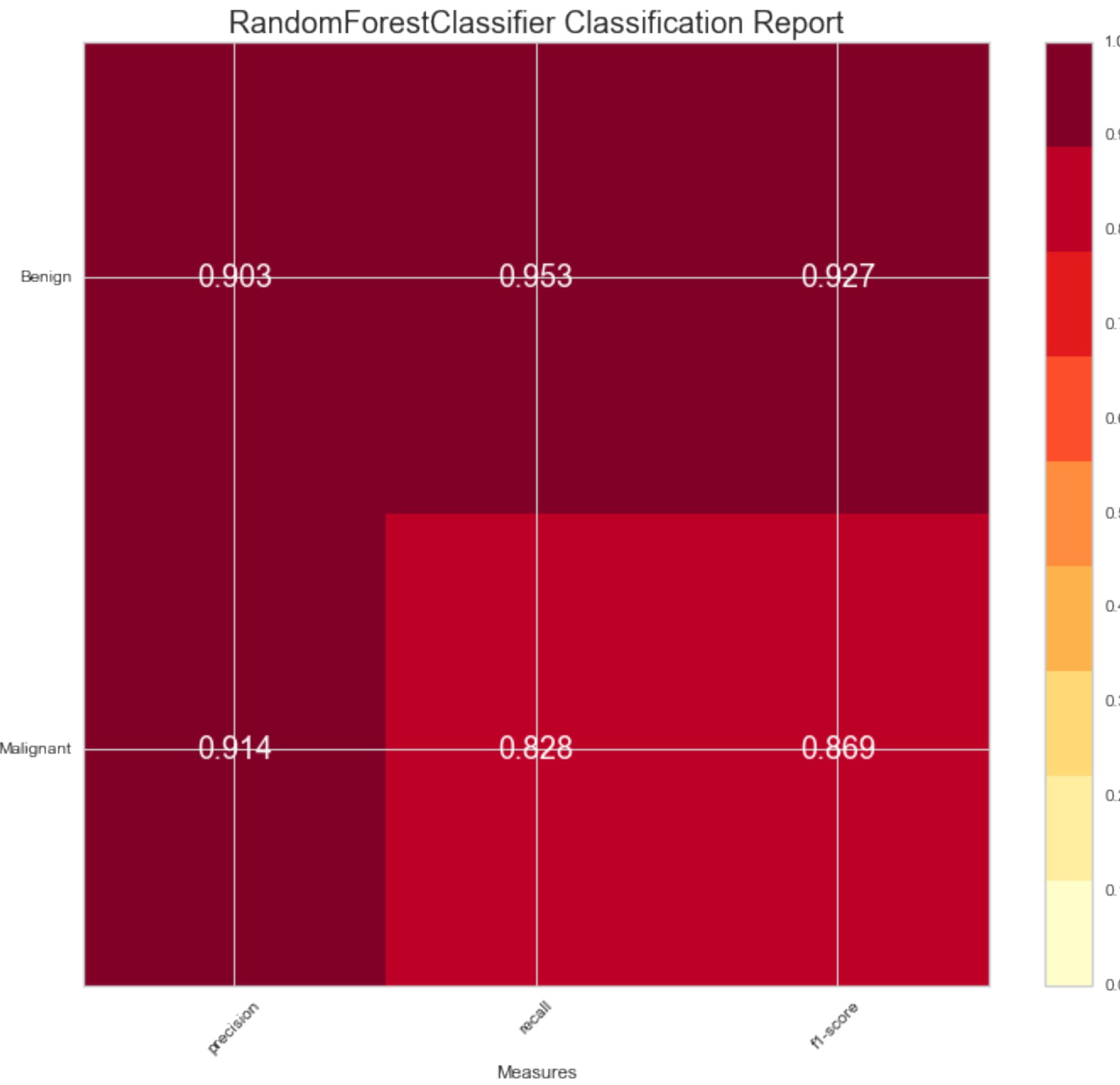
# Visualizing the Classification Report



# Visualizing the Classification Report



# Visualizing the Classification Report

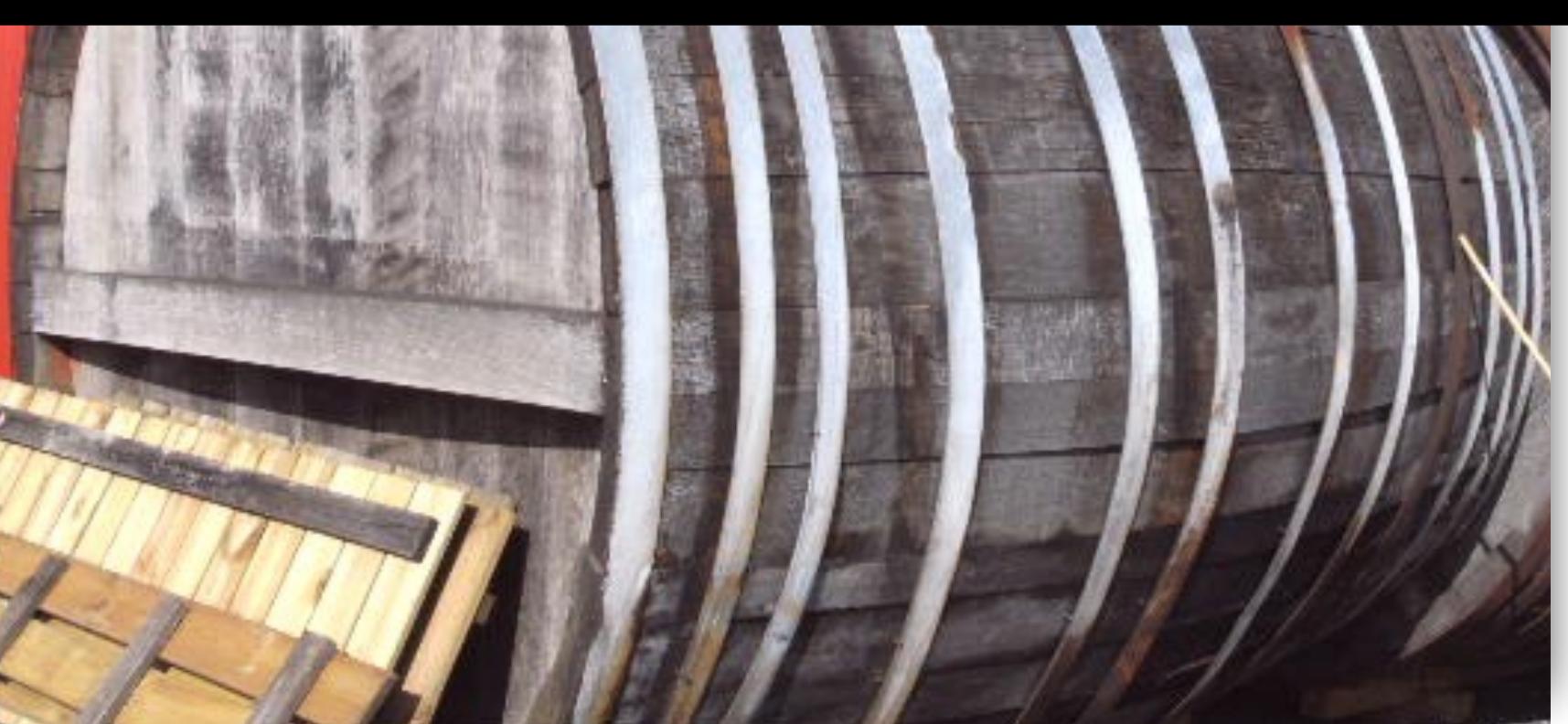


```
from yellowbrick.classifier import ClassificationReport  
  
random_forest_class_report = ClassificationReport( random_forest_model,  
                                                    classes=[ 'Benign', 'Malignant' ] )  
random_forest_class_report.fit(X_train, y_train)  
random_forest_class_report.score(X_test, y_test)  
random_forest_class_report.poof()
```

# Cross-Validation

# Split data for train and test

features(X)



70%

target (y)



# Simple train test split!

```
# Simple Cross-Validation: Split the data set into training and test data  
  
x_train, x_test, target_train, target_test =  
model_selection.train_test_split(X, target, test_size=0.25)
```

# Simple cross-validation!

```
## Train the classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x_train, target_train)

## Making predictions using test data
target_pred = clf.predict(x_test)

## Report metrics
```

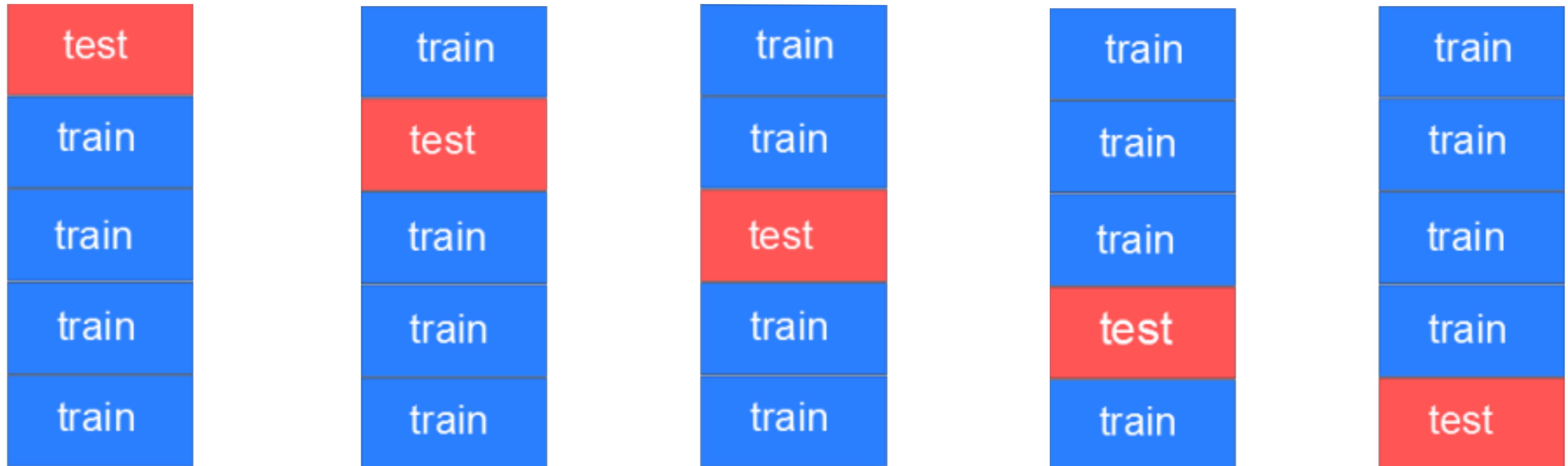
# Cross Validation

**Better approach:** Create a bunch of train/test splits, calculate the testing accuracy for each, and average the results together.

# Cross Validation

1. Split the dataset into  $K$  **equal** partitions (or "folds").
2. Use fold 1 as the **testing set** and the union of the other folds as the **training set**.
3. Calculate testing accuracy.
4. Repeat steps 2 and 3  $K$  times, using a **different fold** as the testing set each time.
5. Use the **average testing accuracy** as the estimate of out-of-sample accuracy.

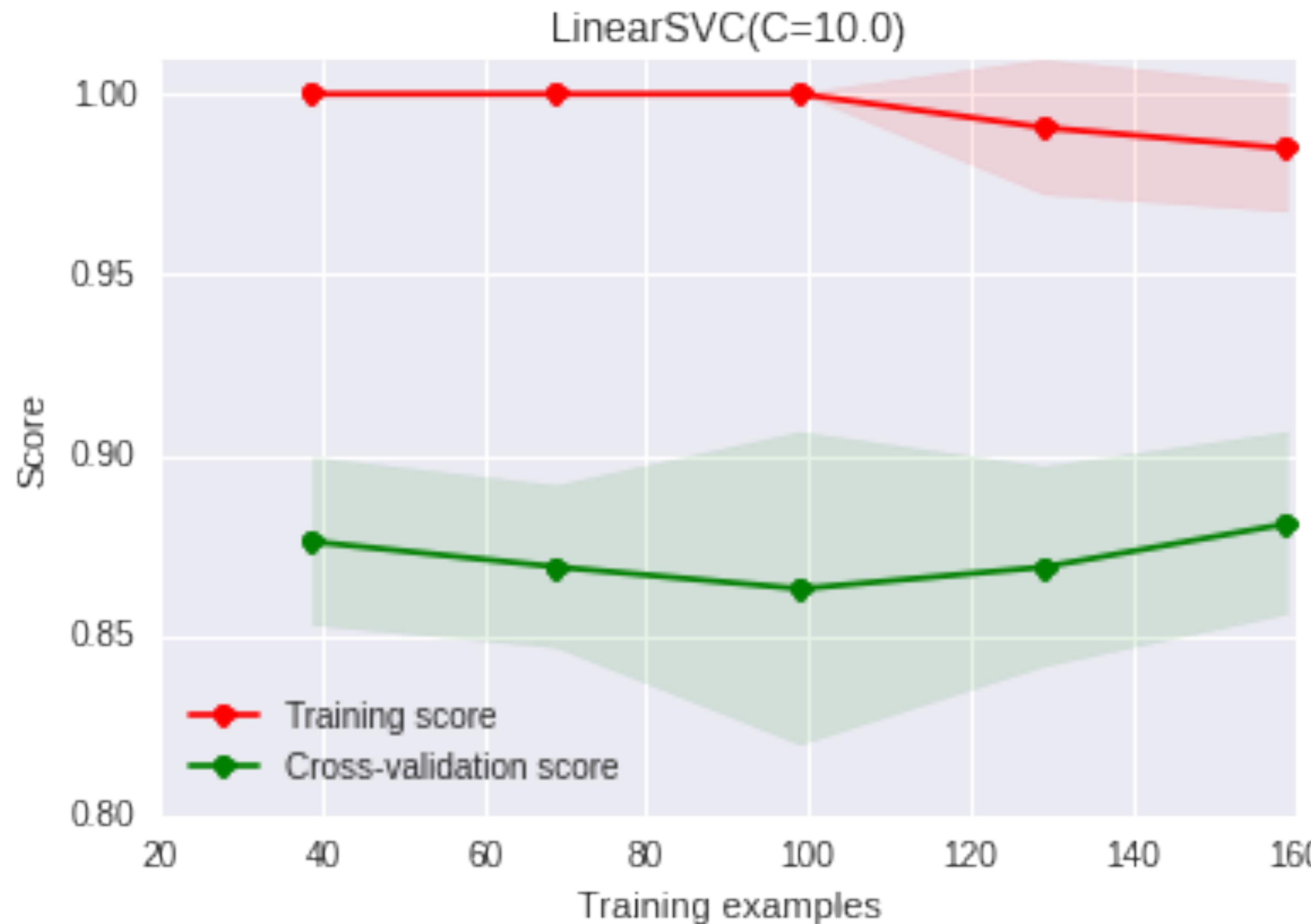
# K-Fold Cross Validation



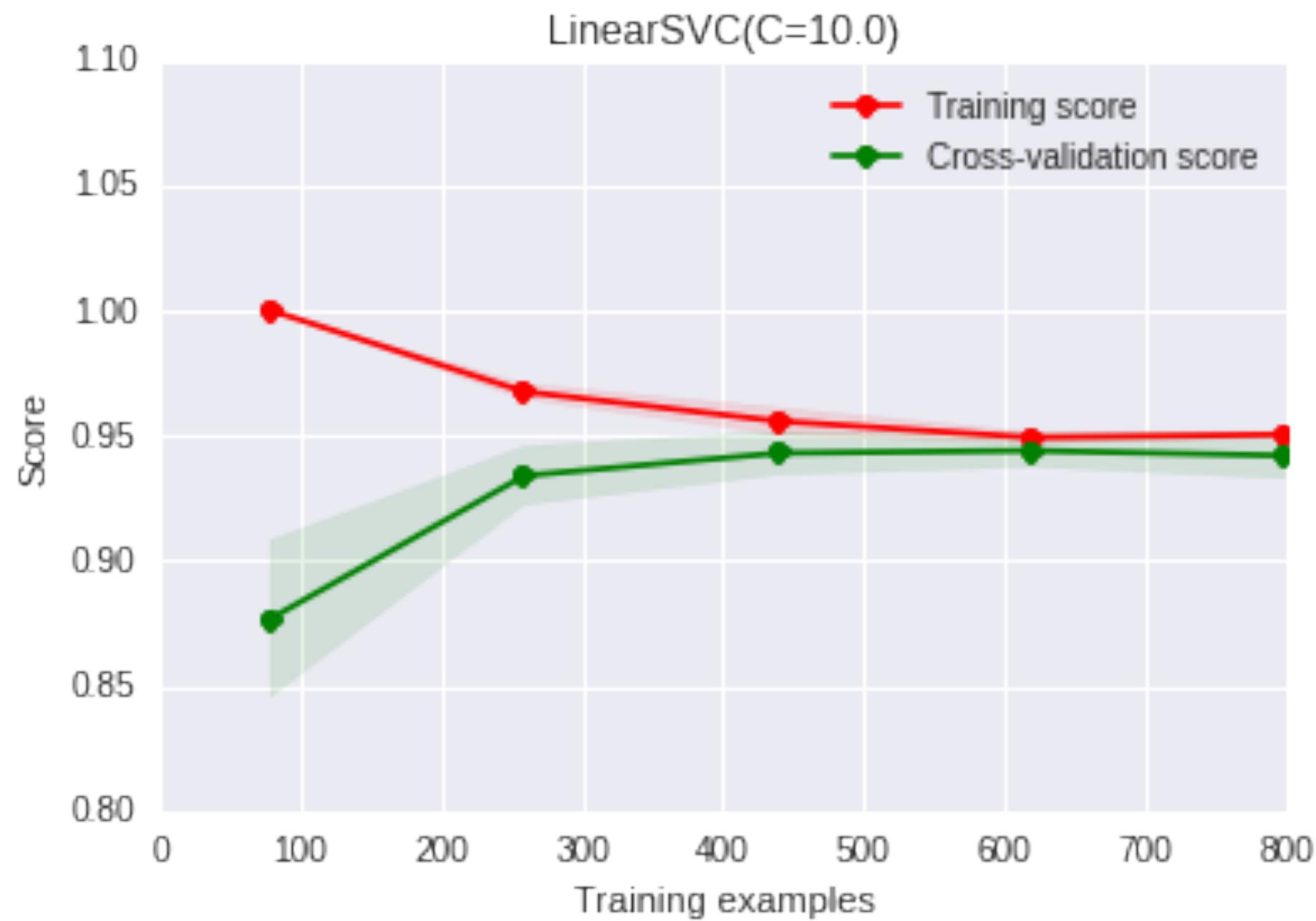
# K-fold Cross-Validation

```
scores = cross_val_score(clf, features, target, cv=<n>)
scores.mean()
```

# Visualizing Goodness of Fit



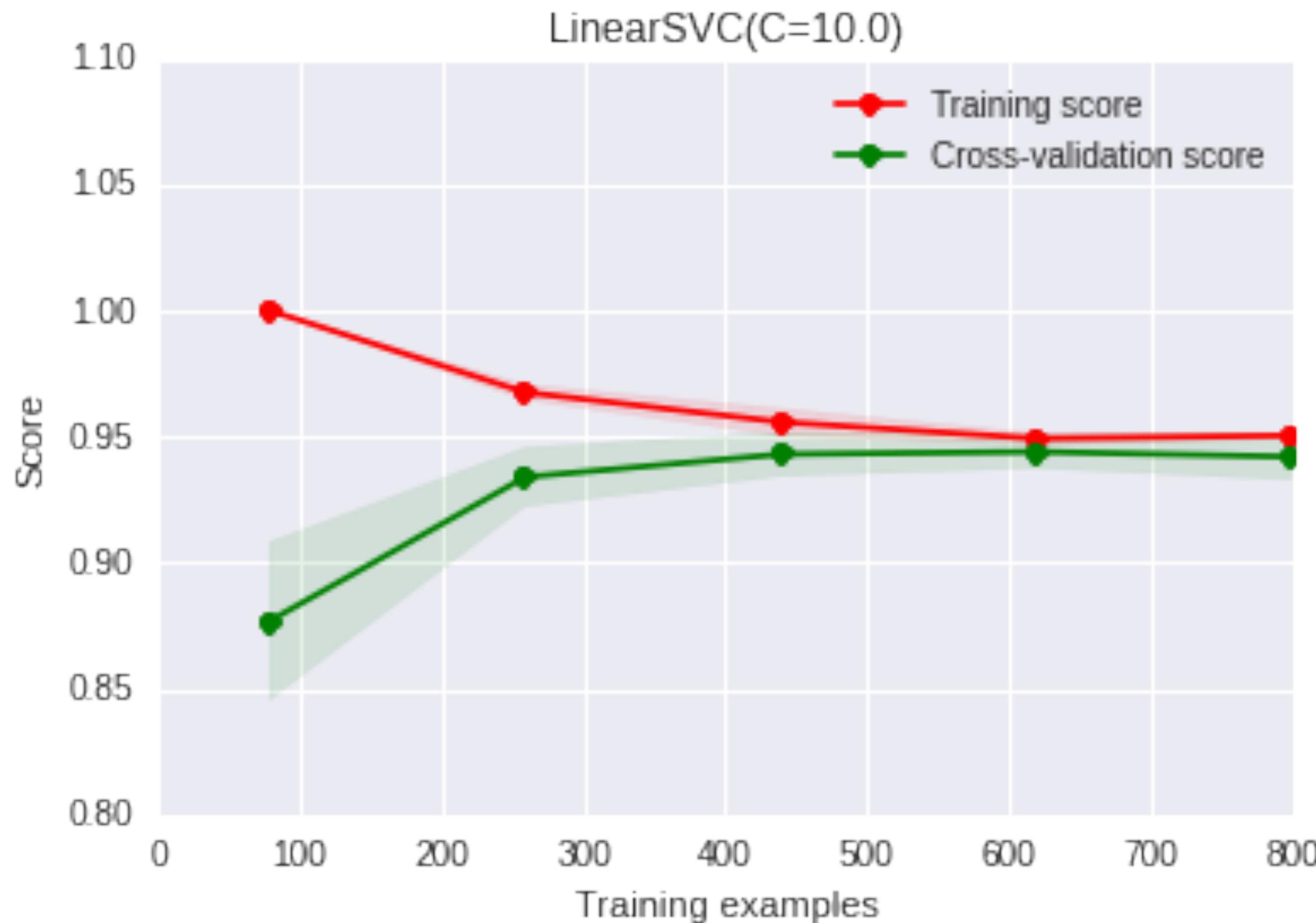
# Visualizing Goodness of Fit



```
from yellowbrick.classifier.learning_curve import LearningCurveVisualizer

viz = LearningCurveVisualizer(GaussianNB())
viz.fit(X,y)
viz.poof()
```

# Visualizing Goodness of Fit



```
from yellowbrick.classifier.learning_curve import LearningCurveVisualizer  
viz = LearningCurveVisualizer(GaussianNB())  
viz.fit(X,y)  
viz.poof()
```

# **In-Class Exercise**

Please complete Worksheet 3: DGA Detection Using Supervised Learning



# Hyper-Parameter Tuning!

on: e.g. Gradient Descent!

# Grid Search

```
RandomForestClassifier(bootstrap=True,  
                      class_weight=None,  
                      criterion='gini',  
                      max_depth=None,  
                      max_features='auto',  
                      max_leaf_nodes=None,  
                      min_impurity_decrease=0.0,  
                      min_impurity_split=None,  
                      min_samples_leaf=1,  
                      min_samples_split=2,  
                      min_weight_fraction_leaf=0.0,  
                      n_estimators=10,  
                      n_jobs=1,  
                      oob_score=False  
)
```

# Tuning these parameters

- GridSearchCV: You provide a list of possible parameters
- RandomizedSearchCV: Random combinations are searched

# Grid Search

```
param_grid = [  
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},  
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001]},  
    'kernel': ['rbf']},  
]
```

# Grid Search

```
# use a full grid over all parameters
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [2, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}

# run grid search
grid_search = GridSearchCV(clf, param_grid=param_grid)
start = time()
grid_search.fit(X, y)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
      % (time() - start, len(grid_search.cv_results_['params'])))
report(grid_search.cv_results_)
```

# Random Search

```
# specify parameters and distributions to sample from
param_dist = {"max_depth": [3, None],
              "max_features": sp_randint(1, 11),
              "min_samples_split": sp_randint(2, 11),
              "min_samples_leaf": sp_randint(1, 11),
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}

# run randomized search
n_iter_search = 20
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,
                                    n_iter=n_iter_search)

start = time()
random_search.fit(X, y)
print("RandomizedSearchCV took %.2f seconds for %d candidates"
      " parameter settings." % ((time() - start), n_iter_search))
report(random_search.cv_results_)
```

# Bayesian Search

```
pip install scikit-optimize
```

```
# run bayesian-optimized hyperparameter search
n_iter_search = 20
bayes_search = BayesSearchCV(clf,
param_distributions=param_dist,
                                         n_iter=n_iter_search)

start = time()
bayes_search.fit(X, y)
print("BayesSearchCV took %.2f seconds for %d candidates"
      " parameter settings." % ((time() - start),
n_iter_search))
report(bayes_search.cv_results_)
```

# Auto ML

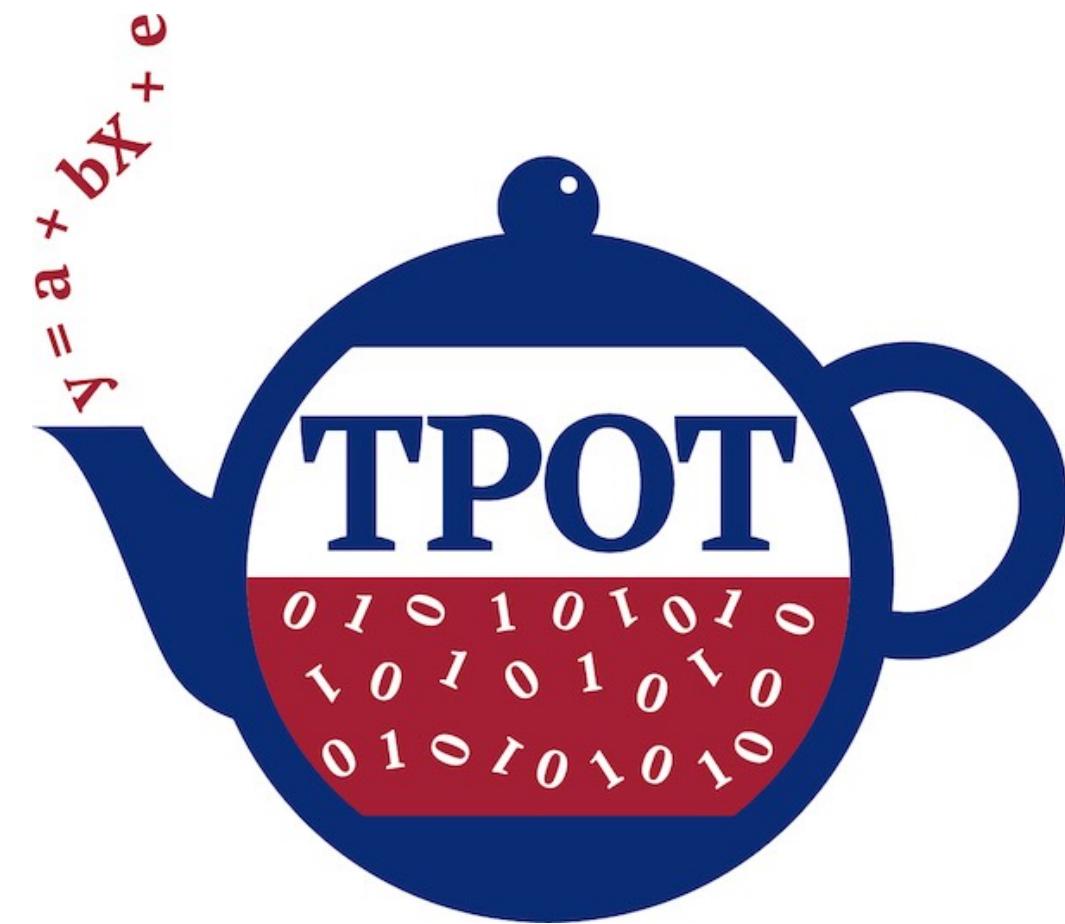
**What else can you automate?**

# Auto ML

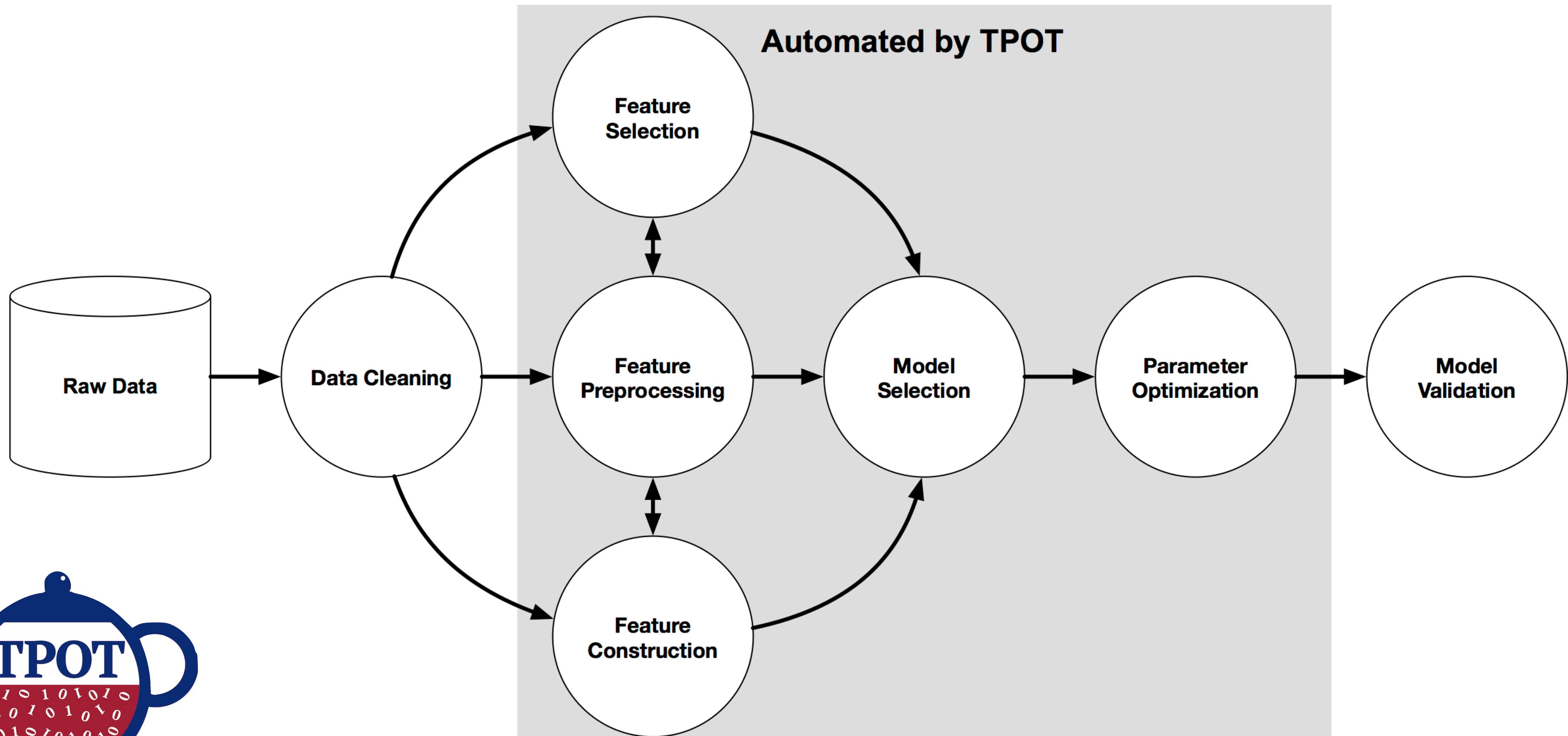
What else can you automate? It turns out, **almost everything** with TPOT.



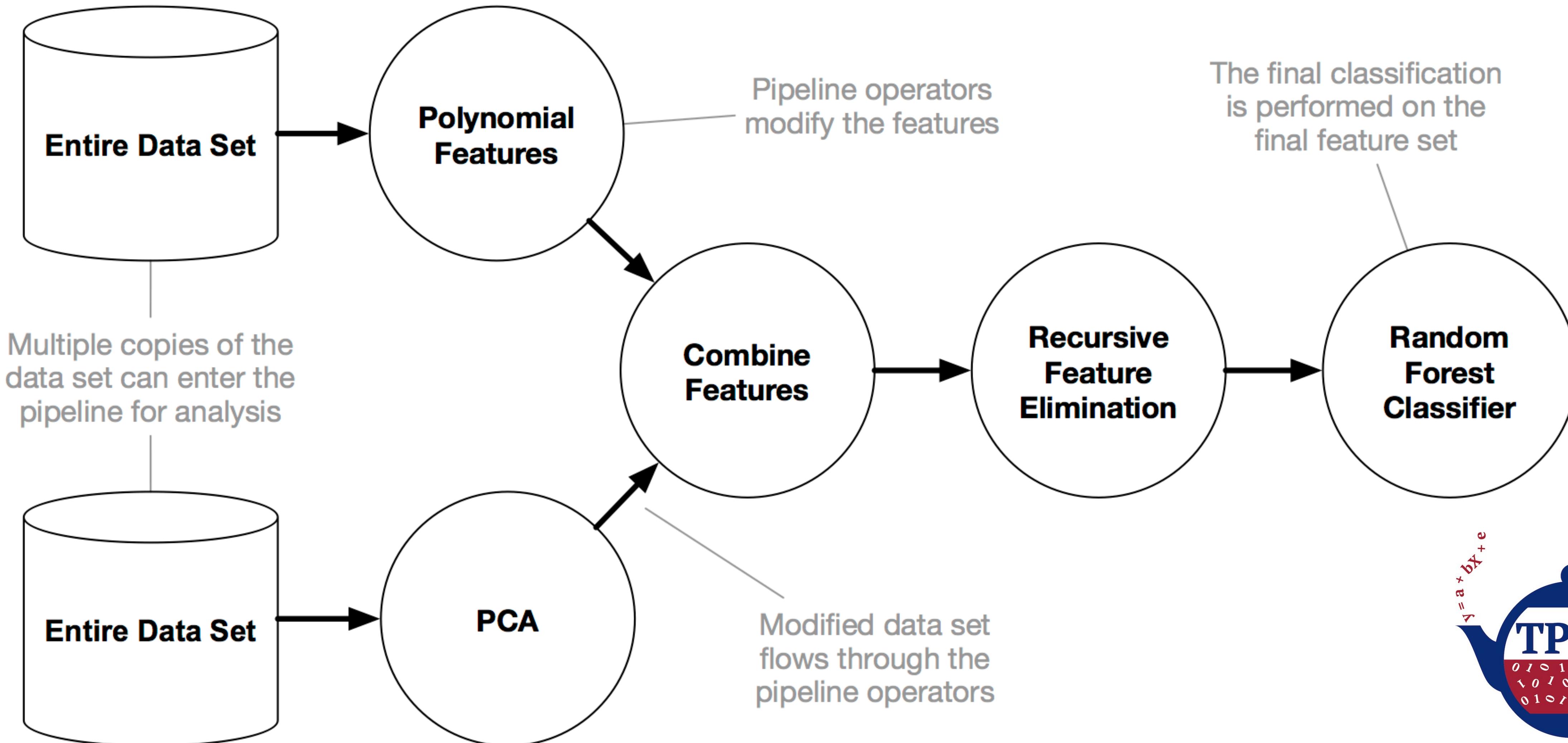
**TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your data.**



# Auto ML



# Auto ML



# Auto ML

```
from tpot import TPOTClassifier  
  
tpot = TPOTClassifier(generations=5, population_size=20, verbosity=2)  
tpot.fit(features_train, target_train)  
print(tpot.score(features_test, target_test))  
tpot.export('tpot_pipeline.py')
```



# Auto ML

Docs available here: <http://epistasislab.github.io/tpot/>



# Questions?

# **Thank You!**

# **Module 5: Hunting with Data Science**

Increasing the Signal-to-Noise Ratio

# **what is it?**

# **Semantics**

# Threat Hunting

Cyber threat hunting is "the process of proactively and iteratively searching through networks to detect and isolate advanced threats that evade existing security solutions."

# Data Science

Data science is an interdisciplinary field about scientific methods, processes, and systems to **extract knowledge or insights from data** in various forms, either structured or unstructured, similar to data mining.

Threat Hunting

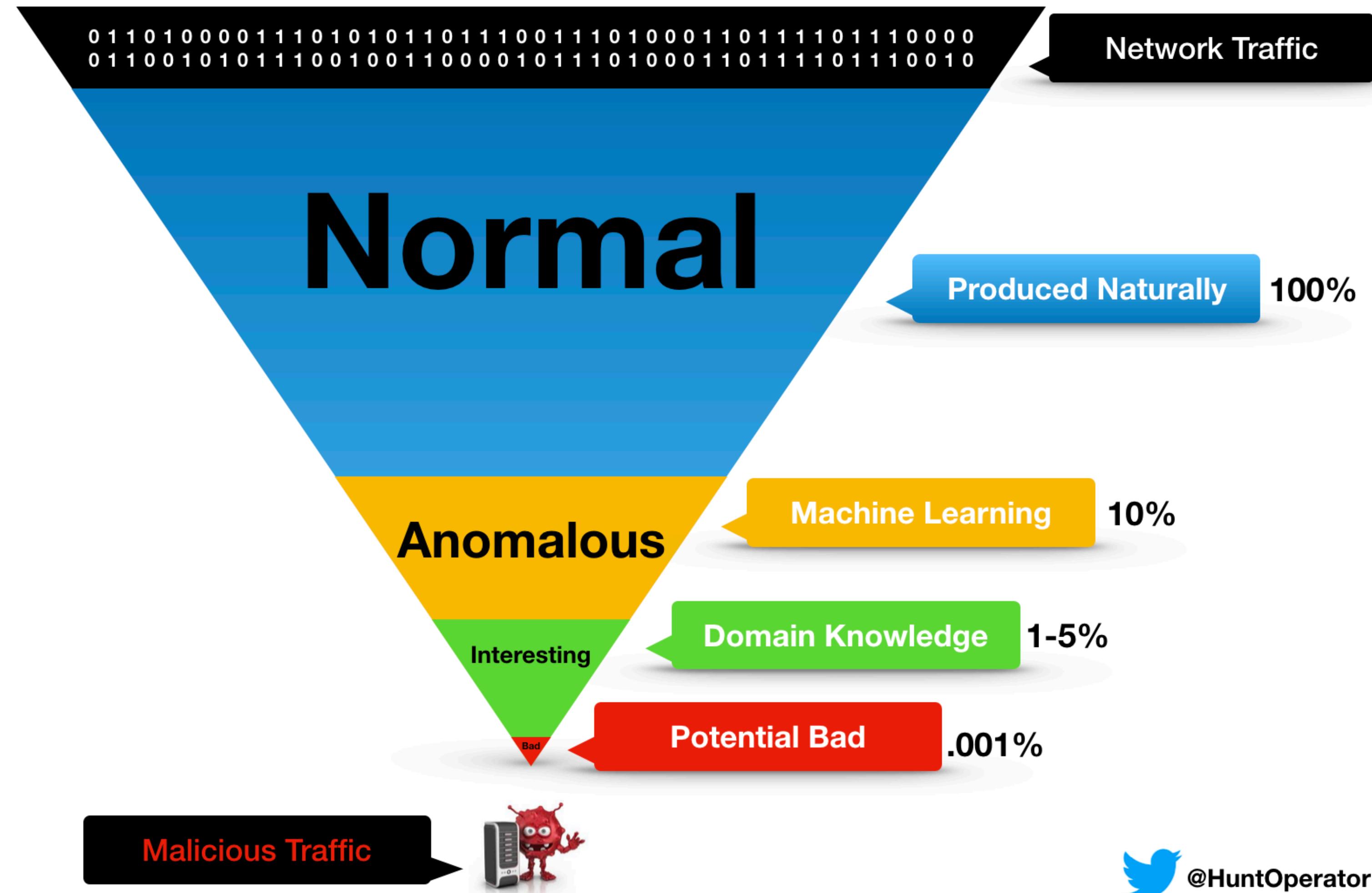
+

Data Science

---

# Hunting with Data Science

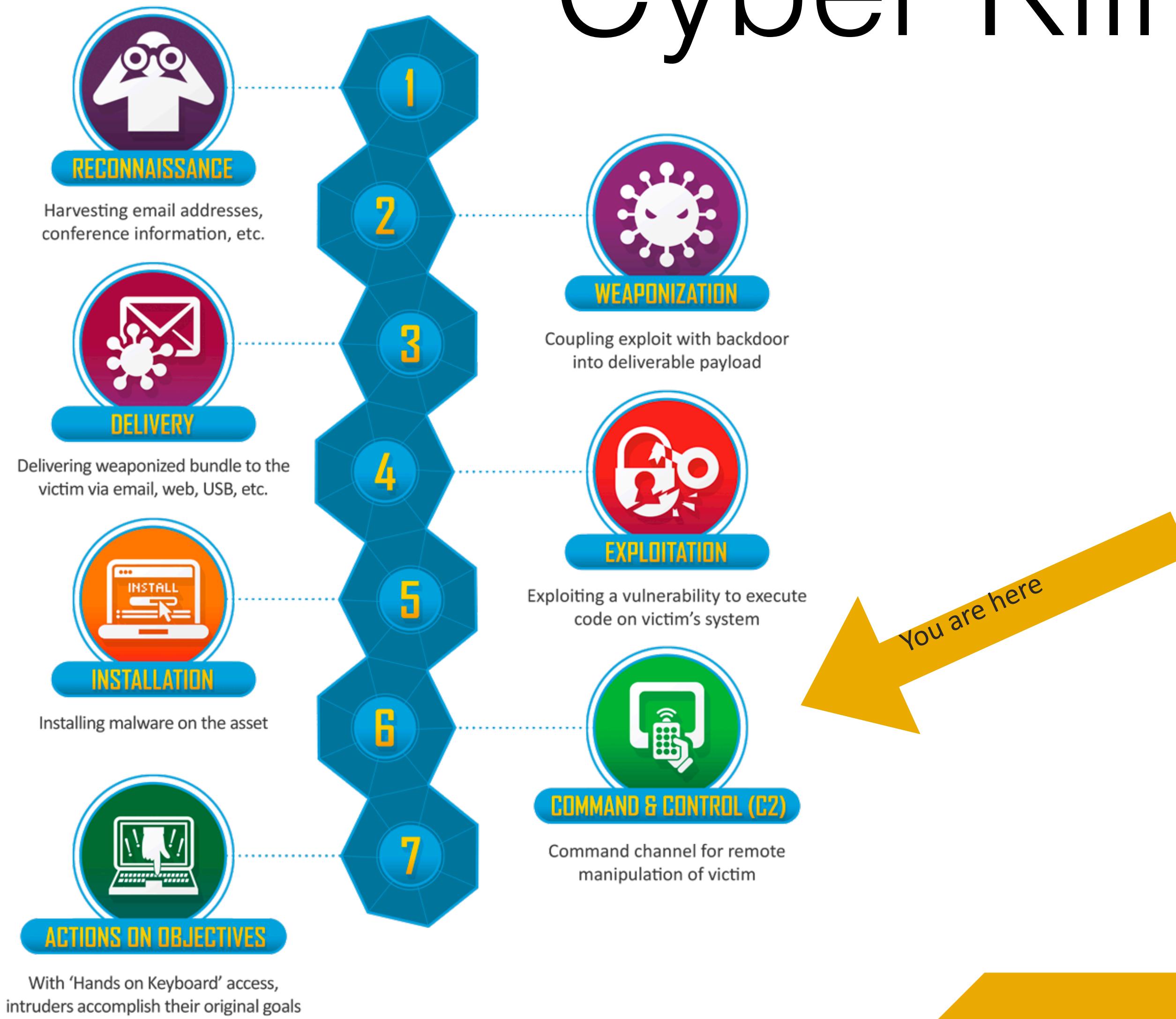
# Data Science Hunting Funnel



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

# Cyber Kill Chain

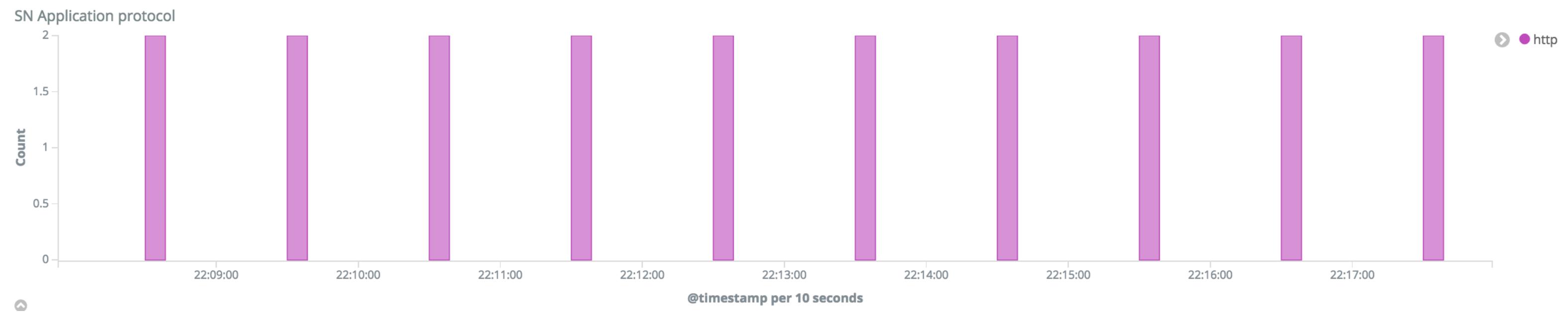
# Cyber Kill Chain



Beaconing  
DGA

# **Beaconing**

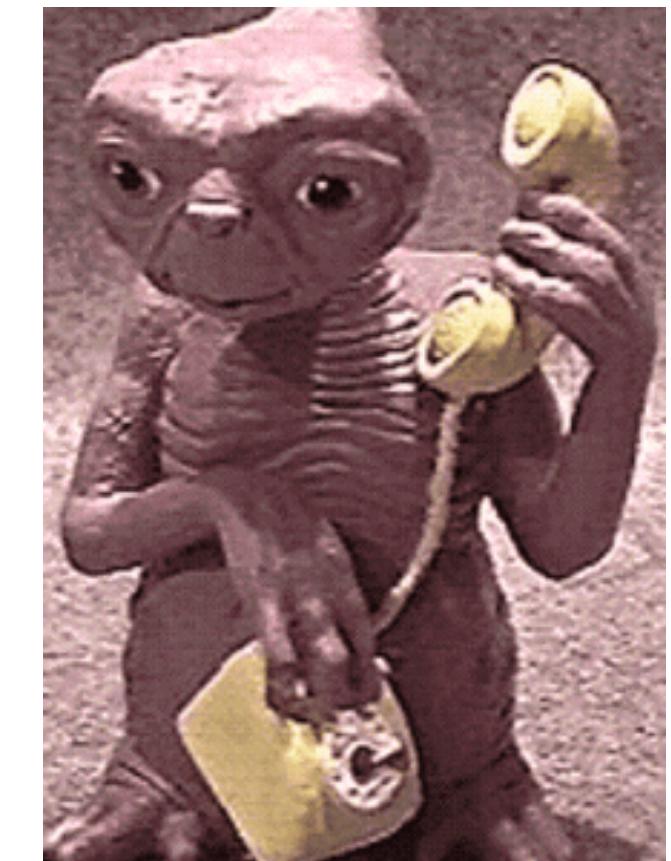
# Beaconing



- Post-Infection

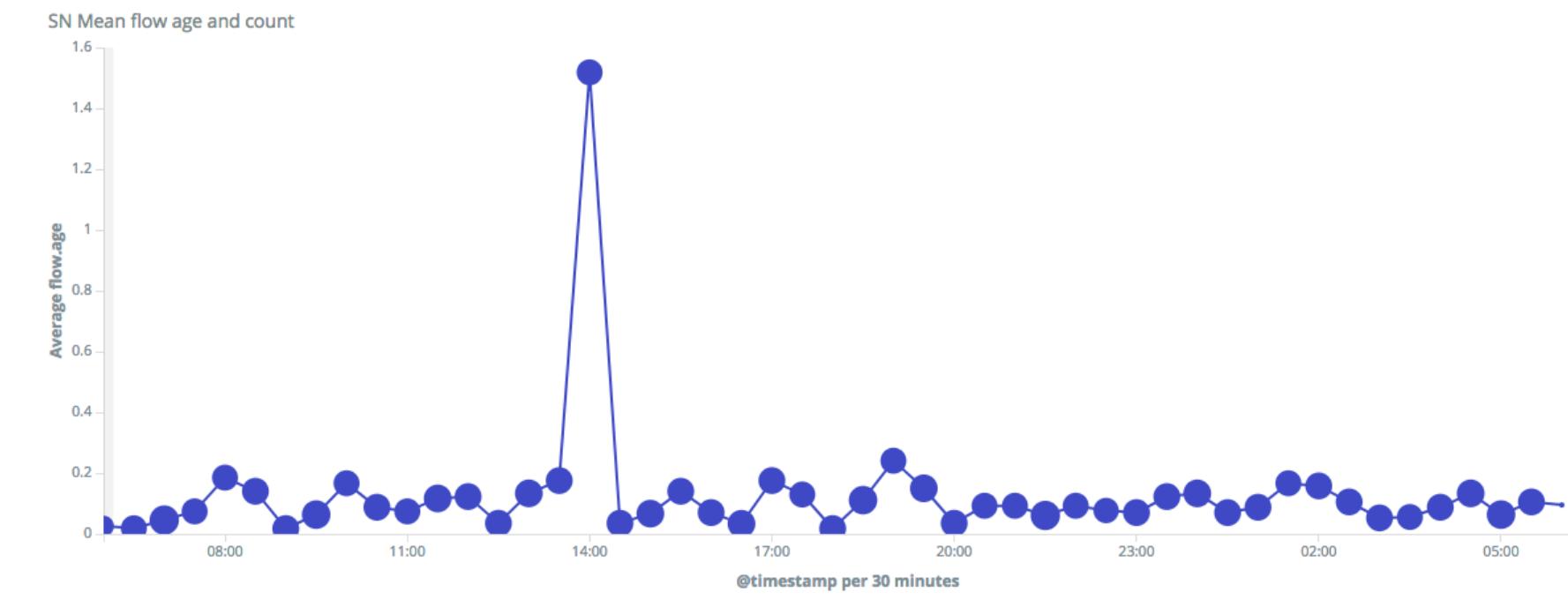
- Early network-related indication of infection

- Used by malware to “phone home” to command and control server



# Detection Challenges

- Hardset Intervals
- Varying window sizes
- Legit Services
- Windows update
- Virus definition updates



# Beaconing Detection

# Beaconing: Detection

- Free Open Source Software
- Designed for data scientists, security researchers
- Written in Python
- Used for rapid prototyping and development of behavioral analytics
- Intended to make identifying malicious behavior in networks as simple as possible.



<https://github.com/austin-taylor/flare>

# Beaconing: Detection

```
[beacon]
es_host=localhost          # IP address of ES Host, which we forwarded to localhost
es_index=logstash-flow-*   # ES index
es_port=9200                # Logstash port (we forwarded earlier)
es_timeout=480              # Timeout limit for elasticsearch retrieval
min_occur=50                # Minimum of 50 network occurrences to appear in traffic
min_interval=30             # Minimum interval of 30 seconds per beacon
min_percent=30              # Beacons must represent 30% of network traffic per dyad
window=3                     # Accounts for jitter... For example, if 60 second beacons
                             # occurred at 58 seconds or 62 seconds, a window of 3 would
                             # factor in that traffic.
threads=8                   # Use 8 threads to process (Should be configured)
period=24                    # Retrieve all flows for the last 24 hours.
kibana_version=5             # Your Kibana version. Currently works with 4 and 5
verbose=True                 # Display output while running script
```



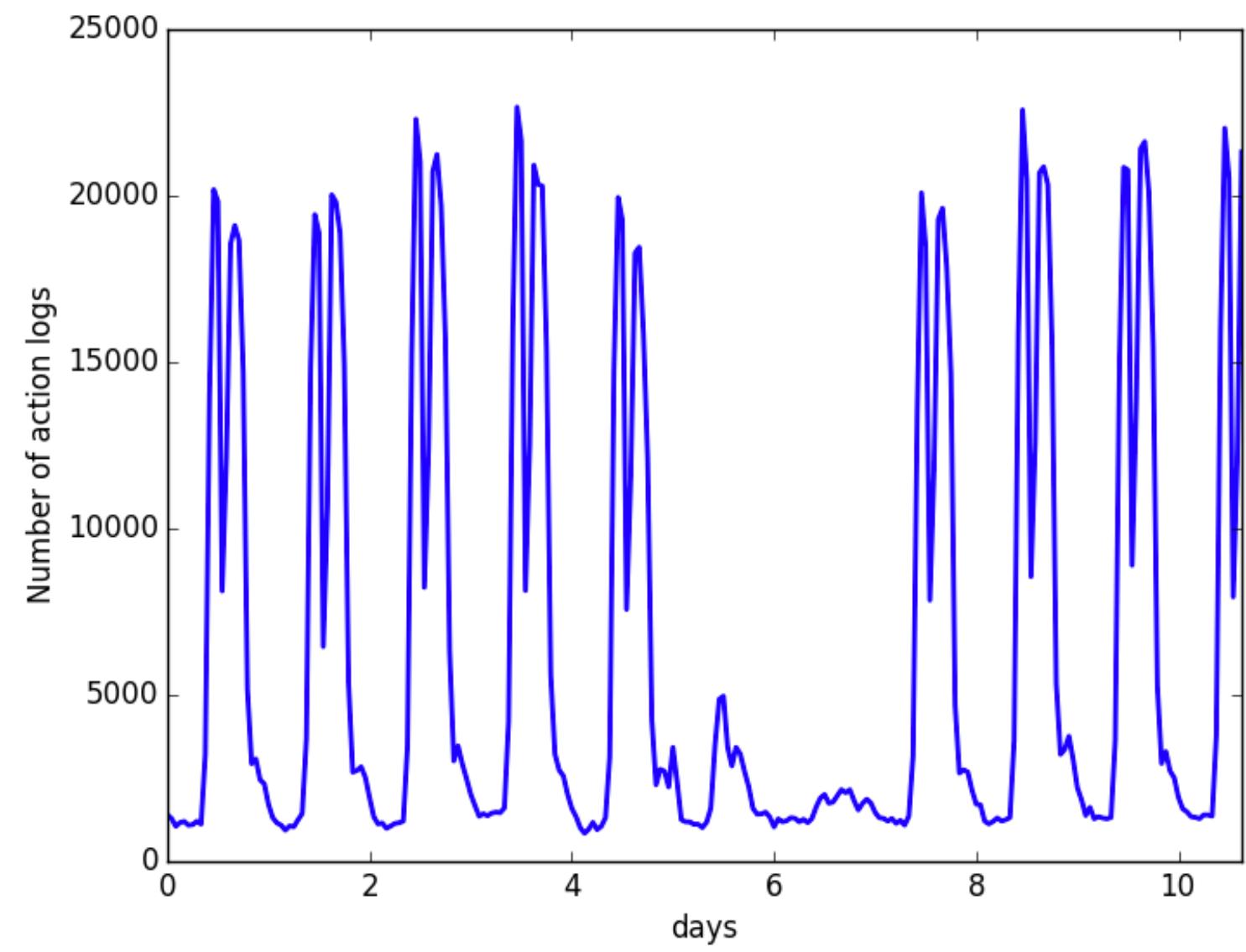
<https://github.com/austin-taylor/flare>

Time ▾	EveBox	src_ip	src_port	proto	dest_ip	dest_port	http.http_method	http.hostname
June 13th 2017, 10:49:01.935	<a href="#">Correlate Flow</a>	192.168.0.53	49182	TCP	160.153.76.129	80	GET	www.huntoperator.com

# Beaconing: Data Science

Simple: src\_ip, dest\_ip, dest\_port -> hash

More Complex: Discrete Fourier Transform (DFT)/Fast Fourier transform (FFT)



# Scenario 1

- A piece of malware has infected a computer (192.168.0.53) on your network and is trying to reach back to its Command and Control (C2) server (160.153.76.129) in periodic intervals



GTK Cyber

# Beaconing: Hunt

flare\_beacon -c configs/selks4.ini -csv beacons.csv

```
Austins-MacBook-Pro:flare_hunter$ flare_beacon -c configs/selks4.ini -csv beacons.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[SUCCESS] Writing csv to beacons.csv
```

108

events to process

# Beaconing: Hunt

```
flare_beacon -c configs/selks4.ini -csv --group --whois --focus_outbound beacons_filtered.csv
```

```
Austins-MacBook-Pro:flare_hunoperator$ flare_beacon -c configs/selks4.ini --whois --focus_outbound -csv beacons_filtered.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[INFO] Enriching IP addresses with whois information
[INFO] Applying outbound focus - filtering multicast, reserved, and private IP space
[SUCCESS] Writing csv to beacons_filtered.csv
```

31

events to process

# Beaconing: Hunt

```
flare_beacon -c configs/selks4.ini -csv --group --whois --focus_outbound beacons_filtered.csv
```

```
Austins-MacBook-Pro:flare_huntoperator$ flare_beacon -c configs/selks4.ini --whois --focus_outbound -csv beacons_filtered.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[INFO] Enriching IP addresses with whois information
[INFO] Applying outbound focus - filtering multicast, reserved, and private IP space
[SUCCESS] Writing csv to beacons_filtered.csv
```

## What was applied?

- group: This will group the results making it visually easier to identify anomalies.
- whois: Enriches IP addresses with WHOIS information through ASN Lookups.
- focus\_outbound: Filters out multicast, private and broadcast addresses from destination IPs**

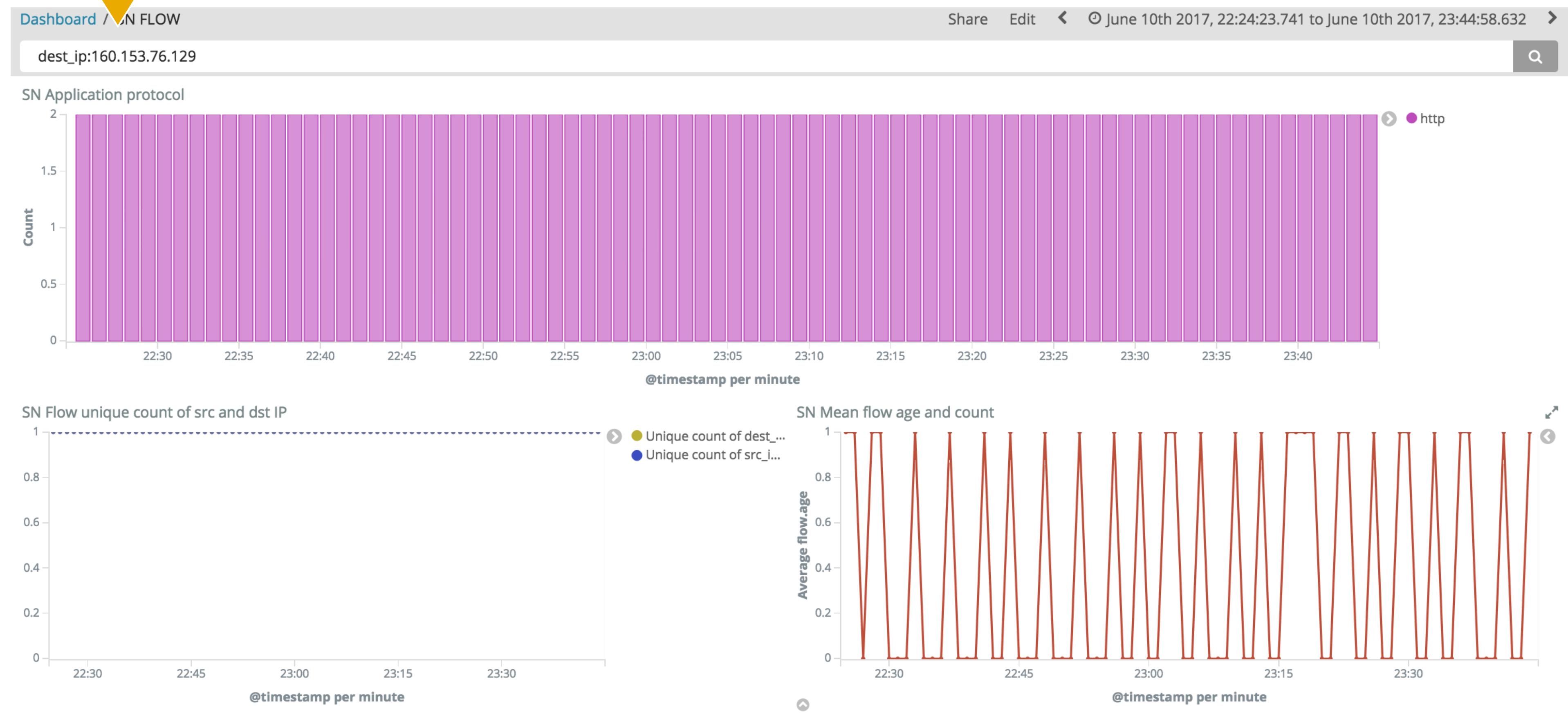
# Beaconing: Hunt

src_ip	dest_whois	dest_ip	dest_port	bytes_toserver	dest_degree	occurrences	percent	interval
192.168.0.100	GOOGLE - Google Inc., US	8.8.8	53	82	16	71	98	3601
	SOFTLAYER - SoftLayer Technologies Inc., US	198.23.79.227	123	90	1	71	98	3601
192.168.0.120	AMAZON-02 - Amazon.com, Inc., US	52.192.78.22	80	1432	3	34	70	1201
	GOOGLE - Google Inc., US	8.8.4.4	53	74	7	139	63	61
192.168.0.53	AS-26496-GO-DADDY-COM-LLC - GoDaddy.com, LLC, US	160.153.76.129	80	620	3	383	97	61
	GOOGLE - Google Inc., US	8.8.4.4	53	233	7	386	95	61
		8.8.8.8	53	65	16	386	95	61
192.168.0.60	AKAMAI-ASN1 , US	104.95.176.44	80	8399	1	3	33	97
	AMAZON-AES - Amazon.com, Inc., US	174.129.1.163	80	1691	1	77	38	101
		204.236.238.3	80	885	1	84	39	111
		23.21.110.37	80	1691	1	53	35	200
		23.21.67.210	80	1691	1	101	32	200
		23.21.98.133	80	1691	1	65	40	101
		23.23.173.186	80	1691	1	43	32	200
		23.23.236.78	80	759	1	313	41	112
		50.16.215.193	80	1691	1	25	40	101
		50.17.254.18	80	1691	1	25	40	200
	DROPBOX - Dropbox, Inc., US	162.125.32.5	443	60	1	53	35	602
	EDGECAST - MCI Communications Services, Inc. d/b/a Verizon Business, US	152.195.54.20	80	556	2	69	49	601
	LLNW - Limelight Networks, Inc., US	69.164.0.0	80	6242	2	3	33	36
	MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation, US	13.107.6.151	443	3985	1	151	41	300

- **bytes\_toserver:** Total sum of bytes sent from IP address to Server
- **dest\_degree:** Amount of source IP addresses that communicate to the same destination
- **occurrences:** Number of network occurrences between dyads identified as beaconing.
- **percent:** Percent of traffic between dyads considered beaconing.
- **interval:** Intervals between each beacon in seconds

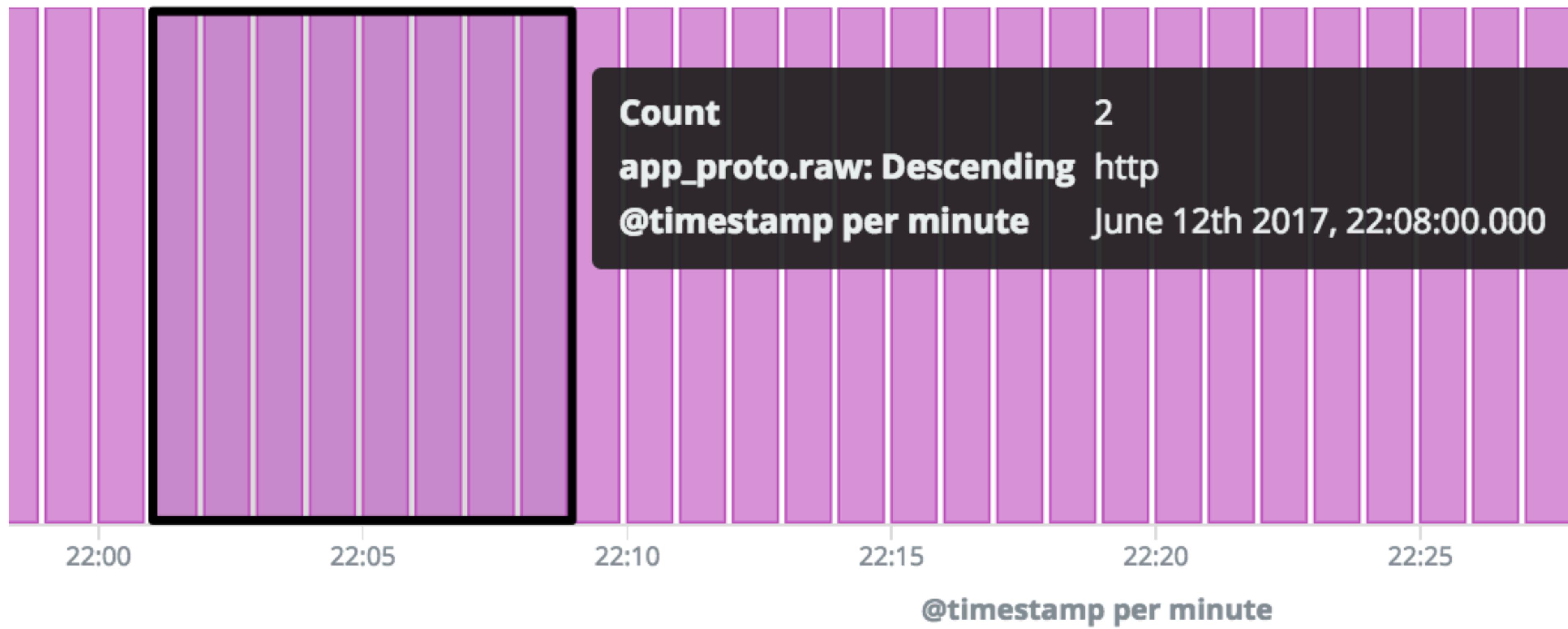
# Beaconing: Hunt

## Validate Results

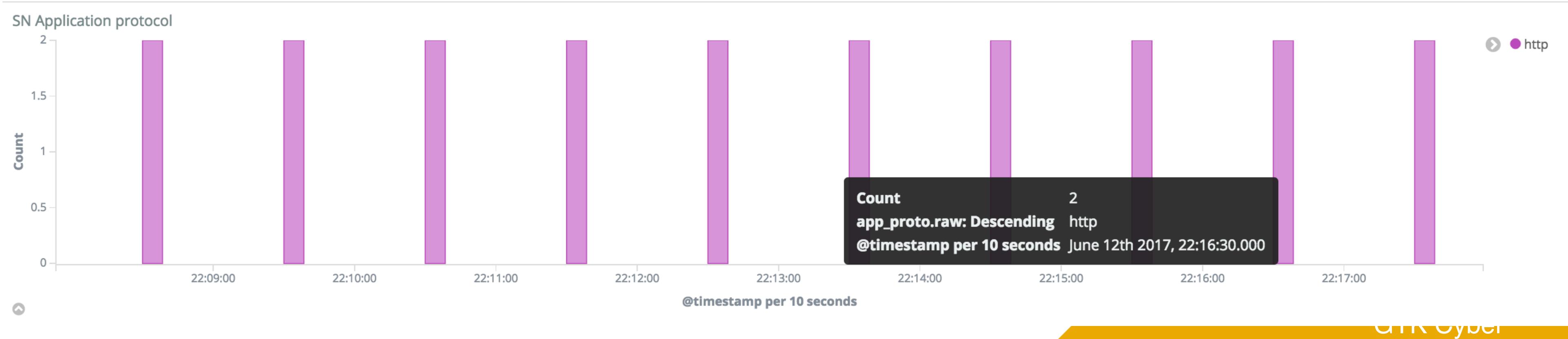
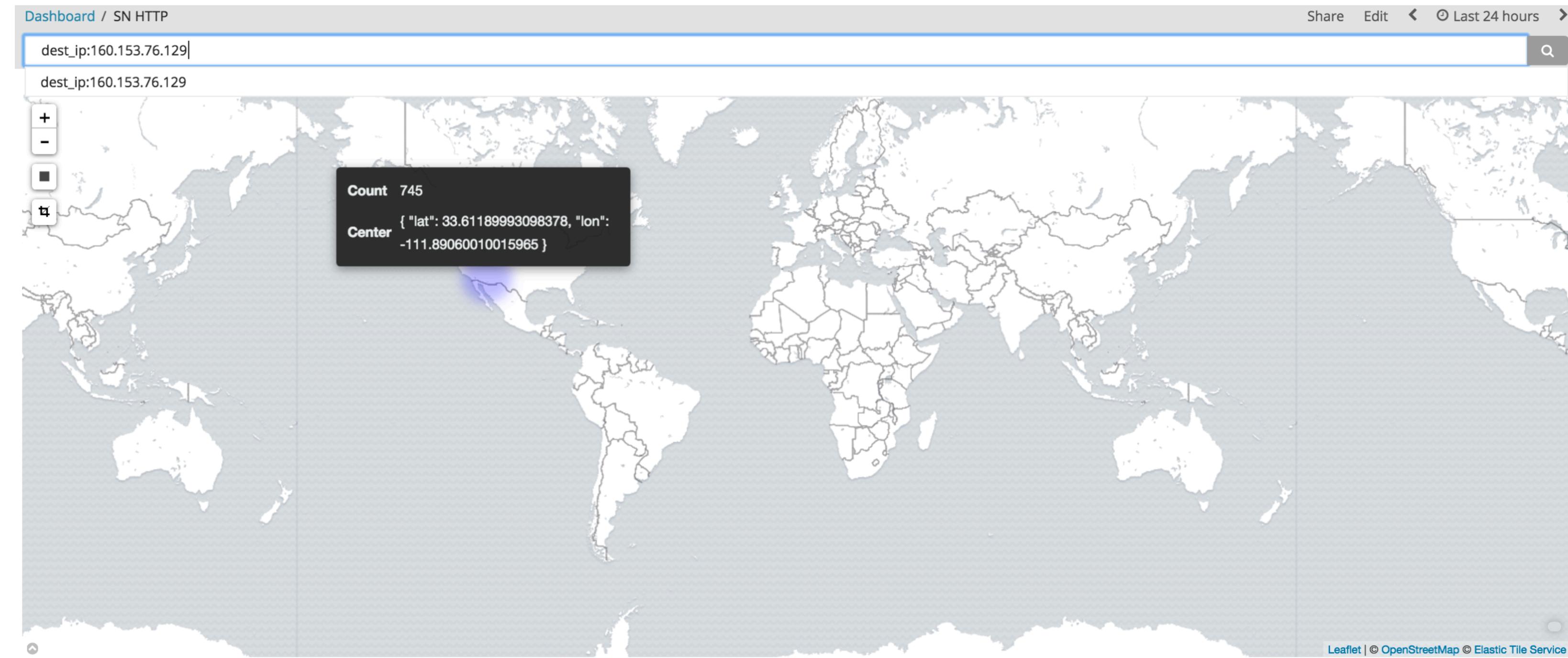


# Beaconing: Hunt

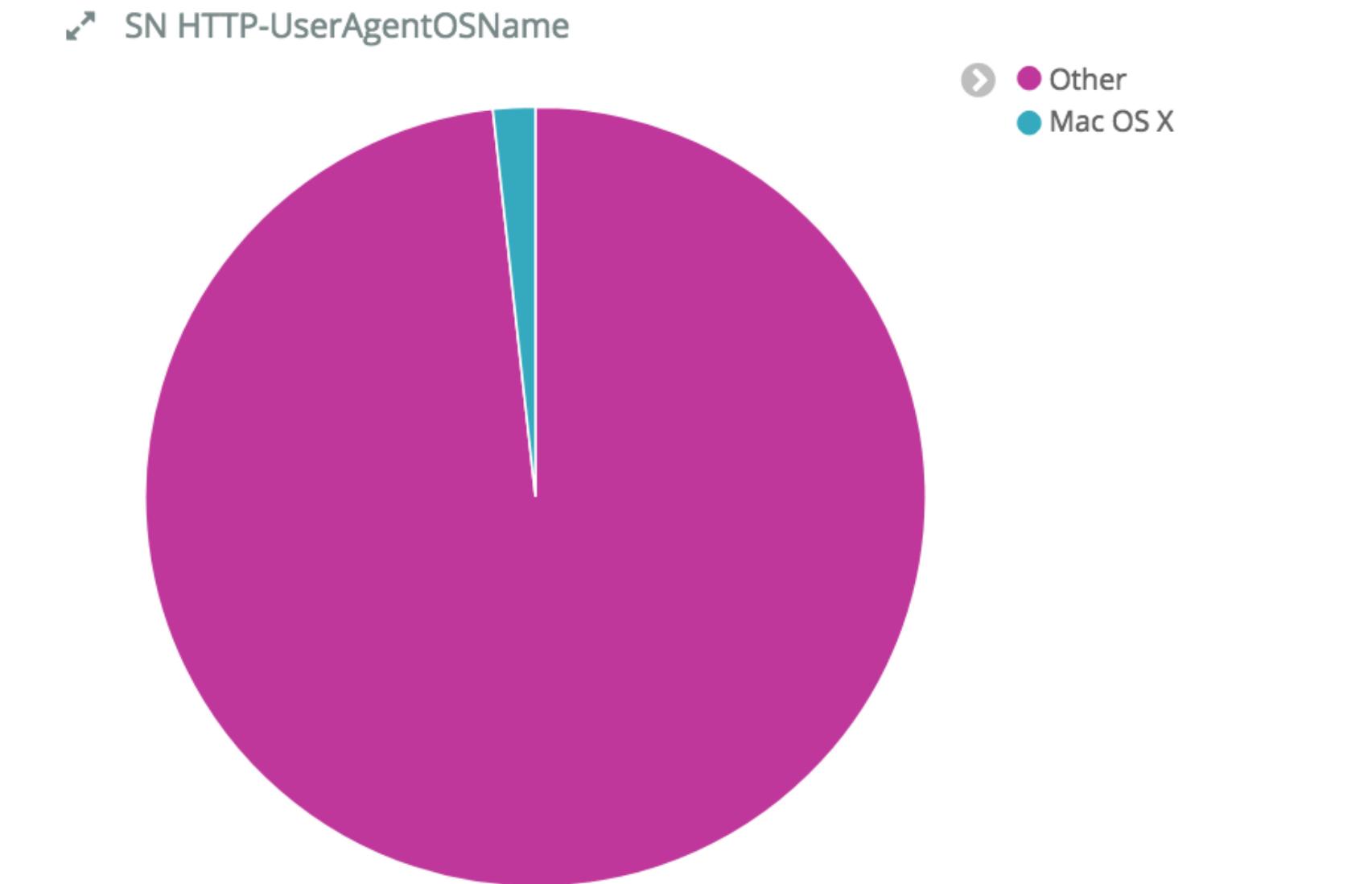
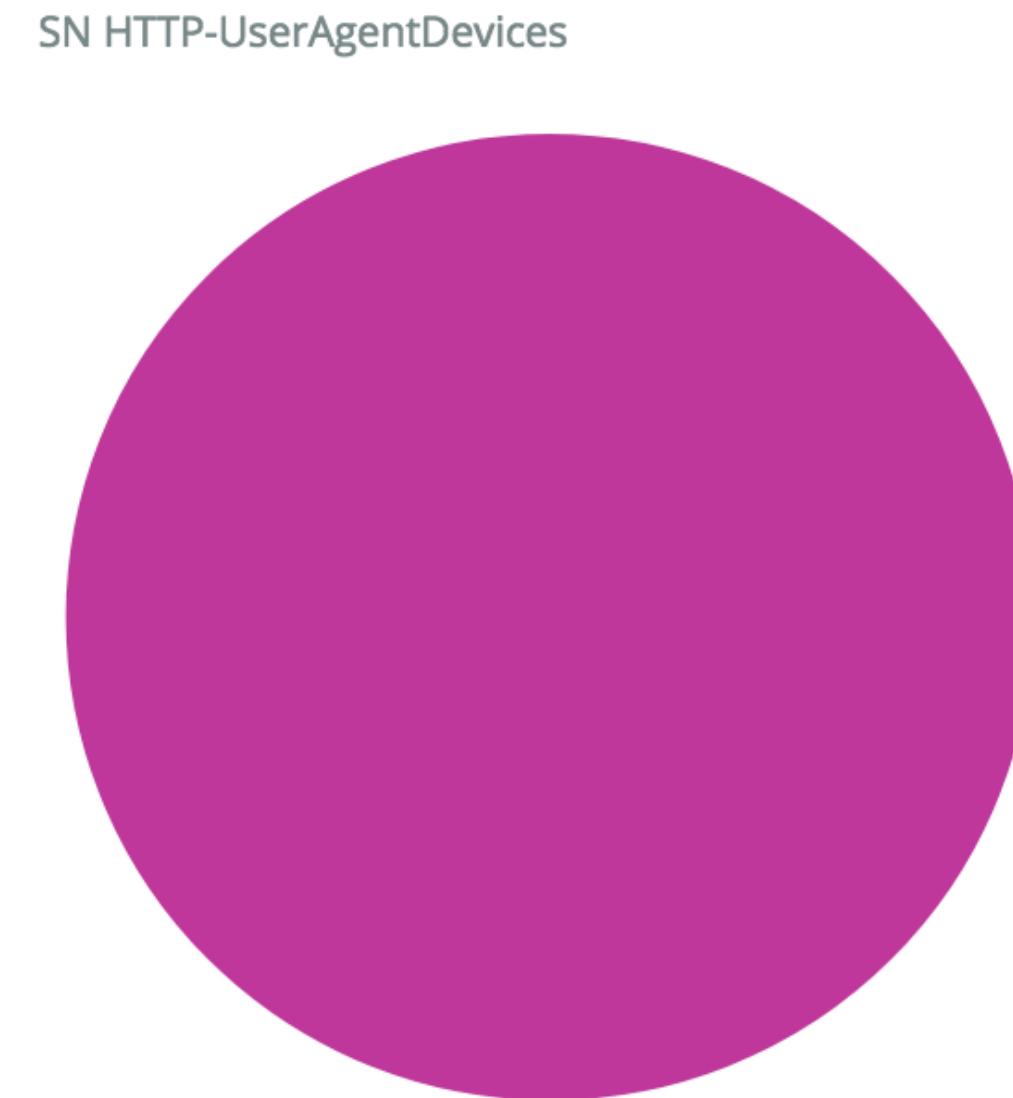
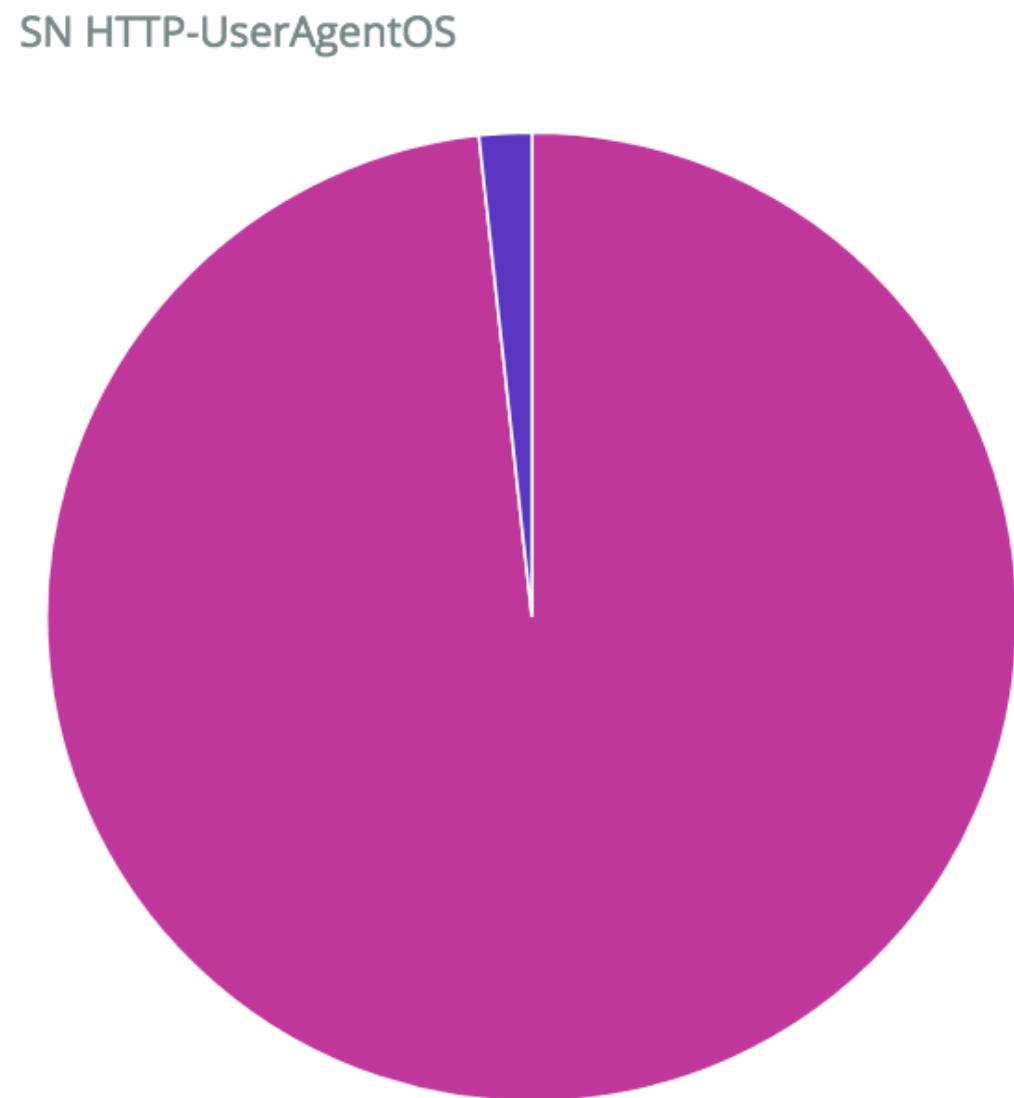
*Drilling in*



# Beaconing: Hunt



# Beaconing: Hunt



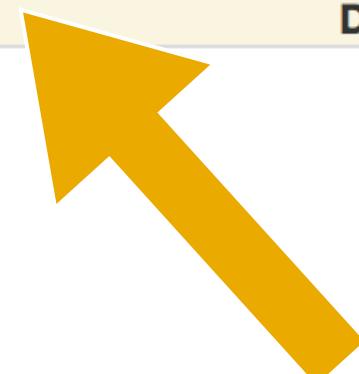
SN HTTP-EventsList

Time	EveBox	src_ip	src_port	proto	dest_ip	dest_port	http.http_method	http.hostname	http.status	http.protocol	http.server
▶ June 13th 2017, 10:49:01.935	<a href="#">Correlate Flow</a>	192.168.0.53	49182	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25
▶ June 13th 2017, 10:48:01.680	<a href="#">Correlate Flow</a>	192.168.0.53	49180	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25
▶ June 13th 2017, 10:47:01.388	<a href="#">Correlate Flow</a>	192.168.0.53	49178	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25
▶ June 13th 2017, 10:46:01.102	<a href="#">Correlate Flow</a>	192.168.0.53	49176	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25



# Beaconing: Hunt

EveBox	Inbox	Escalated	Alerts	Events	Reports ▾
721700456887339					
<button>Refresh</button> <button>Event Type: All ▾</button>					
Timestamp	Type	Source/Dest	Description		
2017-06-13 10:50:03 7 hours ago	FLOW	S: 192.168.0.53 D: 160.153.76.129	TCP 192.168.0.53:49182 -> 160.153.76.129:80; Age: 1; Bytes: 6223; Packets: 19		
2017-06-13 10:50:03 7 hours ago	FLOW	S: 192.168.0.53 D: 160.153.76.129	TCP 192.168.0.53:49182 -> 160.153.76.129:80; Age: 1; Bytes: 6223; Packets: 19		
2017-06-13 10:49:02 7 hours ago	FILEINFO	S: 160.153.76.129 D: 192.168.0.53	/ - Hostname: www.huntoperator.com; Content-Type: text/html		
2017-06-13 10:49:01 7 hours ago	HTTP	S: 192.168.0.53 D: 160.153.76.129	GET - www.huntoperator.com - /		
2017-06-13 10:49:01 7 hours ago	ALERT	S: 192.168.0.53 D: 160.153.76.129	ET POLICY curl User-Agent Outbound		



# Beaconing: Hunt

HTTP		
<p>Hostname: <a href="#">www.huntoperator.com</a> Http User Agent: <a href="#">curl/7.47.0</a> Status: <a href="#">503</a> User Agent.Name: <a href="#">Other</a></p>	<p>Http Content Type: <a href="#">text/html</a> Length: <a href="#">1093</a> Url: <a href="#">/</a> User Agent.Os: <a href="#">Other</a></p>	<p>Http Method: <a href="#">GET</a> Protocol: <a href="#">HTTP/1.1</a> User Agent.Device: <a href="#">Other</a> User Agent.Os Name: <a href="#">Other</a></p>
GeoIP		
<p>City Name: <a href="#">Scottsdale</a> Coordinates.1: <a href="#">33.6119</a> Country Name: <a href="#">United States</a> Latitude: <a href="#">33.6119</a> Longitude: <a href="#">-111.8906</a> Region Name: <a href="#">Arizona</a></p>	<p>Continent Code: <a href="#">NA</a> Country Code2: <a href="#">US</a> Dma Code: <a href="#">753</a> Location.0: <a href="#">-111.8906</a> Postal Code: <a href="#">85260</a> Timezone: <a href="#">America/Phoenix</a></p>	<p>Coordinates.0: <a href="#">-111.8906</a> Country Code3: <a href="#">US</a> Ip: <a href="#">160.153.76.129</a> Location.1: <a href="#">33.6119</a> Region Code: <a href="#">AZ</a></p>
Flow - TCP 192.168.0.53 -> 160.153.76.129		[ Open ]
<p>Age: <a href="#">1</a> Bytes Toserver: <a href="#">752</a> Pkts Toserver: <a href="#">10</a> State: <a href="#">closed</a></p>	<p>Alerted: <a href="#">true</a> End: <a href="#">2017-06-13T10:49:02.275746-0400</a> Reason: <a href="#">timeout</a></p>	<p>Bytes Toclient: <a href="#">5471</a> Pkts Toclient: <a href="#">9</a> Start: <a href="#">2017-06-13T10:49:01.717867-0400</a></p>
Payload		[ PCAP ]
<pre>GET / HTTP/1.1 Host: www.huntoperator.com User-Agent: curl/7.47.0 Accept: */*</pre>	<pre>47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 68 75 6e 74 6f 70 65 72 61 74 6f 72 2e 63 6f 6d 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f 37 2e 34 37 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 0d 0a</pre>	

# Beaconing: Hunt

## Report for IP 160.153.76.129

[Related Reports ▾](#)

Refresh All Sensors Filter... Apply Clear

Alerts Over Time

DNS Hostnames Returning 160.153.76.129

#	Hostname
1806	<a href="#">huntoperator.com</a>

DNS: Top Requested Hostnames

No data.

Outgoing HTTP User Agents

No data.

HTTP: Incoming HTTP Request Hostnames

#	Hostnames
903	<a href="#">www.huntoperator.com</a>

Alerts: Top Alerts

#	Signature
890	ET POLICY curl User-Agent Outbound

Flow

Flows As Client	0
Flows As Server	1794
Bytes To...	1.23 MB
Bytes From...	12.70 MB
Packets To...	16k (16908)
Packets From...	19k (19084)

Incoming TLS Server Names (SNI)

No data.

TLS Versions as Client

No data.

TLS Versions as Server

No data.

HTTP: Top Requested Hostnames

No data.

CASE SOLVED

# **Domain Generation Algorithms (DGA)**

# Domain Generation Algorithms (DGA)

## Why DGA?

- Deterministic value
- Generate large number of domain names
- Easy to burn
- Cheap to register
- Used as a rendezvous point by attacker

vtlfccmfxlkgifuf.com

Yes! Your domain is available. Buy it before someone else does.

vtlfccmfxlkgifuf.com

\$14.99\* \$12.99\*

Add to Cart

vtlfccmfxlkgifuf.us Add this: \$1.00

when you register for 2 years or more. 1st year price \$1.00 Additional years \$19.99

! Get 3 and Save 69%

vtlfccmfxlkgifuf.net

vtlfccmfxlkgifuf.org

vtlfccmfxlkgifuf.info

\$57.97\* \$18.00\*

Add to Cart

In [18]:

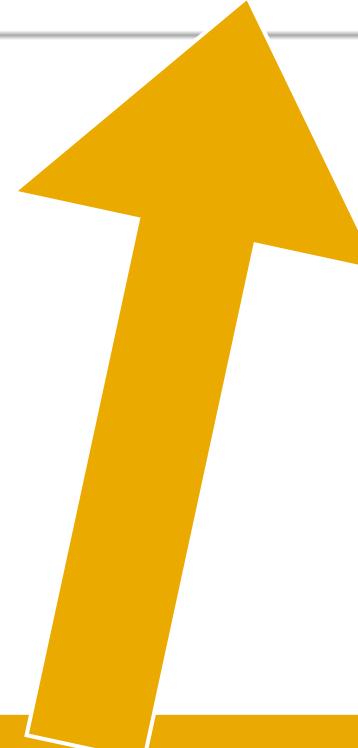
```
1 def generate_domain(year, month, day):
2     """Generates a domain name for the given date."""
3     domain = ""
4
5     for i in range(16):
6         year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFFF0) << 17)
7         month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
8         day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFF8) << 12)
9         domain += chr(((year ^ month ^ day) % 25) + 97)
10
11     return domain + '.com'
```

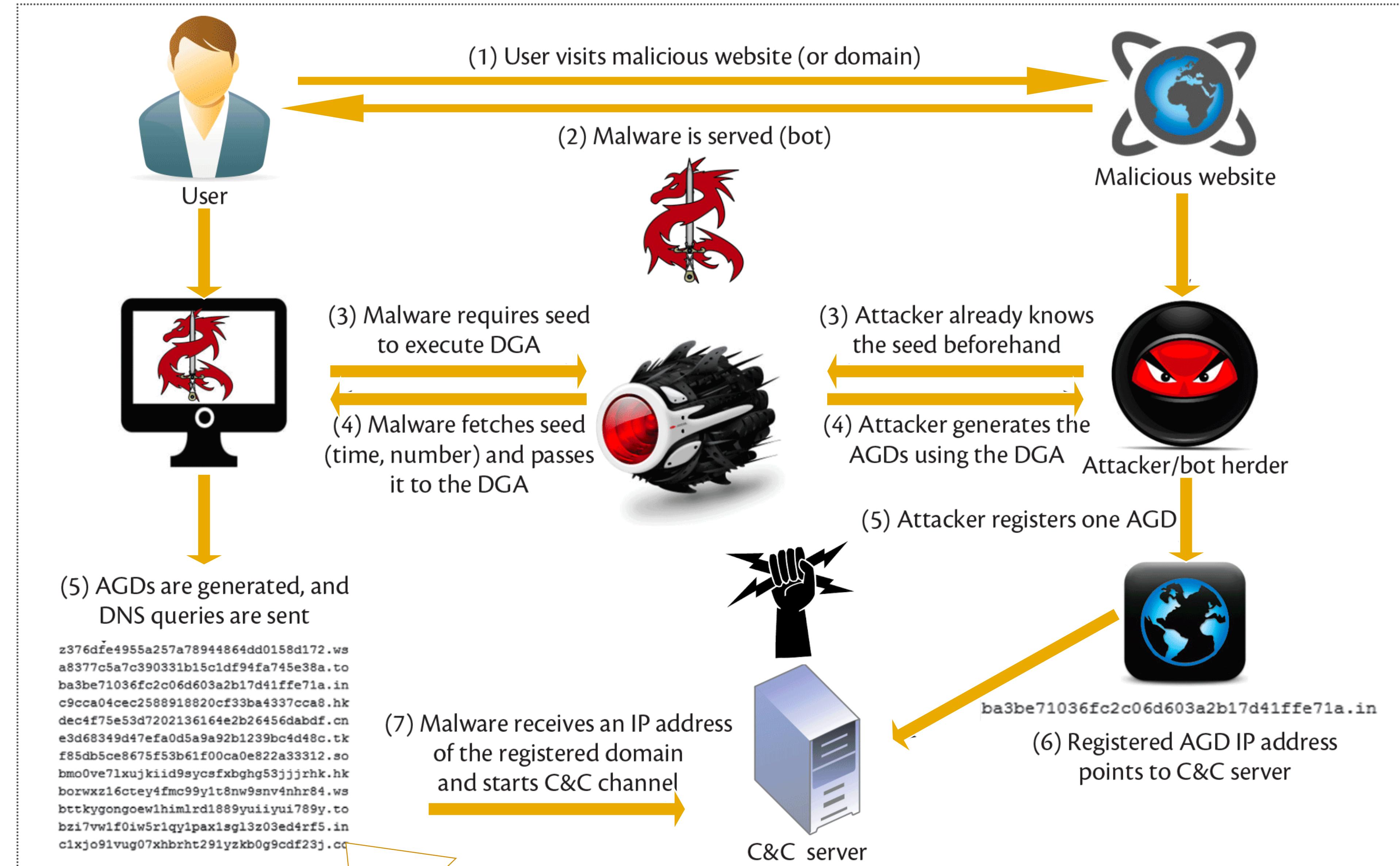
In [19]:

```
1 generate_domain(2017, 6, 23)
```

Out[19]:

```
'vtlfccmfxlkgifuf.com'
```





Source: Aditya K. Sood, Sheralli Zeadally, "A Taxonomy of Autonomous Generation Algorithms", IEEE Security & Privacy, vol. 14, no. , pp. 46-53, July-Aug. 2016, doi:10.1109/MSP.2016.76

GTK Cyber  
We want to detect this



I'M GONNA HAVE TO DATA

SCIENCE THE SH\*T OUT OF THIS

imgflip.com

GTK Cyber

# **Scenario 2**

# Scenario 2

- A piece of malware has infected a computer on your network and is making requests to domains using DGA in an attempt to communicate to a Command and Control Server



GTK Cyber

# DNS Records

Record Count: 15408

```
In [62]: 1 dns_records = pd.read_csv('/Users/huntoperator/Downloads/exported_dns_n
```

```
In [63]: 1 dns_records|
```

Out[63]:

	dns_rrname	count
0	daisy.ubuntu.com	300,129
1	srv.myskybell.com	55,053
2	googleapis.l.google.com	37,005
3	ubuntu.com	30,549
4	www.google.com	28,780
5	www.example.org	26,367
6	www.example.com	26,354
7	www.example.net	26,298
8	myskybell.com	18,296

# Import Flare Tools

## ■ DGA Classifier

- Random Forrest Classifier

- N-Grams

- Uses labelled data

## ■ Alexa - Top 1M most popular visited websites

- Must pay for service now.

- Umbrella/Majestic are free alternatives

## ■ Domain TLD Extract - Extracts the Top Level Domain to be checked against Alexa

- Also calculate degree from here

```
In [64]: 1 from flare.data_science.features import dga_classifier
```

```
In [66]: 1 from flare.tools.alexa import Alexa  
2 from flare.data_science.features import domain_tld_extract
```

```
In [65]: 1 dga_c = dga_classifier()
```

```
[*] Initializing... training classifier - Please wait.  
[+] Classifier Ready
```

# Filter Results

```
In [73]: 1 dns_records['domain_tld'] = dns_records.dns_rrname.apply(lambda x: domain_tld_extract(str(x)))
```

```
In [78]: 1 dns_records['dga_predict'] = dns_records.domain_tld.apply(lambda x: dga_c.predict(x))
```

```
In [129]: 1 dns_records.head()
```

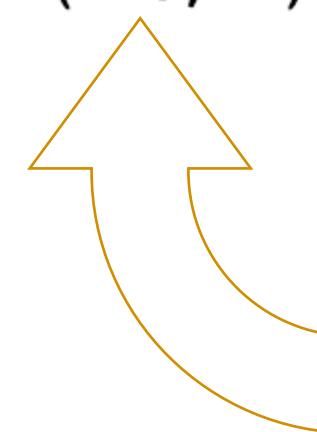
Out[129]:

	dns_rrname	count	domain_tld	dga_predict
0	daisy.ubuntu.com	300,129	ubuntu.com	legit
1	srv.myskybell.com	55,053	myskybell.com	legit
2	googleapis.l.google.com	37,005	google.com	legit
3	ubuntu.com	30,549	ubuntu.com	legit
4	www.google.com	28,780	google.com	legit

```
In [84]: 1 dns_filter = dns_records[dns_records.dga_predict=='dga']
```

```
In [131]: 1 dns_filter.shape
```

Out[131]: (240, 4)



still too many results...

# Filter Results

```
1 beacon_df.groupby(self.beacon_dest_ip)[self.beacon_dest_ip].transform('count').fillna(0).astype(int)
```

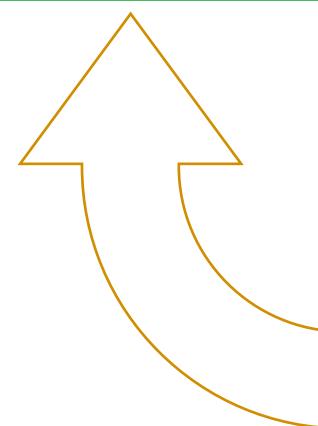
```
1 dns_filter1['domain_degree'] = dns_filter1.groupby('domain_tld')['domain_tld'].transform('count').fillna(0).
```

and yet...

```
In [103]: 1 dns_filter1 = dns_filter[dns_filter['count'] < 10]
```

```
In [105]: 1 dns_filter1.shape
```

```
Out[105]: (162, 4)
```



Still too many results...

# Filter Results

```
In [117]: 1 alexa = Alexa()  
  
In [123]: 1 dns_filter2 = dns_filter1[dns_filter1['domain_degree'] < 2]  
  
In [119]: 1 dns_filter2.shape  
Out[119]: (78, 5)    down to 78  
  
In [125]: 1 dns_filter2['in_alexa'] = dns_filter2.domain_tld.apply(lambda x: alexa.domain_in_alexa(x))
```

And finally...

# Filter Results

apply Alexa top 1 million check...

```
In [127]: 1 dns_filter2[dns_filter2['in_alexa']==False]
```

Out[127]:

	dns_rrname	count	domain_tld	dga_predict	domain_degree	in_alexa
5421	version.mos.svc.ovi.com.glb.as1248.net	9	as1248.net	dga	1	False
5765	ebdr3.com	8	ebdr3.com	dga	1	False
5847	i.s-jcrew.com	8	s-jcrew.com	dga	1	False
5885	in.ml314.com	8	ml314.com	dga	1	False
6382	vtlfccmfxlkgifuf.com	8	vtlfccmfxlkgifuf.com	dga	1	False
7510	tags.wdsvc.net	6	wdsvc.net	dga	1	False
7766	get35.com	5	get35.com	dga	1	False
7920	x64dbg.com	5	x64dbg.com	dga	1	False

and...

```
In [135]: 1 dns_filter2[dns_filter2['in_alexa']==False].shape
```

Out[135]: (57, 6)

57 Results!

# Pass to Analyst

## ■ Identify Process Generating Traffic

### ■ Isolate infected host

Back

### ■ Begin endpoint investigation...

DNS: ANSWER: NXDOMAIN for vtlfccmfxlkjifuf.com

Timestamp 2017-06-23T09:24:33.531143-0400  
Protocol UDP  
Source 2001:558:feed::1:53  
Destination 2601:154:c300:47a0:2173:e85a:fdd6:c224:62530  
In Interface eth0  
Flow ID 1806004061843760

Type answer  
ID 51938  
RCode NXDOMAIN  
RRName vtlfccmfxlkjifuf.com  
RRTYPE  
RData

GeolP

Continent Code: NA

Coordinates: 0: -98.5

GTK Cyber

# Natural Language Processing

# Natural Language Processing Use Case

Challenge: Distinguishing between base64 and URLs

String 1: Q09NRSBUTyBNWSBQWVRIT04gQ0xBU1MgSU4gVkVHQVMhISEh

String 2: forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893

Both of these are properly formatted BASE64 encoded strings,  
but only one is intentional

# Natural Language Processing

Solution: Mark Baggett's freq.py

Produces a **probability score** based off frequency tables in:

- ▲ • Dictionary
- ▼ • Your environment

Higher likelihood scores indicate common

Lower likelihood scores indicate not common

# Freq.py

How does it work?

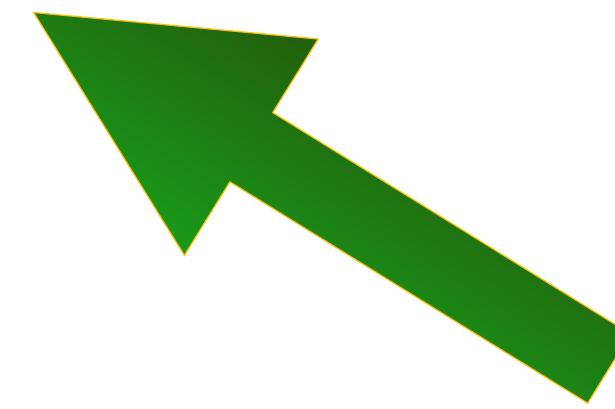
- Identifies likelihood of character occurrence based on frequency analysis

Example: In English text, “m” is usually followed by an “o,” so seeing a “m” followed by something else would be rather unlikely to occur

Similar to high order markov chain

# Using Freq.py

```
>>> from freq import *
>>> fc = FreqCounter()
>>> fc.load("english_lowercase.freq")
```



Pre-built frequency table

# Using Freq.py

How did we score?

```
>>> fc.probability('forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893')  
9.490788394012279
```

```
>>> fc.probability('Q09NRSBUTyBNWSBQWVRIT04gQ0xBU1MgSU4gVkVHQVMhISEh')  
2.578357325221811
```

# Freq-ing awesome...

Build Separate frequency tables for:

- DNS Host names that are normally seen for your environment
- Names of files on your file server
- Host names inside of SSL certificates
- URLs for a specific Web Application
- \* Insert any string you want to measure here \*

*Runs as a restful API and integrates with logstash*

<https://isc.sans.edu/forums/diary/Continuous+Monitoring+for+Random+Strings/20451/>

# Freq.py - Operational Examples

DNS - Parent Domain Frequency Analysis	
Domain	Frequency Score
tahmqkmvzg.com	0.434
dsms0mj1bbhn4.cloudfront.net	1.085
oiugixyilxy.com	1.683
yfksjfsunyiypu.com	2.053
fwhynaxzwkv.com	2.329
yahoodns.net	2.697
muihoc.com	2.914
qfrwozmoaqc.com	3.153
aemmlphbweeuef59.com	3.345
brovztkzbl.com	3.391

SSL - Certificate Common Name Frequency Analysis	
Common Name	Frequency Score
d3fr3g4tgwf	2.781
www.spidh.org	3.425
imap.gmx.net	3.96
www.lilawelt.net	4.538
www.paypal.com	5.324
www.google.com	5.454
www.jeffbryner.com	6.144
*.s3.amazonaws.com	6.185
s3.amazonaws.com	6.185
plesk3.gigaspark.com	6.458

Logstash configs created by Justin Henderson (@SecurityMapper)

# Thank You!