Big Data Analysis

Lec01. Data Processing with Python

About this lecture

- Unit Logistics
  - Scope
  - Schedule
  - Reading Materials
- Introduction to Big Data Analysis
- Programming environment:
  - Anaconda
  - Jupyter notebook
  - Python basics
  - Popular Python packages: pandas, scikit-learn, etc.

Why 3803ICT/3030ICT/7130ICT?

- Big Data Analysis
  - Focus on end-to-end process of data science
  - Cover core concepts, important technologies
- This is an advanced unit
- Related units:
  - 2802ICT Intelligent Systems:
    - Introduce basic machine learning algorithms
  - 2030ICT/7030ICT Introduction to Big Data Analysis
  - 4030ICT/7230ICT Big Data Analysis and Social media
- Sibling 3804ICT/7031ICT/3031ICT Data Mining

Please check the course profile of your cohort for correct requirements

About 3803ICT/3030ICT/7130ICT

- This unit emphasizes on practical and applied aspect of data science
- You will learn core concepts and how to design a data analytics application pipeline
- You will learn some of the most important steps in data science pipeline: data preparation, data preprocessing, data analytics, data visualization.
- This course is NOT about:
  - Teaching you how to do programming
    - If you haven't programmed Python before, then this unit is probably NOT for you!
  - Teaching you advanced statistics
- It is not only practical, industry-relevant unit, but also challenging with theoretical concepts.
- Requires intermediate programming skills

Course structure

W1. Data Processing with Python

W2. Data Exploration with Python

W3. Data Modeling with Python

W4. Data Analytics for Timeseries

W5. Holiday

W6-7. Data Analytics for Texts

W8. Data Analytics for Images

W9. Data Analytics for Graphs

W10-11. Data Analytics for Other Data

W12. Revision

Make sure you read the course profile

Unit Logistics

- Class sessions:
  - 10 lectures (2 hours each)
  - 10 practical lab sessions (2 hours each)
- Assessment components:
  - 10 weekly workshops are graded: 20% (2% each)
    - 1 week to complete (due to next-week Sunday 23:59)
  - 1 Assignment: 35%, due Sunday, week 11 (23:59)
    - Group submission (2 persons/group)
    - Similar content to the labs
  - 1 final exam: 45%
    - 2 hours examination, **date TBA**

Unit Logistics: Reading Materials

- Recommended books:
  - **Build a Career in Data Science.** Emily Robinson. Manning. 2020
  - **Python for Data Analysis**. Wes McKinney. O'Reilly Media. 2017.
  - **Learning Spark: Lightning-Fast Data Analytics**. Jules Damji. O'Reilly Media. 2020
  - **Machine Learning for Time-Series with Python**. Ben Auffarth. Packt Publishing. 2021
  - **Network Science**. Albert-László Barabási. 2016.
  - **Deep Learning for Computer Vision with Python**. Adrian Rosebrock. 2017.
  - **Deep Learning for NLP and Speech Recognition**. Uday Kamath. 2019
- Programming environment:
  - Python, Jupyter notebook
  - Python packages: pandas, scikit-learn, etc.

Unit logistics: time and location

Lecture

- Time: **Mon 8:00 – 9:50**
- Location: **online**

Practical Lab Sessions

- **3803ICT**
  - GC campus: Mon 10AM-11:50AM or 2PM - 3:50PM at G23_2.27
  - Online: Mon 2PM – 3:50PM or Tue 10AM – 11:50AM
- **3030/7130ICT**
  - GC campus: Mon 2PM - 3:50PM G31_3.14 or **Fri 11:00AM – 12:50PM G31_3.14**
  - NA campus: Mon 10AM – 11:50AM N79_4.10 or Mon 2PM-3:50PM at N79_4.10
  - Online: Mon 2PM - 3:50PM or Tue 10AM - 11:50AM

Unit Logistics: contact and communication

- Contact time for each week
  - Two hours for lecture time
  - Two hours for practical sessions
- Communication
  - Please use email only when outside allocated contact time
  - Face-to-face meeting upon request

Exercises and projects: GitHub TODO

- De-factor standard for managing and sharing code
- All students in this class need a Github account
- Submit your labs via Github
- Submit your project via Canvas
- Register it here: https://forms.office.com/r/Yb6BtX0KYC

This Week Content

I. Data Science and Applications
II. Data Storage with Python
III. Data Cleaning with Python

I. Data Science, Applications, and Tools

What is Data Science

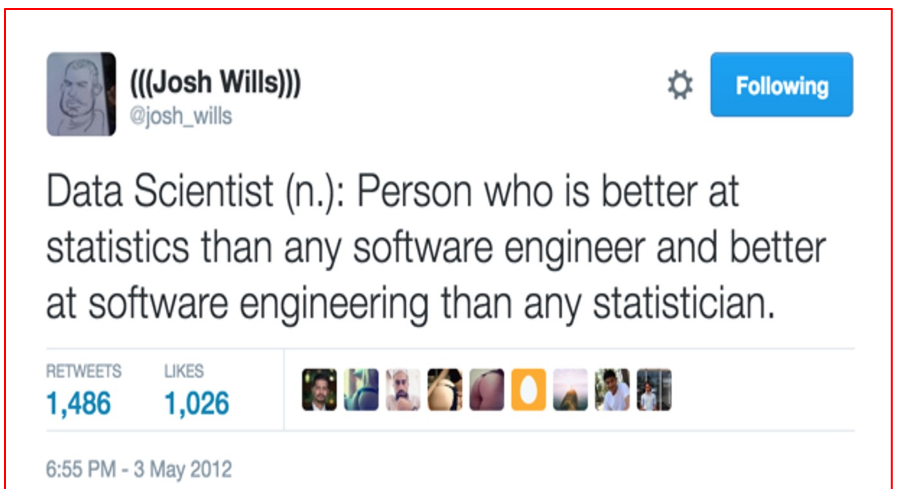
There isn't a definition agreed by all yet!

Wikipedia	"Data Science is the extraction of knowledge from large volumes of data that are structured or unstructured"
NIST, 2015	**"Data science is the empirical synthesis of actionable knowledge from raw data through the data lifecycle process"**
Dhar, 2013	"Data science is the study of generalizable knowledge from data"
Peter Naur, 1974	"[data science is] The science of dealing with data, once they have been established, while the relation of the data to what they represent is delegated to other fields and sciences."

Data Scientists

"A data scientist is someone who can obtain, scrub, explore, model, and interpret data, blending hacking, statistics, and machine learning. Data scientists not only are adept at working with data, but appreciate data itself as a first-class product."

Hilary Mason, chief scientist at bit.ly



Josh Wills, Data Scientist at Slack

Data Scientists in Job Markets

- Salary: very competitive payroll

https://au.indeed.com/jobs?q=data+scientist&l=

- Trend:
  - Businesses Will Need One Million Data Scientists by 2018

https://www.kdnuggets.com/2016/01/businesses-need-one-million-data-scientists-2018.html

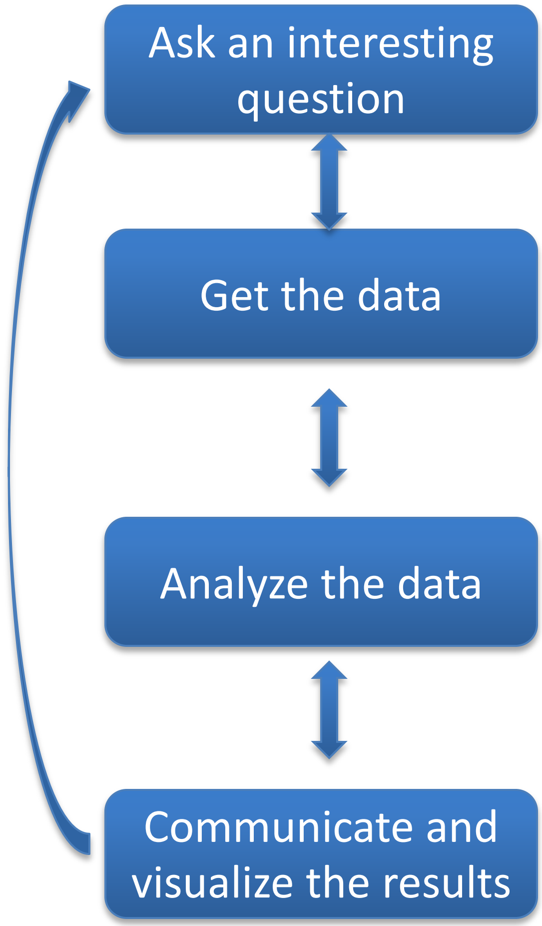
Data Science pipeline

1. Ask an interesting question:

- ➢ What is the goal?
- ➢ What would you do if you had all the data?
- ➢ What do you want to predict or estimate?

2. Get the data:

- ➢ How were the data sampled?
- ➢ Which data are relevant?
- ➢ Are there privacy issues?



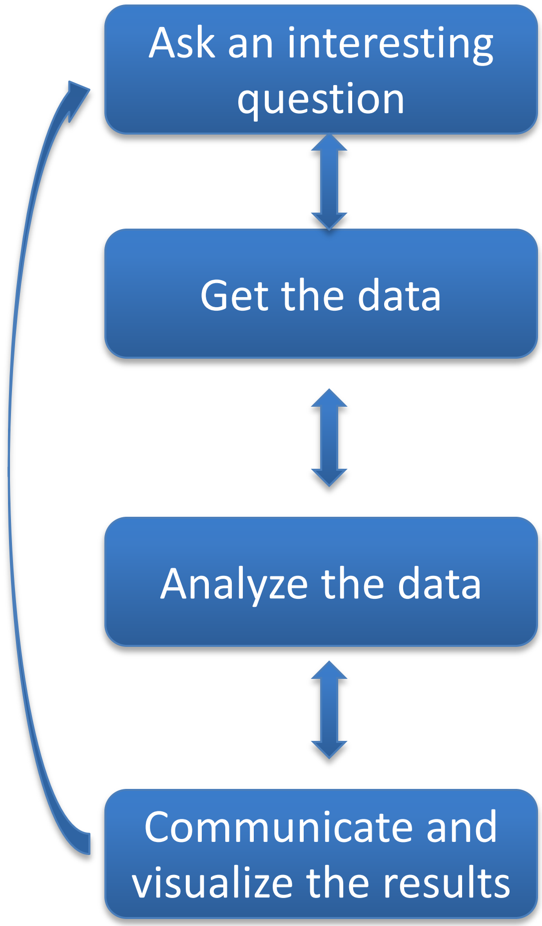
Data Science pipeline

3. Analyze the data:

- Are there anomalies?
- Are there patterns?
- Are there trends?

4. Communicate and visualize the results

- What did we learn?
- Do the results make sense?
- Can we tell a story?



Tools

1. Anaconda
2. Jupyter Notebook
3. Python Basics
4. Python libraries for data analytics

Anaconda: our Python environment

- Will be used in lab sessions
- You can do it on your laptop:
  - https://www.anaconda.com/download
  - Version: preferably Python 3.6

  (The workshops were tested on Python 3.6 but we will test it again for Python 3.7)

Install Anaconda

- This course uses Python
  - Make sure you have a Python development environment set up on your personal computer (we will do it for you in laboratory computers).
  - We will walk through installing a package called Anaconda which has both the development environment and all the Python packages you need pre-installed. It makes life really easy.
- **Anaconda Distribution by Continuum Analytics:**
  1. Go to the website: https://www.anaconda.com/download/
  2. Download the installer (preferably Python 3.6 version) for your OS
  3. Run the installer

Jupyter Notebook

- Jupyter Notebook is a web application for interactive data science
- Create documents that combine live-code with narrative text, equations, images, videos, and visualizations.
- Reproducible record of computations to share on GitHub, Dropbox, and Jupyter Notebook Viewer.
- Shareable: can be exported to PDF, HTML, etc.
- Interactive Widgets: code can produce rich output such as images, videos, LaTeX, and Javascript. Interactive widgets can be used to manipulate and visualize data in realtime.

Python Basics

- We will revisit basics of Python programming
- This course is meant for students with some programming experience
- If you are completely new to programming, this course may be **too advanced** for you

Python Basics: Topics Covered

- Data Types
  - Numbers
  - Strings
  - Print
  - Formatting
  - Lists
  - Dictionaries
  - Booleans
  - Tuples and Sets
- Comparison Operators
- If, elif, and else Statements
- For Loops
- While Loops
- range()
- List Comprehension
- Functions
- Lambda Expressions
- Map and Filter

Python libraries for Data Analytics

https://seaborn.pydata.org/

scikits learn

machine learning in Python

matplotlib

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

- Numpy: great for handling numbers, vectors, matrices
- Scipy: great for numerical optimizations
- Pandas: great for handling tabular/relational data
- Scikit Learn: great for data analytics techniques

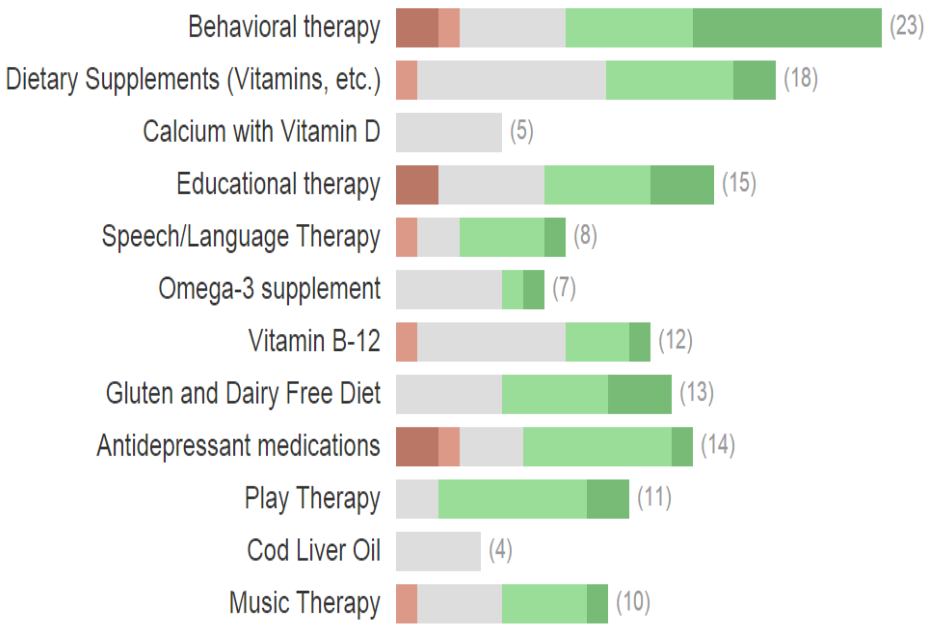
Application: collective intelligence

- Patients come together to share quantitative medical data
  - Patients vote on effectiveness of each treatment
  - Use data from the crowd to derive new insights

CureTogether

curetogether.com

"autism"

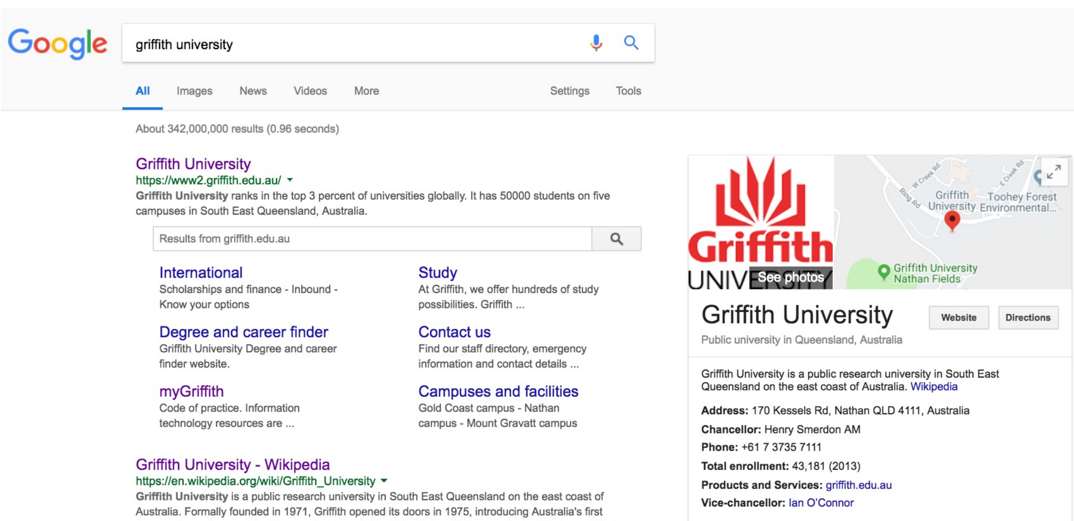


Application: product recommendation

- Amazon uses data to build recommender system
  - Apply data analytics to understand customer behaviours
  - Recommend similar products from users with similar behaviours

Application: knowledge base

❖ Apply data analytics to derive new knowledge



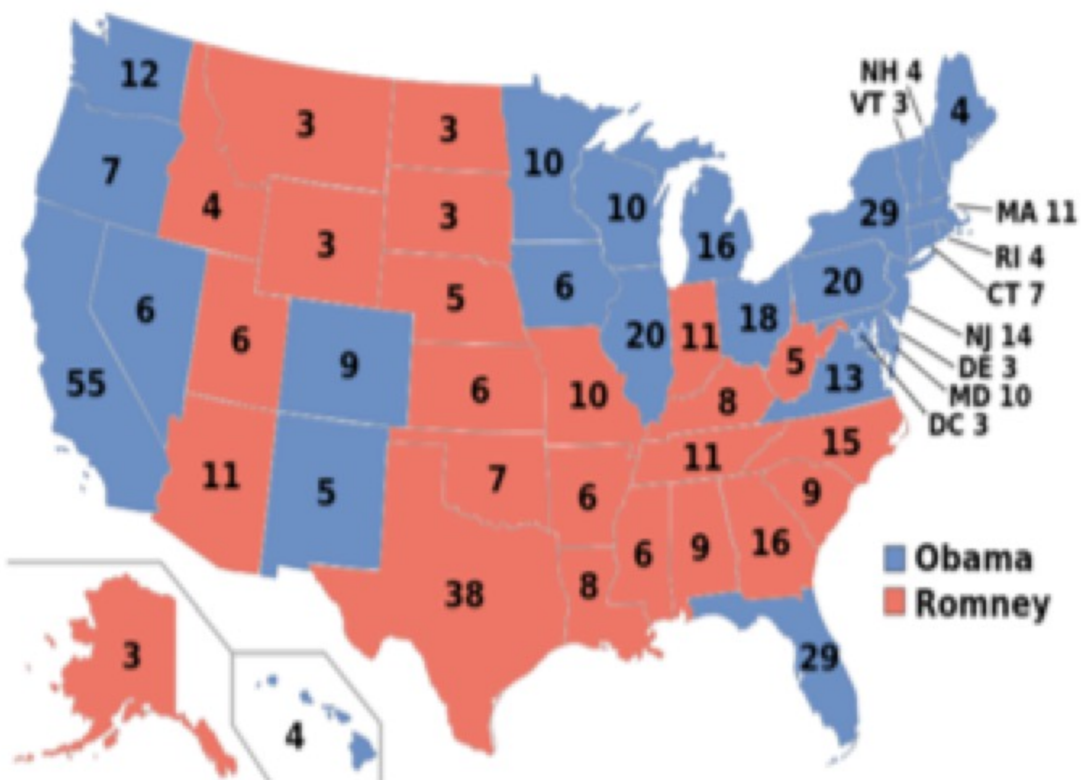
Google Knowledge Graph: provide description of all entities such as people, places, and organizations

IBM Watson – Q&A service: automatically answer a textual question

Application: predict elections

Silver, who made his name by using cold hard math (historical data) to predict elections correctly in 49 out of 50 states in the 2008 and all 50 states in 2012

http://www.slate.com/articles/news_and_politics/politics/2016/01/nate_silver_said_donald_trump_had_no_shot_where_did_he_go_wrong.html

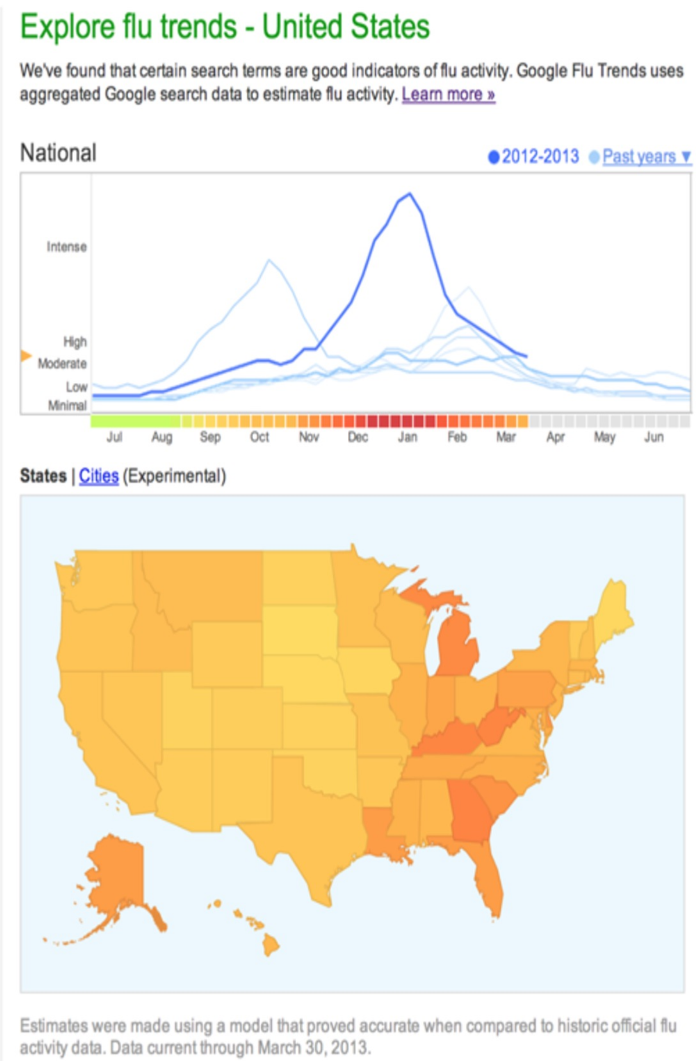


https://en.wikipedia.org/wiki/United_States_presidential_election,_2012

Application: Flu Monitoring

Google Flu Trend: provide estimates of influenza activity for more than 25 countries

- Use the search queries related to flu on Google
- Users tend to search information for potential flu outbreaks
- Predict flu at a location based on the number of queries and their IP location



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

II. Data Storage with Pandas

Data storage: Pandas library

- Pandas is an open source library
  - Process relational data in memory
  - Support SQL-like query
- Rich relation data tool built on top of NumPy
  - Excellent performance
  - Easy-to-use, highly consistent API
- A foundation for data analysis in Python
  - It also has built-in visualization features
  - It can work with data from a wide variety of sources
  - Takes data preparation and preprocessing to the next level

https://pandas.pydata.org/

Pandas vs. SQL implementations

Advantages:

+ Pandas is lightweight and fast.

+ Natively Python, i.e., full SQL expressiveness plus the expressiveness of Python, especially for function evaluation.

+ Integration with plotting functions like Matplotlib.

Disadvantages:

- Tables must fit into memory.

- No post-load indexing functionality: indices are built when a table is created.

- No transactions, journaling, etc.

- Large, complex joins are slower.

Pandas: features

- Series
- DataFrames → Tables
- Operations
- Data Input and Outputs
- …

Pandas: Series

- Series: a named, ordered dictionary
  - Easy data search and retrieval.

- One-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.).
  - The keys of the dictionary are the indexes
  - Built on NumPy's **ndarray**
  - Values can be any NumPy data type object

- Why use Series?
  - Better for single-valued entries of the same type (Integer, String, etc.).
  - Easy and efficient search using index.

Pandas: Series

❖ Create a series:

```
ser1 = pd.Series([1,2,3,4],index = ['USA', 'Germany','USSR', 'Japan'])
```

```
ser1
```

```
USA        1
Germany    2
USSR       3
Japan      4
dtype: int64
```

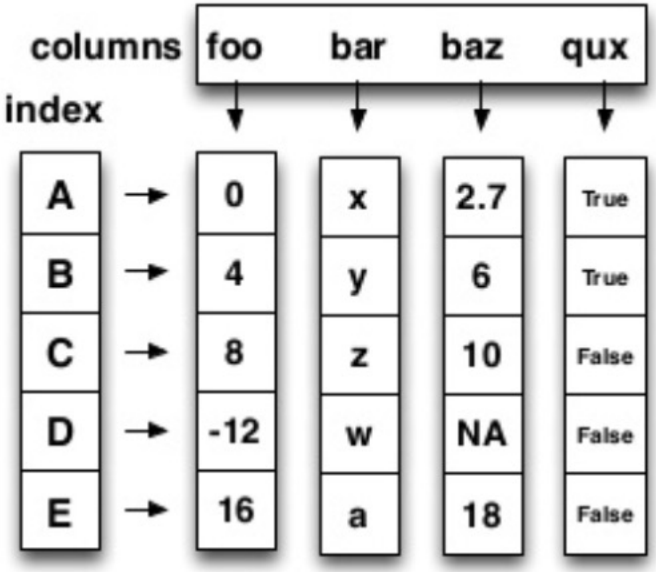
```
ser2 = pd.Series([1,2,5,4],index = ['USA', 'Germany','Italy', 'Japan'])
```

```
ser2
```

```
USA        1
Germany    2
Italy      5
Japan      4
dtype: int64
```

Pandas: DataFrame

- DataFrame: a table with named columns (like in the relational model)
  - Represented as a dictionary
    - columnName -> series
  - Each Series object represents a column
  - Each column can have a different type
  - Row and column indices
  - Size mutable: insert and delete columns



- Why Use DataFrames?
  - better for series with multiples attributes of different types.
  - Easy and efficient search elements by index.

Pandas: DataFrame

❖ Create a data frame:

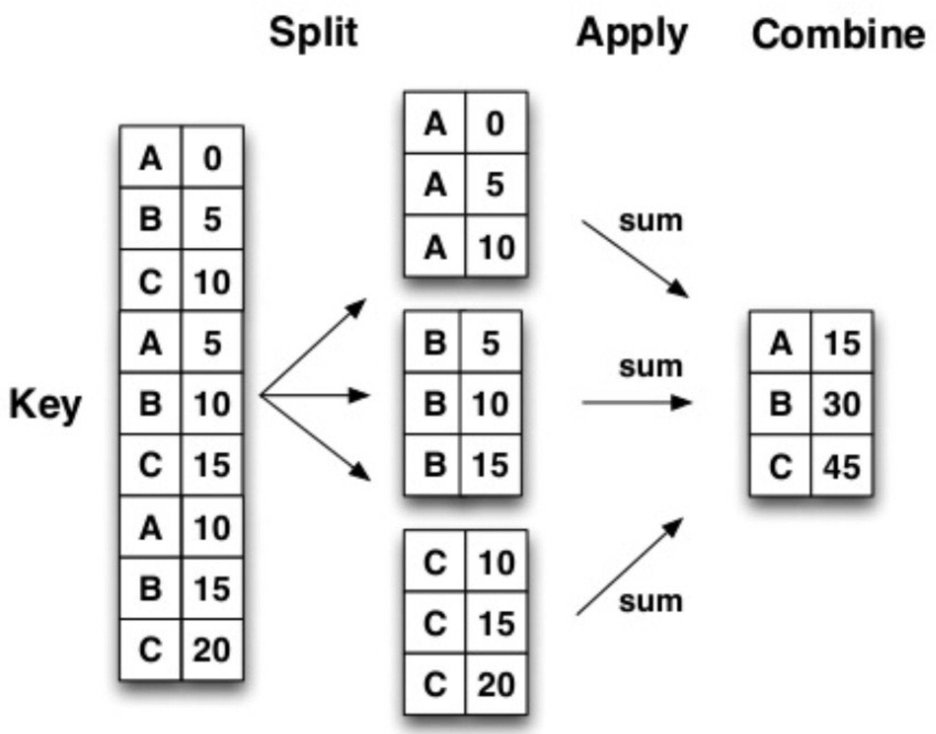
```
df = pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split())
```

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

Pandas: Groupby

- **GroupBy** allows you to group together rows based on a column and perform an aggregate function on them.
  - Ex: sum students by grading (HD, D, C, P, F).



Pandas: Data Input and Output

- Raw data is organized in different structures for the purpose of distribution and processing.
- How to read raw data into Pandas:
  - CSV
  - Excel
  - HTML
  - …
- Installation (using Anaconda Prompt):

```
conda install sqlalchemy
conda install lxml
conda install html5lib
conda install BeautifulSoup4
```

CSV

- Example:

CSV Input

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

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

CSV Output

```
df.to_csv('example.csv',index=False)
```

Excel

- Pandas can read and write excel files, keep in mind, this only imports data. Not formulas or images, having images or macros may cause this read_excel method to crash.

Excel Input

```
pd.read_excel('Excel_Sample.xlsx',sheetname='Sheet1')
```

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Excel Output

```
df.to_excel('Excel_Sample.xlsx',sheet_name='Sheet1')
```

HTML

- You may need to install `htmllib5`, `lxml`, and `BeautifulSoup4`. In your terminal/command prompt run:

```
conda install lxml
conda install html5lib
conda install BeautifulSoup4
```

- Then restart Jupyter Notebook.

- Pandas can read table tabs off of html. For example:

HTML

- Example: Pandas `read_html` function will read tables off of a webpage and return a list of DataFrame objects.

```
df = pd.read_html('http://www.fdic.gov/bank/individual/failed/banklist.html')
```

```
df[0]
```

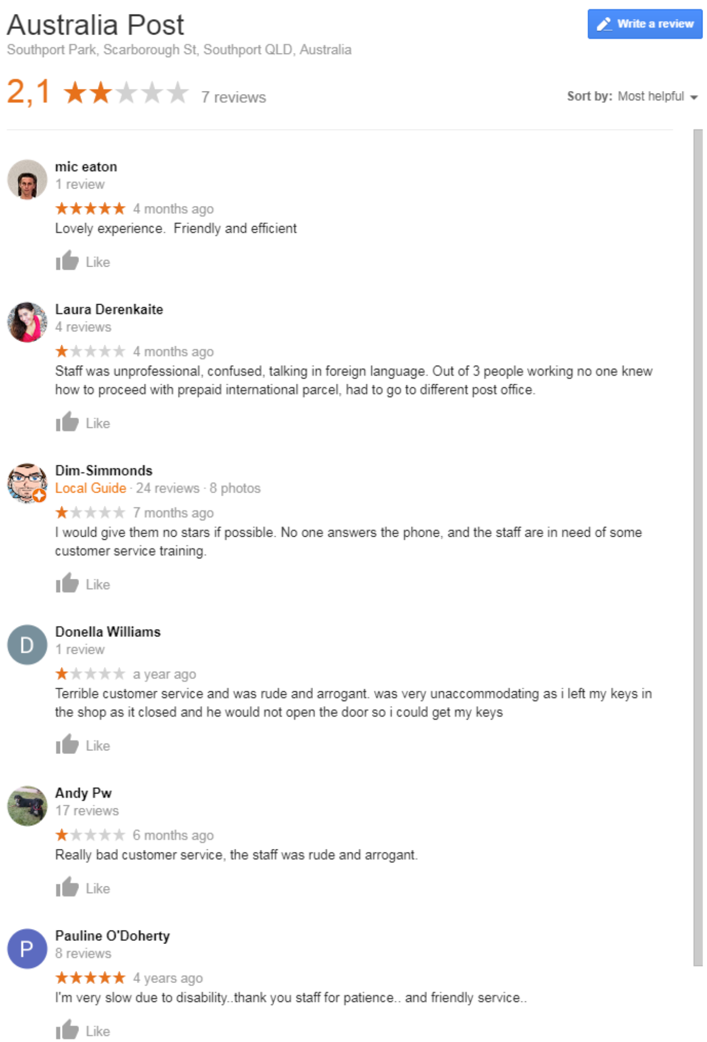
	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date	Loss Share Type	Agreement Terminated	Termination Date
0	First CornerStone Bank	King of Prussia	PA	35312	First-Citizens Bank & Trust Company	May 6, 2016	July 12, 2016	none	NaN	NaN
1	Trust Company Bank	Memphis	TN	9956	The Bank of Fayette County	April 29, 2016	August 4, 2016	none	NaN	NaN
2	North Milwaukee State Bank	Milwaukee	WI	20364	First-Citizens Bank & Trust Company	March 11, 2016	June 16, 2016	none	NaN	NaN
3	Hometown National Bank	Longview	WA	35156	Twin City Bank	October 2, 2015	April 13, 2016	none	NaN	NaN
4	The Bank of Georgia	Peachtree City	GA	35259	Fidelity Bank	October 2, 2015	April 13, 2016	none	NaN	NaN
					United Fidelity Bank	July 10	July 12			

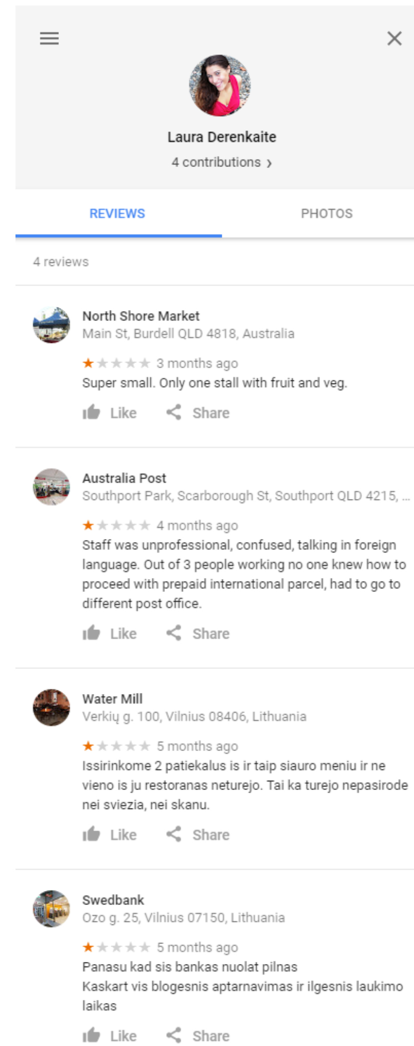
III. Data Cleaning with Python

Data cleaning

- ❖ Data cleaning: very time-consuming step in data analytics
- ❖ Why data cleaning?
  - ➢ Make analytical result reliable
    - ○ Result could be skewed/bias → mistakes, wrong business decisions (e.g. fake reviews lead to wrong conclusion of customer opinion)
  - ➢ Reduce the effort of data analytics:
    - ○ Simple models on clean data can outperform complex models on dirty data
- ❖ Challenges:
  - ➢ Too many sources produce dirty data
  - ➢ No generic framework for data cleaning → handle case by case
  - ➢ Need background knowledge → human involvement→ costly, time consuming, not scalable

Why data cleaning? – Bad reviews



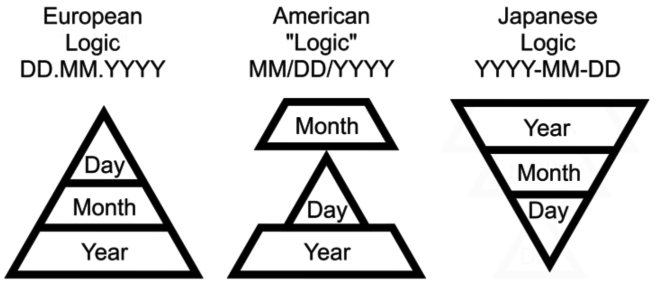


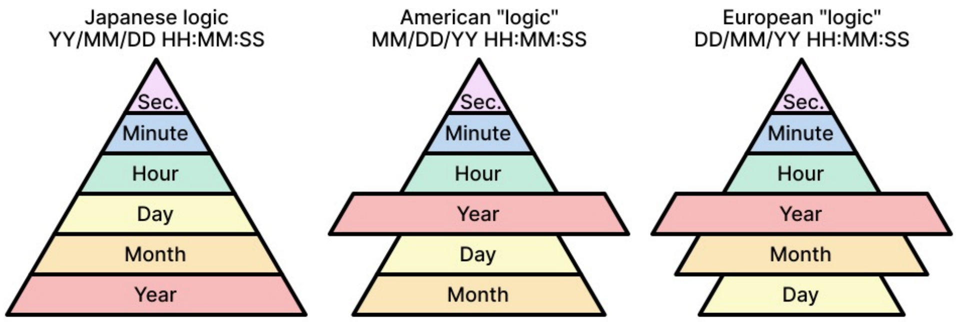
Types of “dirty” data

- Formatting
- Missing data
- Erroneous data
- Irrelevant data
- Inconsistent data
- Malicious data
- Outliers

Formatting

- **What:** the same entity can be inconsistently formatted
- **Why:** different standards, different platforms
- **Examples:**
  - Dates (US format …)
  - Phone numbers (parentheses, dashes)
- **How to handle:**
  - Identify all standard formats
  - Convert to the same format





Missing data

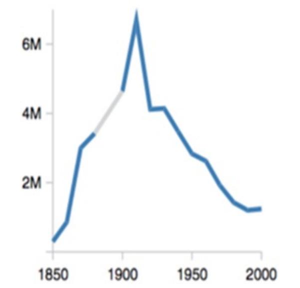
- **What:** Some of the data are not there
- **Why:**
  - Invalid data and ingestion
    - Invalid city
    - Missing state
    - Missing zipcode
  - Unexpected incident
    - Server crash

Customer	Address Line 1	Address Line 2	City	State	Zip
Antony Mc James IV	123 Untidy Cir	Suite# 234	Cumbersome	TX	76849
Miller Johnson	1102 Messy Data St	Middletow	OH		
Betty Flyier 6483 Phew Lane	Apt A4	FixMe	CA	91103	

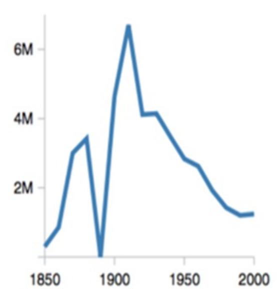
Dealing with missing data

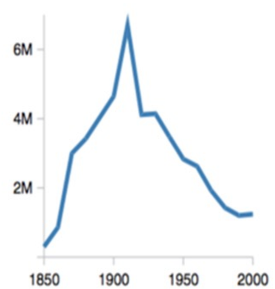
Knowledge about domain and data collection should drive your choice!

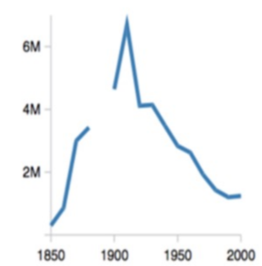
- Create new classification (e.g. set values to zero)
- Interpolate based on existing data
- Omit missing data



U.S. census counts of people working as ''Farm Laborers''; values from 1890 are **missing due to records being burned in a fire**







Erroneous data

- **What:** recurring error data in a particular case
- **Why:**
  - Software bug
  - Interrupted process that generates data: e.g. user is disconnected during a Web survey session
- **How to handle:**
  - Dig into unreasonable data
  - Look at things aren’t being combined properly: e.g. sum, product

Irrelevant data

- **What:** data whose non-existence does not affect your results
- **Why:** redundant data crawled from the Web
- **Example:** you are interested in only data from Gold Coast city → all data from the rest of the world are irrelevant
- **How to handle:**
  - Simply throw all irrelevant data (or select the subset of data that are relevant)
    - Define the rules to filter out (or manually)
    - Be careful! Over-cleaning irrelevant data might become missing data

Inconsistent data

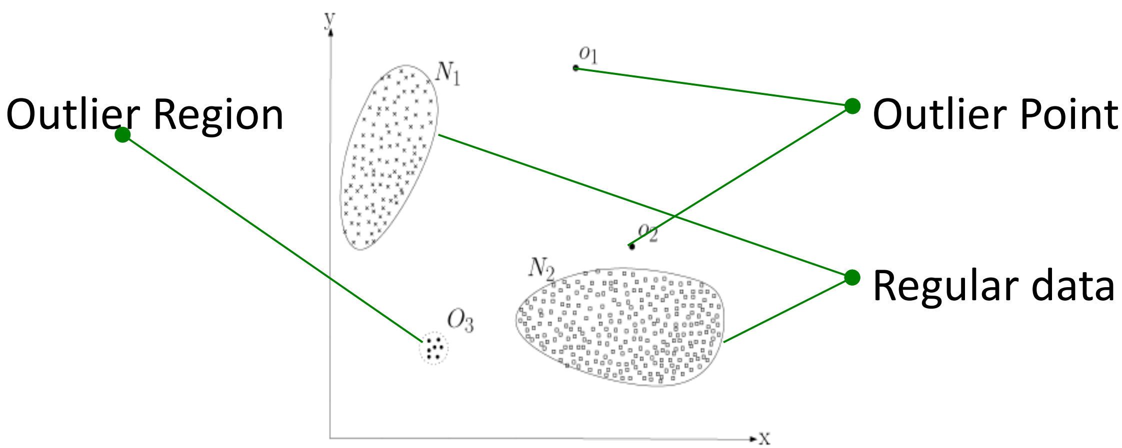
- **What:** the same data can be represented in different ways
- **Why:**
  - Different platforms
  - Different input preferences
- **Examples:**
  - Same address can be written in many different ways (street abbreviation, order between number and street, zip).
  - Movies/Books might have different names in different countries
- **How to handle:**
  - List all variations and data representations
  - Normalize/combine them all together to get the correct results

Malicious data

- **What:** data is intended to cause undesired effects
- **Why:**
  - People trying to game/trick/cheat your system
  - Malicious attacks: e.g. spam
- **How to deal:**
  - Identify the attacks
  - Filter them out from the results

Outliers

- **Outlier:** an observation point that is distant from other observations (a.k.a. noises, anomalies)
  - E.g. malicious data are often outliers
  - But non-malicious data can still be outlier



A simple example of anomalies in a two-dimensional data set [Chandola et al. 2009].

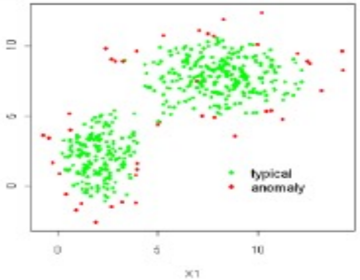
Source of Outliers

- **Human errors**
  - Due to measurements (measure wrongly)
  - By accidents (spelling, typos, data entry errors, etc.)
  - Due to human bias (e.g., coding disease code in hospital), etc.
- **Errors due to machines**
  - Software bugs
  - Data crawling errors
  - Data integration errors, etc.
- **Others**
  - Errors may be deliberate, duplicates, stale data (artifact of caching, not being refreshed).

How to handle outliers

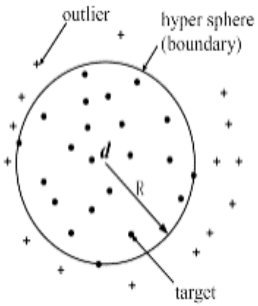
Distance-based

Based on well-defined distance metrics to compute the dissimilarity between two data points



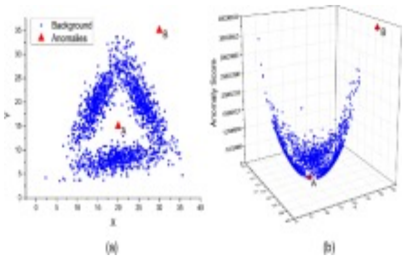
Domain-based

Learn domain of normality that characterises normal data



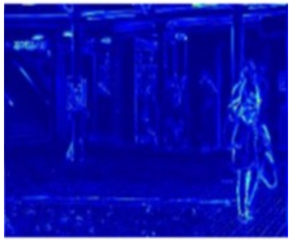
Information theoretic

Based on the information content of data to decide (e.g., entropy, mutual information, KL divergence)



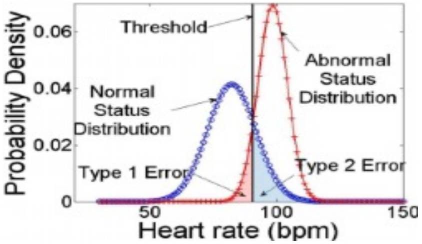
Reconstruction-based

Based on reconstruction error to decide (e.g., PCA, auto-encoder)



Probabilistic

Based on probabilistic likelihood to decide (e.g., parametric and non-parametric mixture model)



Data Cleaning: Summary

- Look at your data and examine it
- Question your results:
  - Look for weird things
  - Suspect "good" things even if you like your results
- Best practices:
  - Check frequencies of continuous and categorical variables for detection of unexpected values. For continuous variables, look into data "clumps" and "gaps".
  - Check the type or numeric variables: decimal, integer, and date.
  - Check the meanings of misinformative values, e.g., "NA", the blank, "", the number '0', the letter 'O', the dash, "-", and the dot "."
  - Check for out-of-range data: Values "far out" from the fences" of the data.
  - Check for outliers: Values "outside" the fences of the data.
  - Check for missing values, and the meanings of their coded values, e.g, the varied string of "9s", the number "0", the letter "O", the dash "-", and the dot ".".
  - Check the logic of data, e.g., response rates cannot be 110%, and weigh contradictory values, along with conflict resolution rules, e.g., duplicate DOB: 12/22/56 and 12/22/65.
  - Last but not least, check for the typos.

Bruce Ratner [IDS, ch.5]

Data cleaning: Usecase

- Review data from Amazon
- Import data

```
ds_folder = "../dataspace/amazon-reviews/"
dataset = getDF(ds_folder + 'reviews_Cell_Phones_and_Accessories_5.json.gz')
dataset.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A30TL5EWN6DFXT	120401325X	christina	[0, 0]	They look good and stick good! I just don't li...	4.0	Looks Good	1400630400	05 21, 2014
1	ASY55RVNIL0UD	120401325X	emily l.	[0, 0]	These stickers work like the review says they ...	5.0	Really great product.	1389657600	01 14, 2014
2	A2TMXE2AFO7ONB	120401325X	Erica	[0, 0]	These are awesome and make my phone look so st...	5.0	LOVE LOVE LOVE	1403740800	06 26, 2014
3	AWJ0WZQYMYFQ4	120401325X	JM	[4, 4]	Item arrived in great time and was in perfect ...	4.0	Cute!	1382313600	10 21, 2013
4	ATX7CZYFXI1KW	120401325X	patrice m rogoza	[2, 3]	awesome! stays on, and looks great. can be use...	5.0	leopard home button sticker for iphone 4s	1359849600	02 3, 2013

Usecase: Cleaning Amazon reviews

- Formatting
  - Convert data to a standard format, so it can be manipulated easily.

reviewTime is not datetime type so we have to convert it to datetime type

```
df['reviewTime_convert']=pd.to_datetime(df.reviewTime)
```

```
df[['reviewTime','reviewTime_convert']].head(5)
```

	reviewTime	reviewTime_convert
0	05 21, 2014	2014-05-21
1	01 14, 2014	2014-01-14
2	06 26, 2014	2014-06-26
3	10 21, 2013	2013-10-21
4	02 3, 2013	2013-02-03

Usecase: Cleaning Amazon reviews

- Formatting
  - Data of same type should be in the same format

Change the type of unixReviewTime to datetime format.

```
df['unixReviewTime_convert'] = pd.to_datetime(df['unixReviewTime'],unit='s')
df[['unixReviewTime','unixReviewTime_convert']].head()
```

	unixReviewTime	unixReviewTime_convert
0	1400630400	2014-05-21
1	1389657600	2014-01-14
2	1403740800	2014-06-26
3	1382313600	2013-10-21
4	1359849600	2013-02-03

Usecase: Cleaning Amazon reviews

- Formatting

Change overall to integer type

```
df['overall']=df['overall'].astype(int)
df['overall'].head()
```

```
0    4
1    5
2    5
3    4
4    5
Name: overall, dtype: int64
```

Usecase: Cleaning Amazon reviews

- Check for missing values
  - Displays the number of missing values in each column.
  - Should we remove the *reviewerName* column?

```
df.isnull().sum()
```

```
reviewerID                   0
asin                         0
reviewerName              3519
helpful                      0
reviewText                   0
overall                      0
summary                      0
unixReviewTime               0
reviewTime                   0
reviewTime_convert           0
unixReviewTime_convert       0
year                         0
length_review                0
helpfulness_rate             0
dtype: int64
```

Usecase: Cleaning Amazon reviews

- **Dealing with missing values:** We notice that some of the reviewer names are missing but since the reviewer Id are available anyway, we can replace missing values with "Unknown".

```
df.fillna('Unknown',inplace=True)
```

```
df.isnull().sum()
```

```
reviewerID                0
asin                      0
reviewerName              0
helpful                   0
reviewText                0
overall                   0
summary                   0
unixReviewTime            0
reviewTime                0
reviewTime_convert        0
unixReviewTime_convert    0
year                      0
length_review             0
helpfulness_rate          0
dtype: int64
```

Usecase: Cleaning Amazon reviews

- Check for invalid data
  - **overall** must be between **0 – 5**

```
df.loc[(df['overall']<0) | (df['overall']>5)]
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime	reviewTime_convert	unixReviewTime_conv

No invalid data!

Usecase: Cleaning Amazon reviews

- Check for invalid data
  - **reviewTime** must be between **May 1996 – July 2014**

```
df.loc[(df[‘reviewTime_convert’]<‘1996-05-01’) |
        (df[‘reviewTime_convert’]>’2014-07-31’)]
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime	reviewTime_convert	unixReviewTime_conv

No invalid data!

Usecase: Cleaning Amazon reviews

- **Cleaning irrelevant data**: The field *unixReviewTime* has the same meaning of *reviewTime* in the dataset, thus we can remove this column.

```
del df['unixReviewTime']
```

Usecase: Cleaning Amazon reviews

- **Cleaning inconsistent data**: two different reviewers should not have the same ID.

```
group = df.groupby('reviewerID')['reviewerName'].unique()
group[group.apply(lambda x: len(x)>1)].head(10)
```

```
reviewerID
A102TCMSQAJIDR                                   [crmorris, Unknown]
A103CDLPIN7OO9                                   [Patricia, Unknown]
A103D3GYGFHS96                            [Daisy Cartagena, Unknown]
A103EFN0LE0PPT                                       [Unknown, tmar]
A10436JZZW87TE                                     [Hector, Unknown]
A104R1UFTJNBKK                            [Amazon Customer, Unknown]
A105K765Z5SX1A                            [Springs Shopper, Unknown]
A105S56ODHGJEK    [Peace Daddy "Eclectic ReflectionZ", Unknown]
A10DHJK4D0QFKR                                        [stu, Unknown]
A10DHVWCLL3EA3                                 [Bassinator, Unknown]
Name: reviewerName, dtype: object
```

References

[1] https://www.slideshare.net/e2m/introduction-to-ipython-jupyter-notebooks

[2] https://www.slideshare.net/mbussonn/jupyter-a-platform-for-data-science-at-scale