

Data Science in the Wild

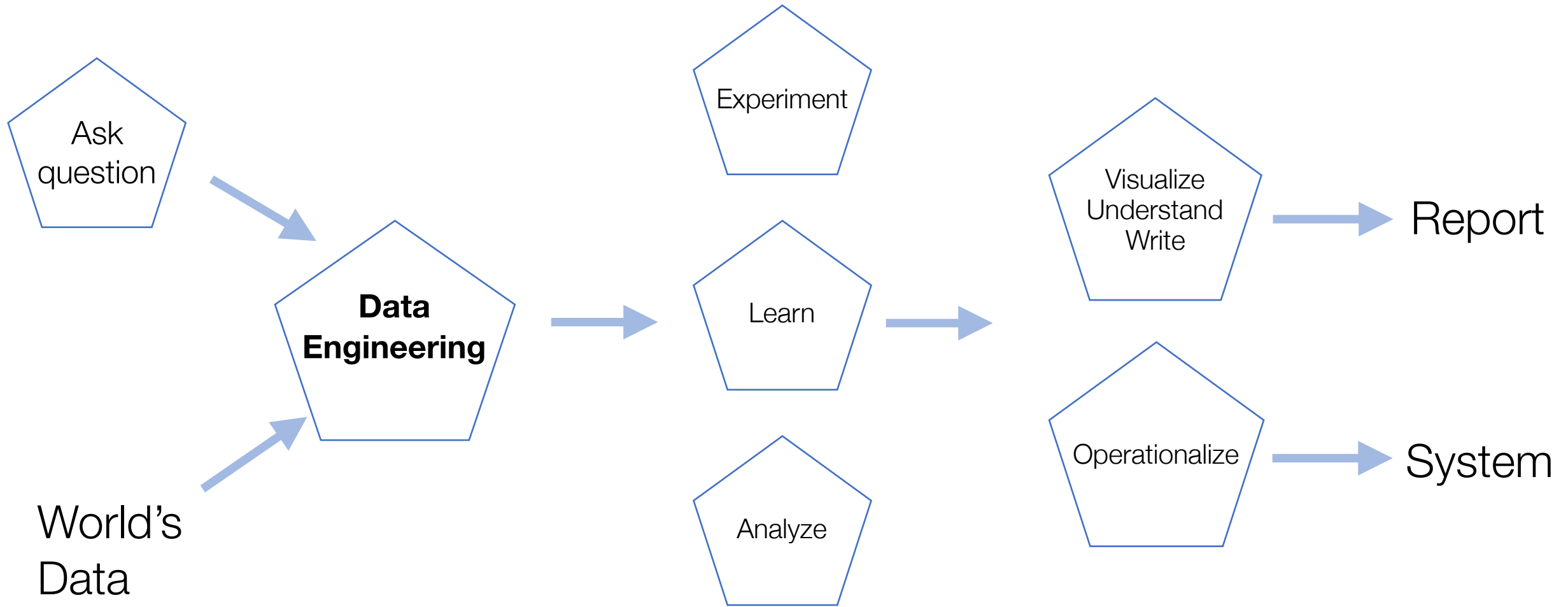
Lecture 3: ETL - Extract, Transform, Load

Eran Toch

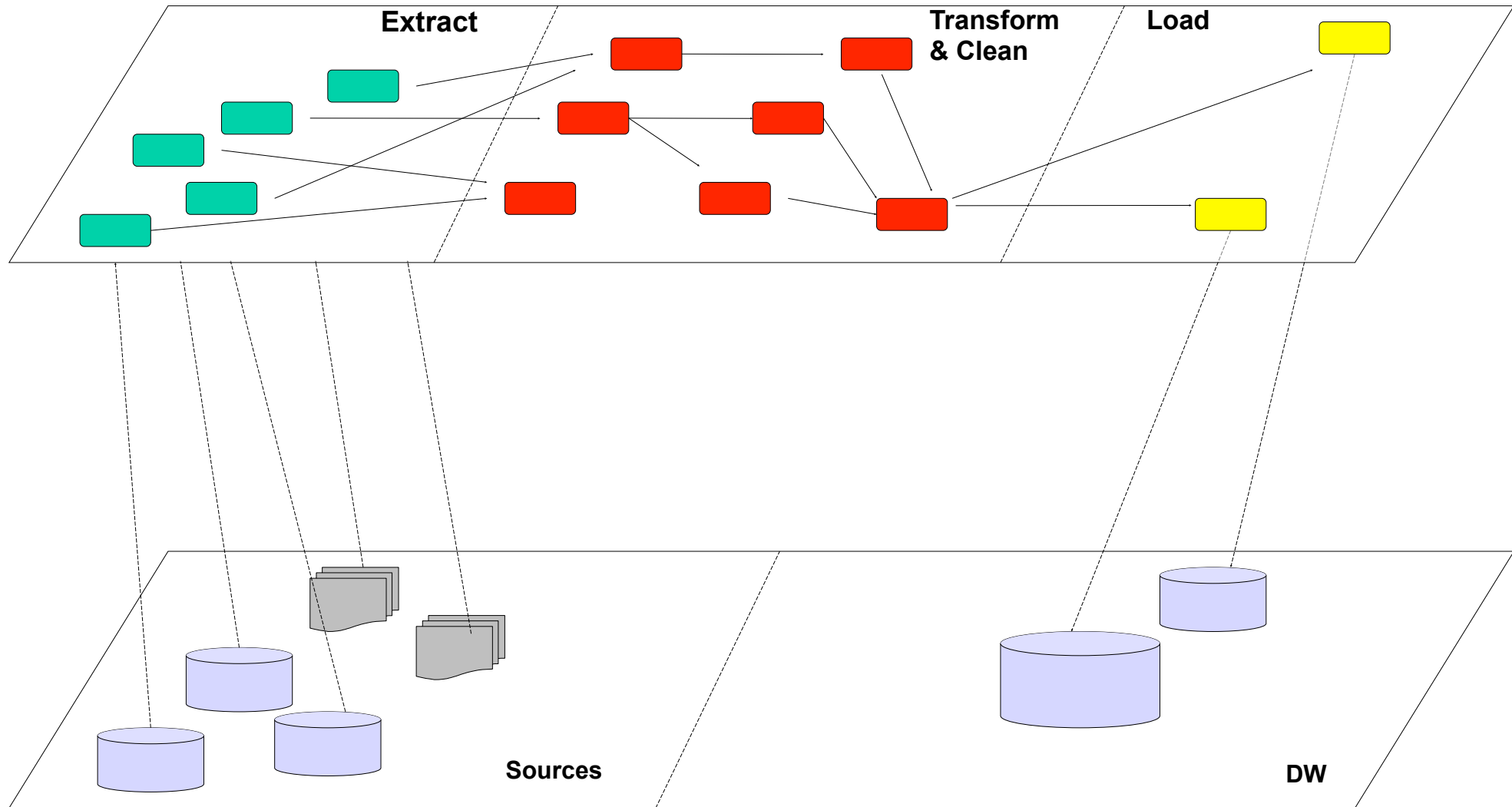


**CORNELL
TECH**

The Data Science Model



ETL Pipeline



ETL: Practical Considerations

- Typically, ETL takes 80% of the development time in a DW project (Vassiliadis et al.).
- ETL is particularly difficult to generalize beyond one data science task
 - Why?

Figure 1. Magic Quadrant for Data Integration Tools

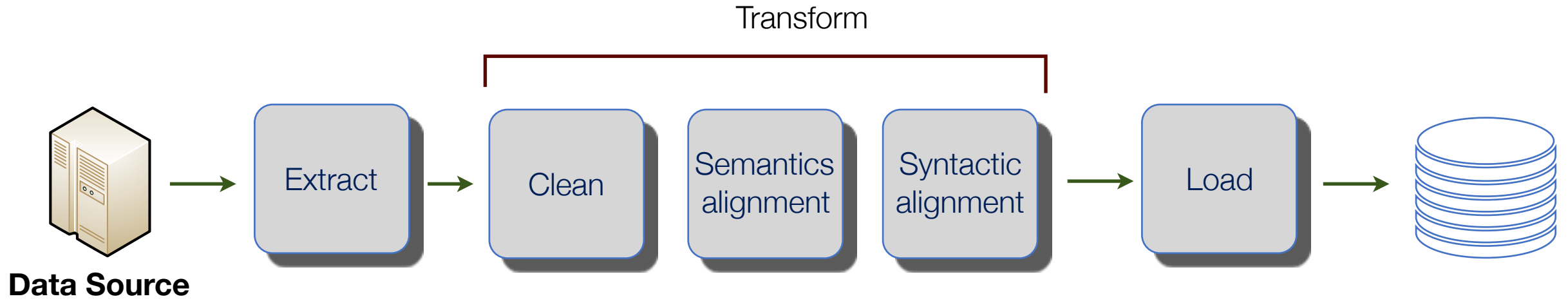


Source: Gartner (July 2018)

Agenda

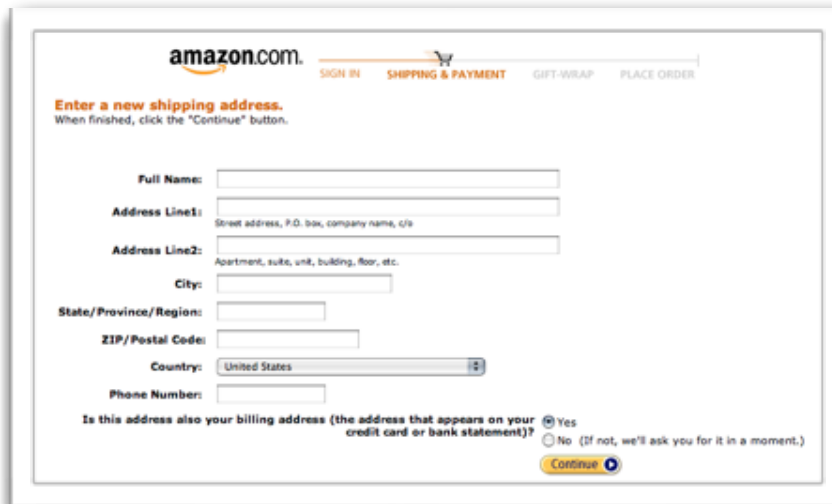
1. ETL Processes
2. Pandas
3. Cleaning datasets
4. Handling missing data
5. Handling outliers
6. Understanding data sources

ETL Processes



Methods for Data Access

Online Access



amazon.com. SIGN IN SHIPPING & PAYMENT GIFT-WRAP PLACE ORDER

Enter a new shipping address.
When finished, click the "Continue" button.

Full Name:

Address Line1:
Street address, P.O. box, company name, c/o

Address Line2:
Apartment, suite, unit, building, floor, etc.

City:

State/Province/Region:

ZIP/Postal Code:

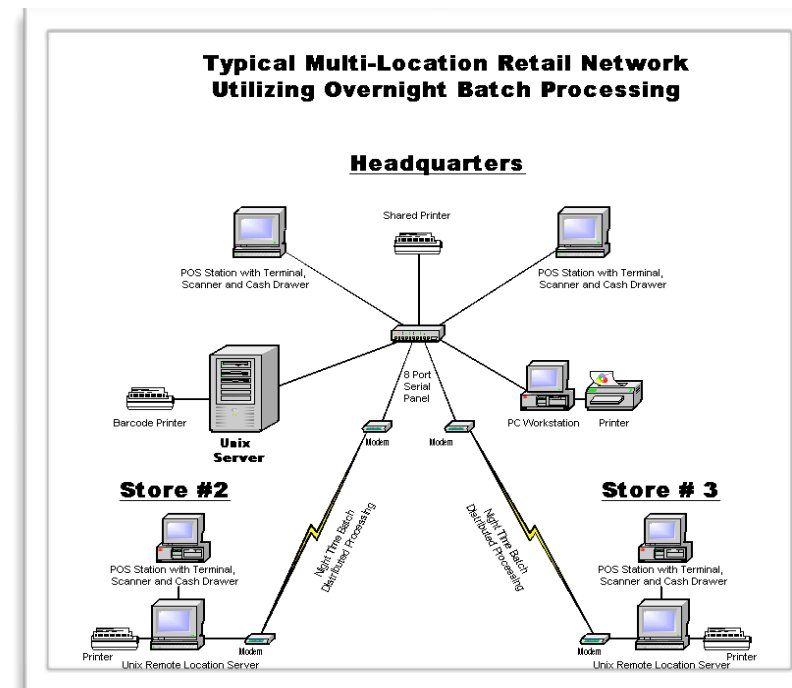
Country:

Phone Number:

Is this address also your billing address (the address that appears on your credit card or bank statement)? Yes No (If not, we'll ask you for it in a moment.)

Continue

Batch Access

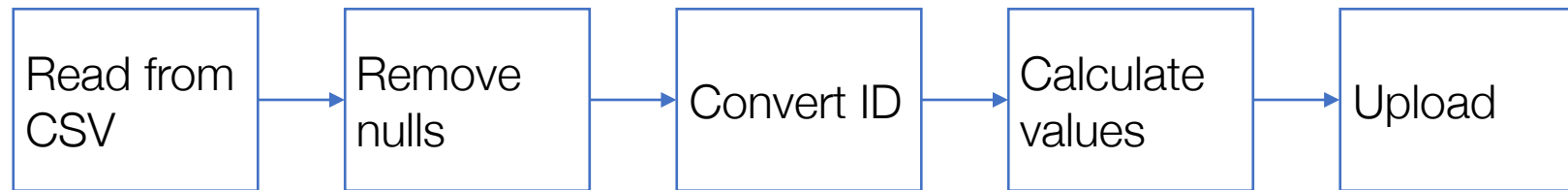


Extracting Data

- Technical processes of data extraction:
 - Access rights and authorization
 - Finding the data
 - Accessing the right fields
- Are there good times to extract the data?
 - At night?
 - On weekends?

ETL Pipelines

- ETL pipelines describe how data can be transferred and transformed



- Pipelines can run in parallel, for each data items
- Pipelines can be activated using:
 - Pull - when the database requests the data
 - Push - when the original data is changed

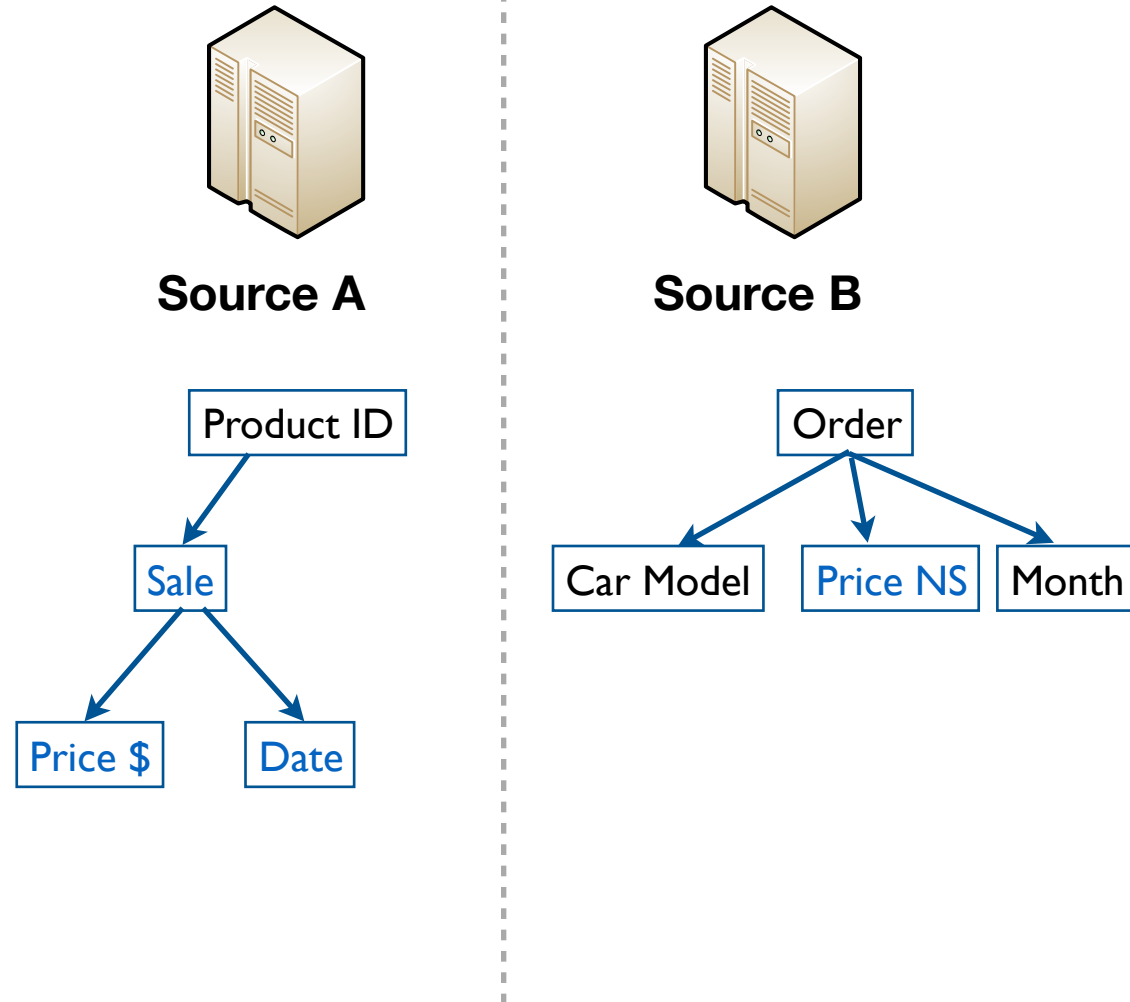
Whole data vs. Delta Extraction

- The first data extraction extracts all the data in the operational systems
- The “Delta” extraction takes only the data that was updated since the last extraction

Date	Purchase Sum	Order ID
1/1/2011	13	100
2/1/2011	23	101
3/1/2011	22	102
4/1/2011	34	103
5/1/2011	56	104
6/1/2011	2	105

Alignment Processes

- Aligning data from several data sources
- We describe two levels of alignment:
 - Semantic
 - Syntactic



Semantics

- Semantics is the study of meaning that is used by humans to express themselves through language.
- Which can be programming language, or a database model.

Semantics (from **Greek** sēmantiká) is the study of meaning

Semantic Alignment

- The meaning of the data items needs to be aligned.
 - Dates (When is the 12/3/2011?)
 - Addresses
 - Names
 - Phone numbers
- And the semantic relationship between items
 - Order <> Sale
 - Product <> Car
 - Product <> Inventory Item
 - Product (Screwdriver) <> Product Part

Key alignment is central

- Key alignment:
 - Order \leftrightarrow Sale?
 - Product \leftrightarrow Car?
 - Customer_id \leftrightarrow Account_id?
 - Username \leftrightarrow Account_id?
 - Are there accounts with multiple usernames?

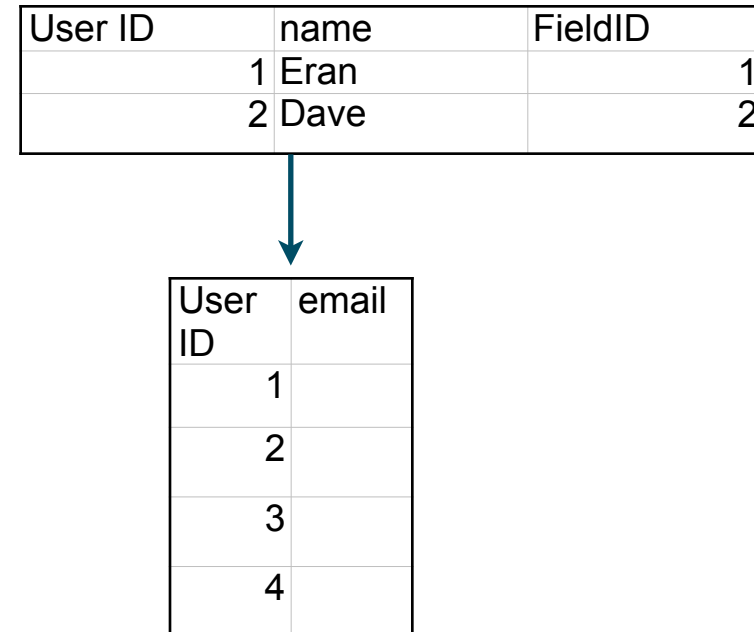
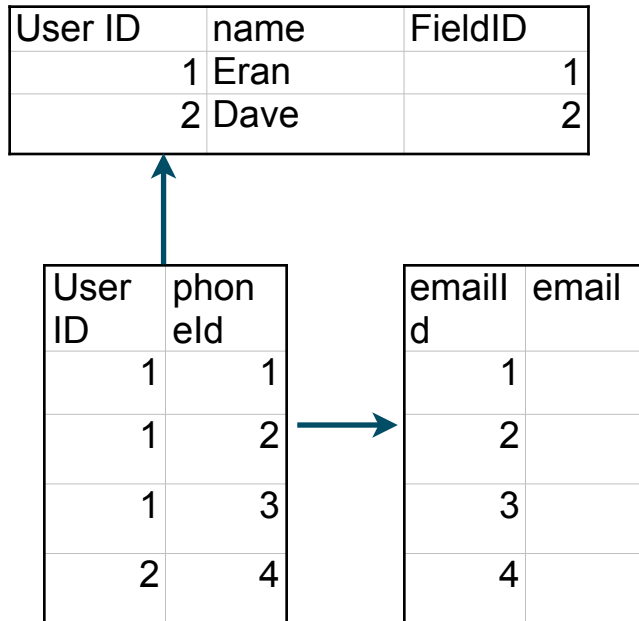
Syntax

- the rules and principles that govern the sentence structure.
- In our domain, the data structure.

syntax (from [Ancient Greek σύνταξις](#))
is an "arrangement"

Syntax Alignment

Syntactic heterogeneity represents the difference in structure that represent data elements:



Summary

- Batch vs. online
- Handling deltas
- Semantics vs. syntax

Data Manipulation

Records

Features			
ID	F1	F2	F3
1	A	126.3	North
2	B	33.4	South
3	A	33.1	West
4	B	98.4	East

Data Object

- Filter horizontally (remove features)
- Filter vertically (remove records)
- Change values
- Combine objects
- Aggregate (eliminate records)
- Drill down (create new records)

PANDAS Package

Pandas is a Python library that allows loading, transforming, and loading to/from databases and many types of files

- DataFrame: A table made of rows and columns
- Series: a single column (a one-dimensional labeled array)

ID	F1	F2	F3
1	A	126.3	North
2	B	33.4	South
3	A	33.1	West
4	B	98.4	East

Why Pandas?

- Rich ecosystem:
 - [Statistics and Machine Learning](#)
 - [Visualization](#)
 - IDE
 - API
 - [Domain Specific](#)
 - [Out-of-core](#)
 - [Data validation](#)
 - [Extension Data Types](#)
 - [Accessors](#)

Example

<https://www.kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region>

```
wdata = pd.read_csv("sudeste.csv")  
wdata.head()
```

	wsid	wsnm	elvt	lat	lon	inme	city	prov	mdct	date	...	tmax	dmax	tmin	dmin	hmdy	hmax	hmin	wdsp	wdct	gust
0	178	SÃO GONÇALO	237.0	-6.835777	-38.311583	A333	São Gonçalo	RJ	2007- 11-06 00:00:00	2007- 11-06	...	29.7	16.8	25.5	10.8	35.0	58.0	32.0	3.2	101.0	6.5
1	178	SÃO GONÇALO	237.0	-6.835777	-38.311583	A333	São Gonçalo	RJ	2007- 11-06 01:00:00	2007- 11-06	...	29.9	13.6	29.0	12.2	39.0	39.0	35.0	3.6	94.0	6.4
2	178	SÃO GONÇALO	237.0	-6.835777	-38.311583	A333	São Gonçalo	RJ	2007- 11-06 02:00:00	2007- 11-06	...	29.0	14.0	27.4	13.6	44.0	44.0	39.0	2.5	93.0	6.9
3	178	SÃO GONÇALO	237.0	-6.835777	-38.311583	A333	São Gonçalo	RJ	2007- 11-06 03:00:00	2007- 11-06	...	27.4	16.9	25.8	14.1	58.0	58.0	44.0	1.7	96.0	5.8
4	178	SÃO GONÇALO	237.0	-6.835777	-38.311583	A333	São Gonçalo	RJ	2007- 11-06 04:00:00	2007- 11-06	...	26.3	17.0	25.3	16.4	57.0	58.0	56.0	3.1	110.0	7.5

5 rows x 31 columns

Reading from

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

Transformation

- Selecting first n rows

```
table2 = wdata.head(4)
```

- Selecting last n rows

```
table2 = wdata.tail(4)
```

- Selecting a subset of rows

```
table2 = wdata[5:8]
```

```
>>> table2
```

- Selecting a subset of columns

```
table2 = wdata[['lat','lon']]
```

- Adding fields

```
table2 = wdata['new_column'] = 7
```

Many other operations:

- Concatenating dataframes
- Appending
- Managing indexes
- Selecting data

Pipelining

- Extract a table from a file-like source or database
- Use `.pipe` when chaining together functions that expect Series, DataFrames or GroupBy objects. Instead of writing

```
>>> f(g(h(df), arg1=a), arg2=b, arg3=c)
```

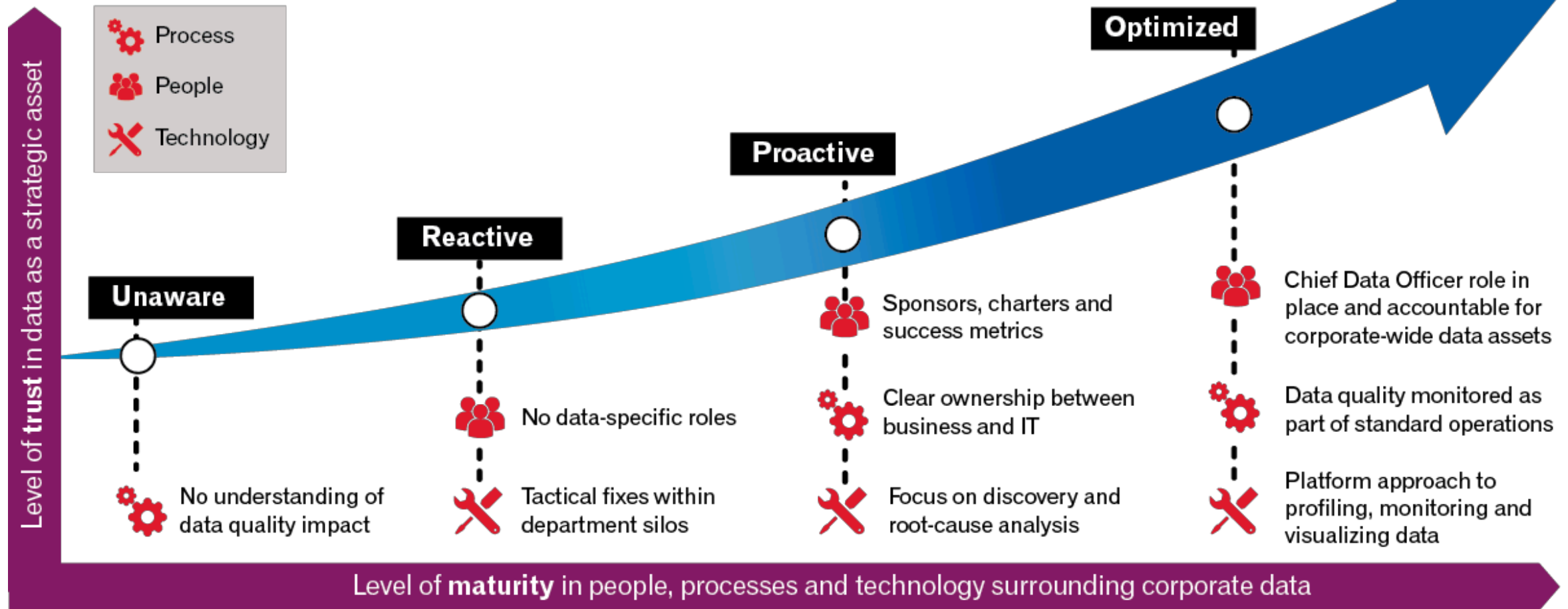
- Pipes allow:

```
>>> (df.pipe(h)
...   .pipe(g, arg1=a)
...   .pipe(f, arg2=b, arg3=c)
...   )
```


Why is Data Quality Important?

- Garbage in, Garbage out
 - Some machine learning algorithms are particularly sensitive to missing values and noisy data
 - If we don't understand the errors, we don't understand the data
- And no silver bullet
- In general, organizations need to plan and apply a data quality strategy

Data quality sophistication curve



<https://www.edq.com/blog/how-sophisticated-is-your-data-quality-strategy/>

“Wild” Data is dirty

- Completeness: relevant data is missing
- Noisy: the data is not recorded correctly
- Consistency: the data does not agree with itself
- Uniqueness: the same entities are recorded more than once
- Timeliness: the data is not up to date



Data Quality and Data Cleaning: An Overview , Theodore Johnson, Tamraparni Dasu

Incomplete

- Incomplete: missing attributes, missing records
 - Name = ""
 - Names, addresses, descriptions
 - How can you distinguish between optional and missing values?
- Reasons
 - Bugs and human errors
 - Evolving systems (started to produce data from version X+1)

Noisy

- Noisy: containing errors or outliers (spelling, typing errors, automated and human errors)
 - Price = 1.2, ..., price = 120000
- Reasons
 - Faulty sensors and data entry processes
 - Bugs in algorithmic calculations
 - Wrong predictions by ML

Inconsistent

- Containing differences in codes or names (different semantics for similar objects, variations, abbreviations, truncation and initials)
 - name="Eran Toch" / name = "Toch Eran"
 - Price = 1.2 / Price = 120
- Reasons
 - Multiple data sources
 - Non normalized data structures do not have strong reference mechanism (such as foreign keys.)

Cleaning Processes for Missing Data

- Rough checks: number of records, number of duplicates
- If we have some specification of the data: compare the data to the specification
- Scan individual records, and analyze whether there are differences between records
- Compare estimates (averages, frequencies, medians) of missing values to the actual data

Recognizing missing data

- Missing data can be identified as N/A
 - `np.nan` for numbers
- Or have some other value that represents missing data
 - -1 if you're lucky
 - 0 if you're not
- Understanding the proportion:

```
wdata['wdsp'].isnull().sum() / len(wdata)
0.09464619075978652
```

```
wdata.isna().sum()
Out[15]:
wsid          0
wsnm          0
elvt          0
lat           0
lon           0
inme          0
city          0
prov          0
mdct          0
date          0
yr            0
mo            0
da            0
hr            0
prcp      8371184
stp           0
smax          0
smin          0
gbrd      4108820
temp          31
```

Mechanisms for Missing Data

Missing Completely at Random (MCAR)

- The propensity for a data point to be missing is completely random.
- No relationship between whether a data point is missing and any values in the data set, missing or observed.

Missing at Random (MAR)

- MAR requires that the cause of the missing data is unrelated to the missing values but may be related to the observed values of other variables (higher probability for people in the U.S. not to report their salary)

Missing not at Random (MNAR)

- Missing value depends on the hypothetical value (e.g. people with high salaries generally do not want to reveal their incomes in surveys)

Solutions

- Listwise deletion (complete-case analysis)
- Dropping Variables
- Data Imputation

These solutions will work with MCAR, have a potential to work with MAR and will never work with MNAR

Listwise deletion

- Listwise deletion (complete-case analysis)
 - Removes all data for an observation that has one or more missing values
- Can work well for a small number of missing values
- Will work only on MCAR (Missing Completely at Random), and those are rare to support

```
df.dropna(inplace=True)
```

Table 1

	city	da	date	dewp	dmax
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
5	False	False	False	n/a	False
6	False	False	False	False	False
7	False	False	False	n/a	False

Dropping Features

- If the data has so many missing values, it might be preferable to remove the feature from our analysis

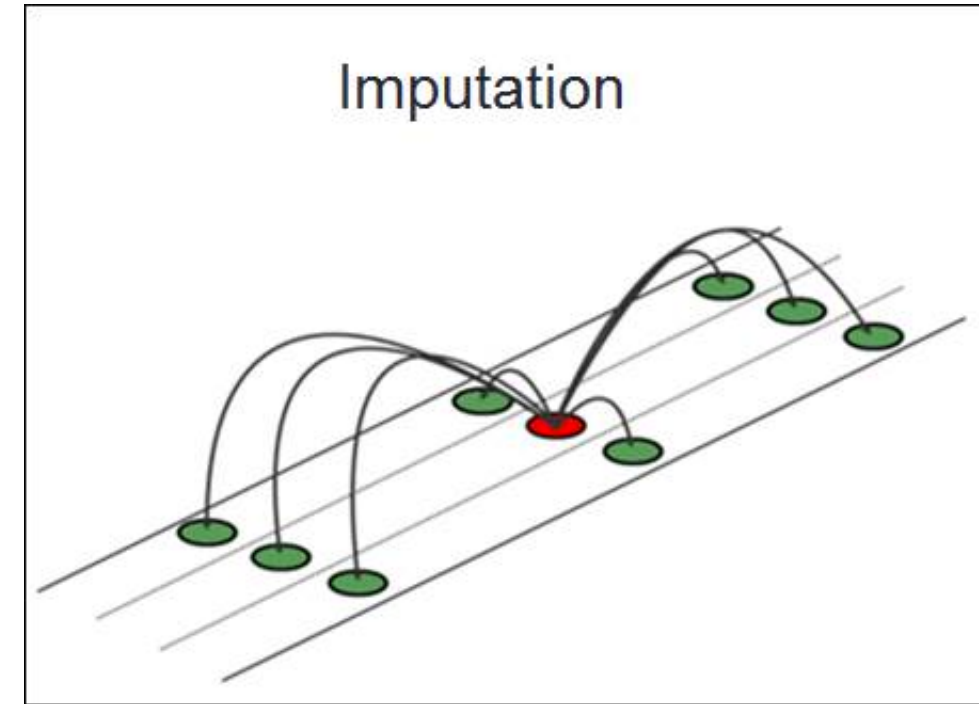
```
df.drop('Publisher', axis=1, inplace=True)
```

Table 1

	city	da	date	dewp	dmax
0	False	False	False	n/a	False
1	False	False	False	False	False
2	False	False	False	n/a	False
3	False	False	False	False	False
4	False	False	False	n/a	False
5	False	False	False	n/a	False
6	False	False	False	False	False
7	False	False	False	n/a	False

Data imputation

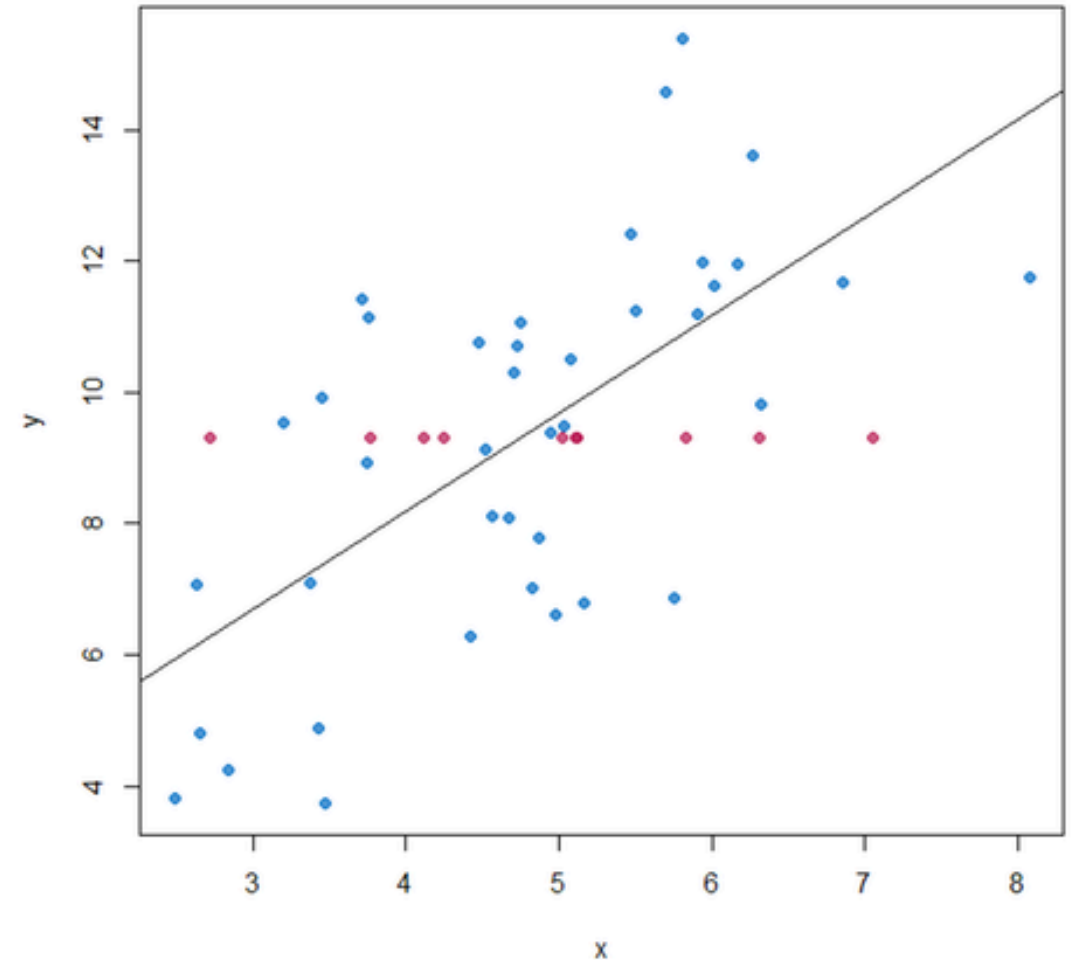
- Create new values instead of the missing ones
- Several methods have the potential of working well with MCAR:
 - Mean Imputation
 - Median Imputation
 - Regression Imputation
- Few can work with MAR:
 - Multiple Imputation
 - KNN



<https://www.mathworks.com/matlabcentral/fileexchange/60128-sequential-knn-imputation-method>

Mean imputation

- The missing value on a certain variable is replaced by the mean of the available cases
- The method maintains the sample size (and the mean), but variability, covariances and correlation are reduced
- This method often causes biased estimates (Enders, 2010; Eekhout et al, 2013).



<https://www.iriseekhout.com/missing-data/missing-data-methods/imputation-methods/>

Implementation

- Using the sklearn.impute library

```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
X = [[np.nan, 2], [6, np.nan], [7, 6]]
print(imp.fit_transform(X))
```

```
[[nan 2. ]
 [6.  nan]
 [7.  6. ]]
```

X

Mean of the first
column

```
[[6.5 2. ]
 [6.  4. ]
 [7.  6. ]]
```

imp.fit_transform(X)

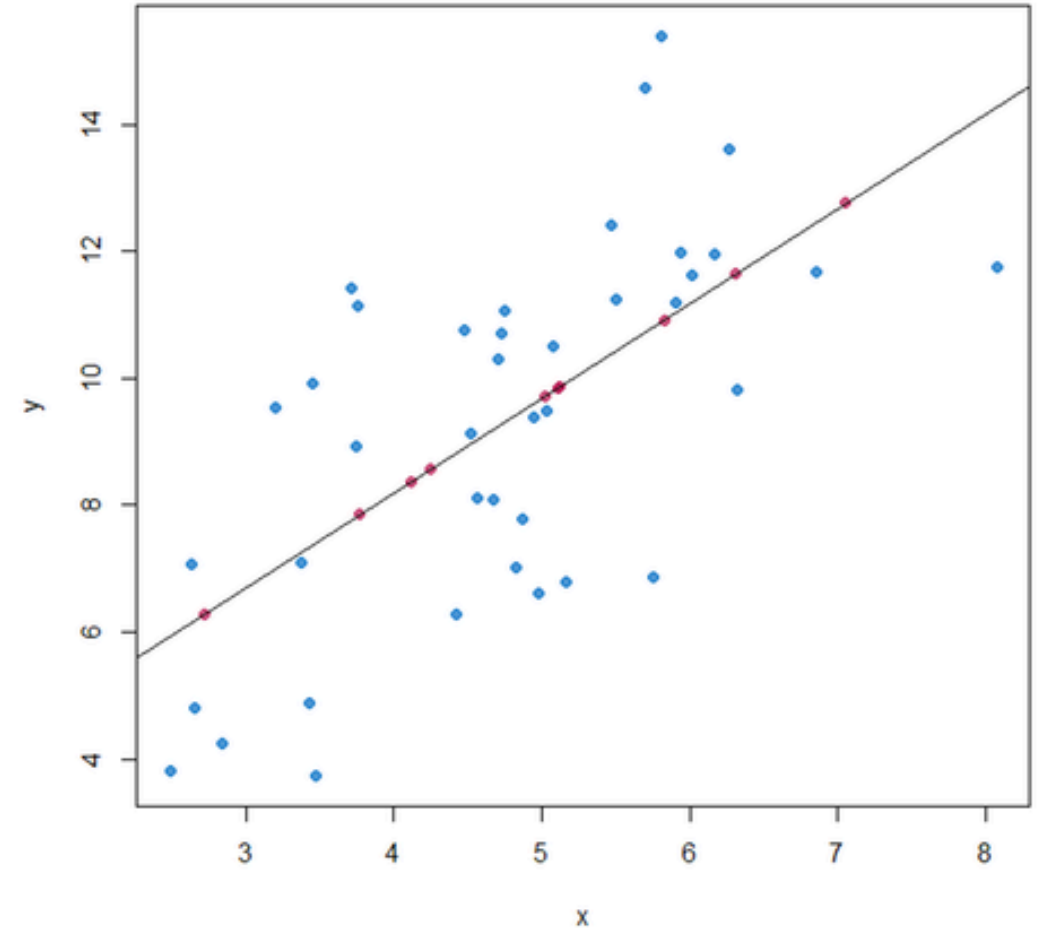
Mean of the
second column

SimpleImputer

- “mean”, then replace missing values using the mean along each column. Can only be used with numeric data.
- “median”, then replace missing values using the median along each column. Can only be used with numeric data.
- “most_frequent”, then replace missing using the most frequent value along each column. Can be used with strings or numeric data.
- “constant”, then replace missing values with fill_value. Can be used with strings or numeric data.

Regression imputation

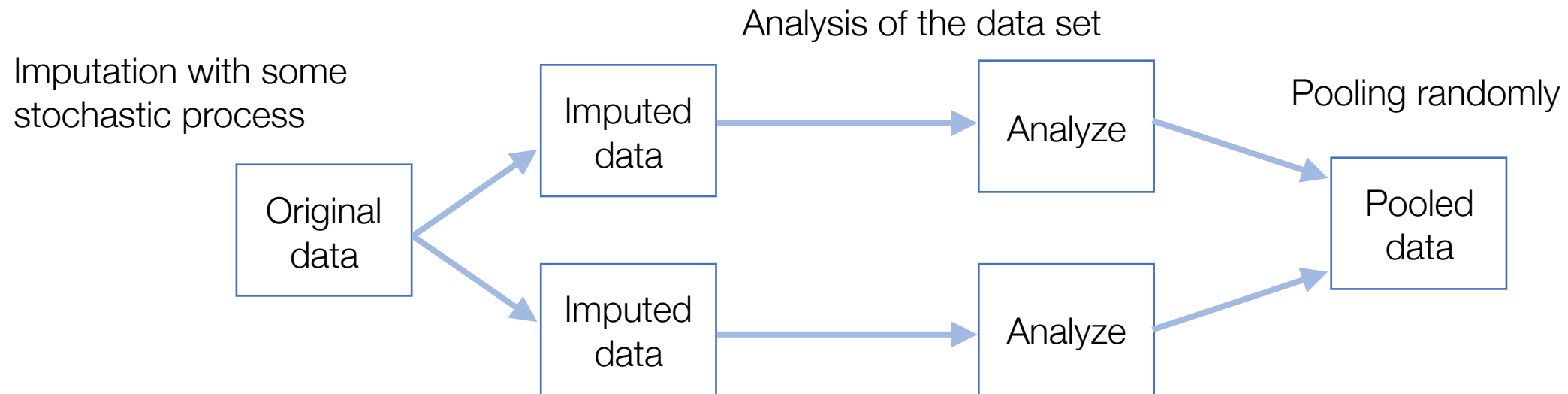
- The information in the complete observations is used to predict the values of the missing observations
- The method assumes that the imputed values fall directly on a regression line with a nonzero slope, so it implies a correlation of 1 between the predictors and the missing outcome variable
- Assumes linear relationships, and does not include any stochastic properties



<https://www.iriseekhout.com/missing-data/missing-data-methods/imputation-methods/>

Other methods for imputation

- KNN imputation and many other methods of machine learning
- In multiple imputation, the process is repeated multiple times, which adds some healthy uncertainty to the imputed data (works well with MAR)



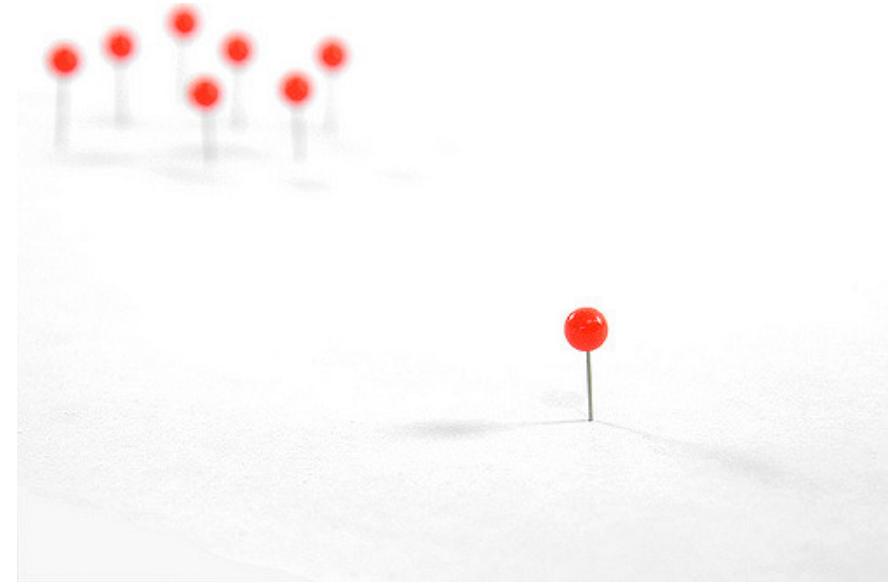
Summary

- The challenge of missing values
- Deletion
- Imputation

Outliers

- One of the elements of noisy data is outliers
- Definition of Hawkins (1980):

“An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different statistical mechanism”



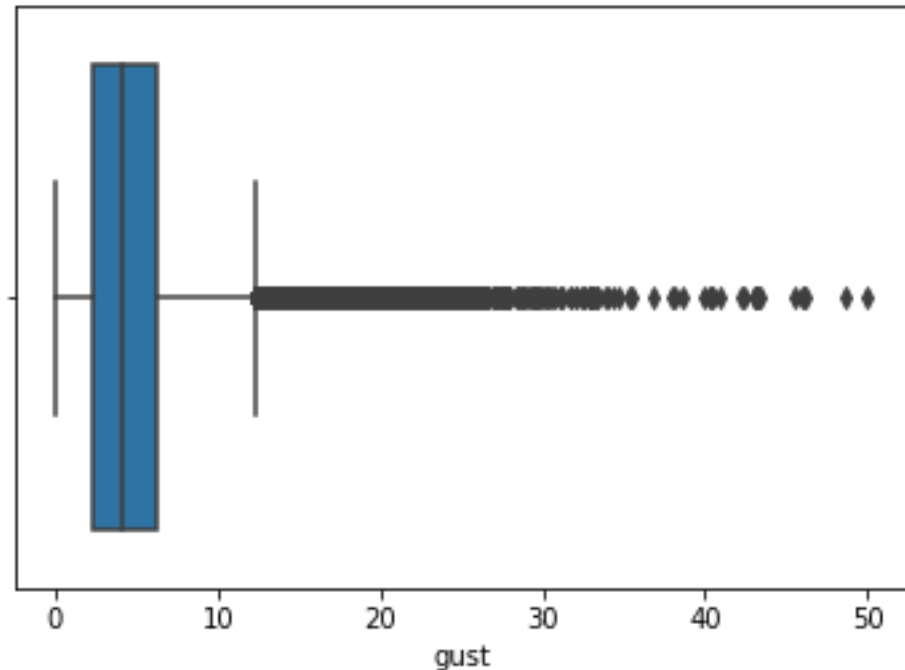
Applications of Outlier Detection

- Fraud detection
- Health analysis
- Detecting measurement errors
- Approaches:
 - Supervised
 - Unsupervised (our focus)

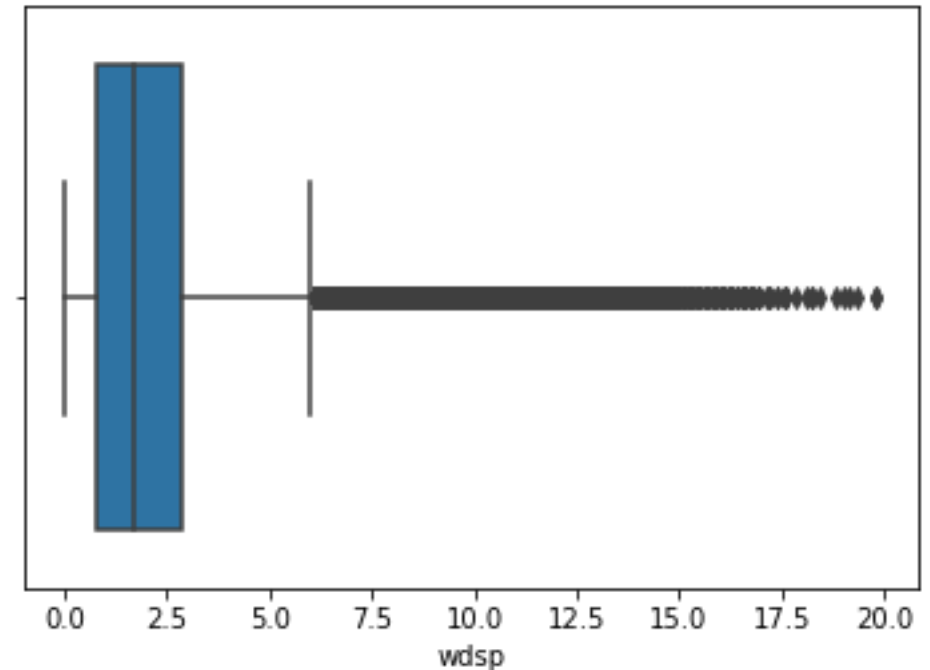
Understanding outliers

One way to understand outliers is to visualize them

```
sns.boxplot(x=wdata['gust'])
```



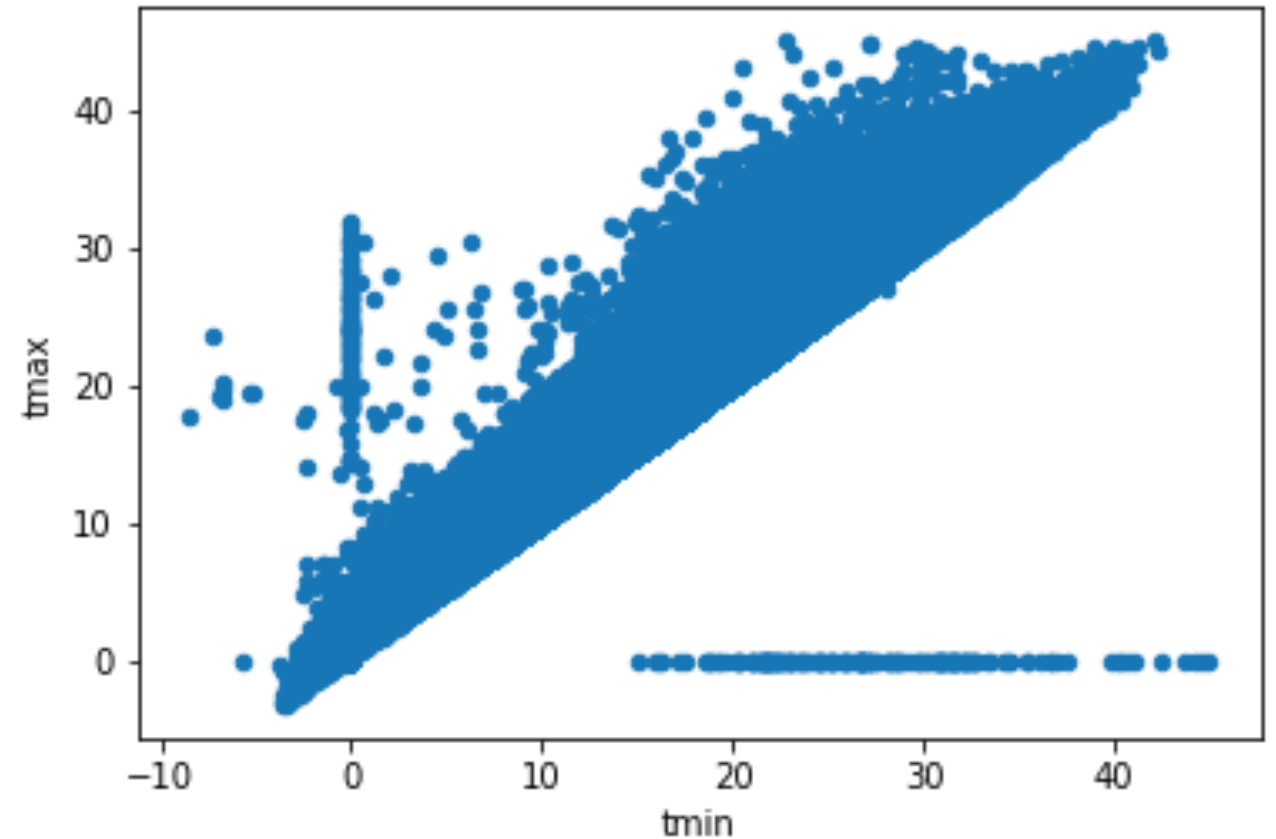
```
sns.boxplot(x=wdata['wdsp'])
```



Multi-dimensional analysis

- Multivariate outlier analysis is crucial with meaningful data
- Is there a way to automatically find outliers?

```
wdata.plot.scatter(x='tmin', y='tmax')
```

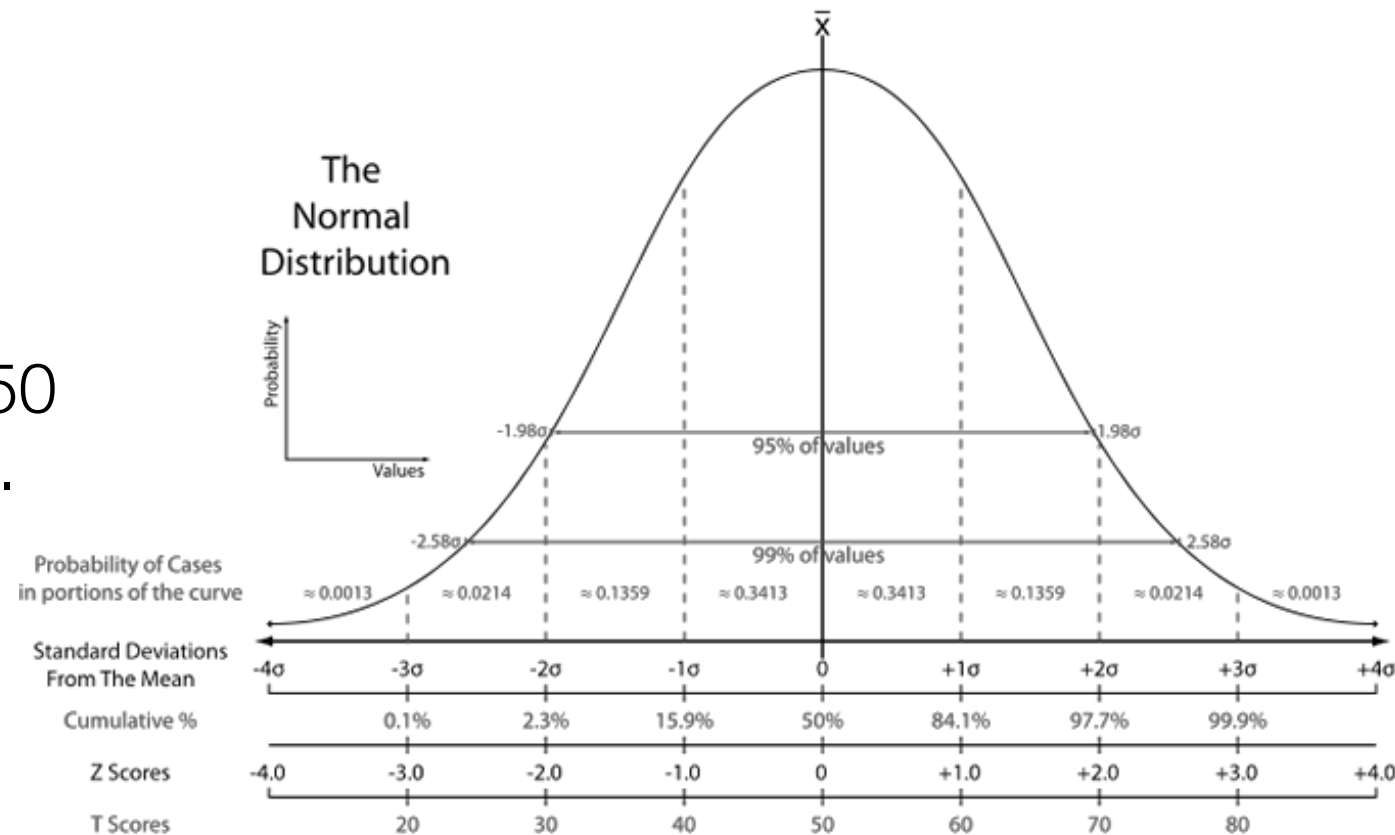


Z-Score

- z-score is the number of standard deviations from the mean a data point is

$$z = (x - \mu) / \sigma$$

- Let's say you have a test score of 190. The test has a mean (μ) of 150 and a standard deviation (σ) of 25. Assuming a normal distribution, your z score would be:
 - $= 190 - 150 / 25 = 1.6$



Finding outliers using z-score

- Calculating z scores

```
z = np.abs(stats.zscore(wsub['dmin']))
print(z)
[0.76290726 0.54725236 0.33159745 ... 0.57723393 0.50021432 0.42319472]
```

- Creating a threshold:

```
threshold = 3
print(np.where(z > 3))
(array([ 11, 12, 13, ..., 96474, 96475, 97825]),)
```

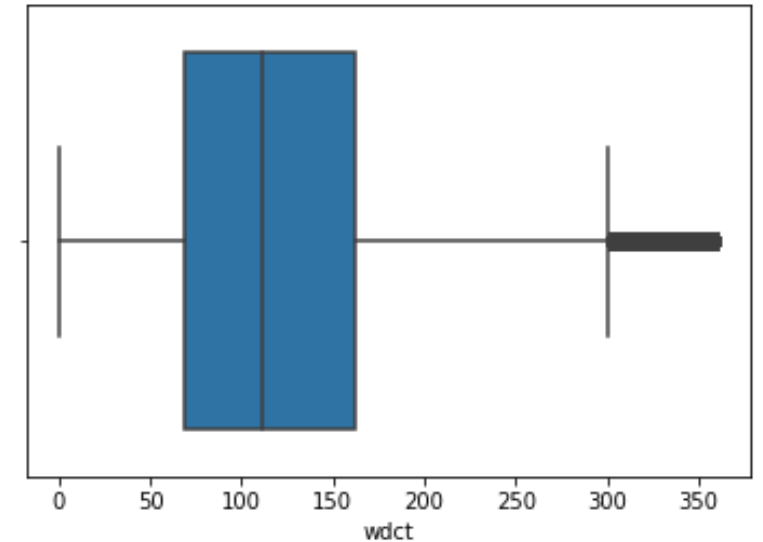
- Retrieving the value:

```
print(z[96474])
2.4265308103829626
```

<https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba>

IQR method

- The IQR (interquartile range) method is somewhat more robust (developed by John Tukey)
 - Also called H-spread
- IQR is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, $IQR = Q3 - Q1$.



Calculating IQR

- First, we can calculate the IQR for the whole dataset

```
Q1 = wsub.quantile(0.25)
```

```
Q3 = wsub.quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
print(IQR)
```

```
wsid      0.000
elvt      0.000
lat       0.000
lon       0.000
yr        5.000
mo        6.000
da       15.000
hr       12.000
prcp      0.200
stp       6.200
smax      6.100
smin      6.200
gbrd     2436.911
temp      6.500
dewp      5.800
tmax      6.900
dmax      5.600
tmin      6.000
dmin      6.100
hmdy     39.000
hmax     39.000
hmin     39.000
wdsp      1.900
wdct     93.000
gust      4.500
```


Finding outliers

```
print(wsub < (Q1 - 1.5 * IQR)) |(wsub > (Q3 + 1.5 * IQR))
```

Table 1

	city	da	date	dewp	dmax	dmin	elvt	gbrd	gust	hmax	\
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	
5	False	False	False	False	False	False	False	False	False	False	
6	False	False	False	False	False	False	False	False	False	False	
7	False	False	False	False	False	False	False	False	False	False	
8	False	False	False	False	False	False	False	False	False	False	
9	False	False	False	False	False	False	False	False	False	False	
10	False	False	False	False	False	False	False	False	False	False	
11	False	False	False	True	True	True	False	False	False	False	
12	False	False	False	True	True	True	False	False	False	False	
13	False	False	False	True	True	True	False	False	False	False	

Summary

- The problem with outliers
- Unsupervised statistical methods
- And their limitations

Summary