Week 2:
Data Preparation

Visual Data Analytics

University of Sydney

THE UNIVERSITY OF
SYDNEY

Outline

- Data frames

- Operations on data frames

- Data types

- Data profiling

Data frames

What is data?

- Data can be *structured* or *unstructured*
- Structured data has
    - Observations/ cases in rows
    - Variables/ features in columns
- Think of structured data as something that can be put into a spreadsheet.

Example

	A	B	C	D	E
1	Player	Games Played	Minutes	Points	Birthplace
2	LeBron James	1394	53142	37860	Akron, USA
3	Kevin Durant	974	35776	26565	Washington DC, USA
4	James Harden	962	33378	23915	Los Angeles, USA
5	Russell Westbrook	1054	36352	23781	Long Beach, USA
6	Chris Paul	1178	40623	21235	Winston-Salem, USA
7	Stephen Curry	852	29256	20843	Akron, USA
8	DeMar DeRozan	993	34087	20812	Compton, USA
9	Damian Lillard	734	26648	18149	Oakland, USA
10	Rudy Gay	1089	34135	17463	New York, USA
11	Paul George	765	25711	15713	Palmdale, USA
12	Kyle Lowry	1055	33750	15556	Philadelphia, USA
13	Giannis Antetokounmpo	686	22365	15283	Athens, Greece
14	Anthony Davis	629	21641	15075	Chicago, USA
15	Bradley Beal	668	23211	14772	St Louis, USA
16	Kyrie Irving	637	21756	14770	Melbourne, Australia

Current as of January 2023

US Cities and States

	A	B
1	Akron	Ohio
2	Los Angeles	California
3	Long Beach	California
4	Winston-Salem	North Carolina
5	Compton	California
6	Oakland	California
7	New York	New York
8	Palmdale	California
9	Philadelphia	Pennsylvania
10	Chicago	Illinois
11	St Louis	Missouri
12		

Observations and variables

- In the first table:
    - What are the observations?
    - What are the variables?
- How about the second table?

Pandas

- The Python library *pandas* stores data in a basic object called a *data frame*.
- It also provides functions for reading in and manipulating data frames.

NBA Data

```python
import pandas as pd
NBA = pd.read_csv('../data/NBA1.csv')
NBA
```

```
##                      Player  Games Played  Minutes  Points            Birthpl
## 0            LeBron James            1394    53142   37860              Akron,
## 1            Kevin Durant             974    35776   26565      Washington DC,
## 2             James Harden             962    33378   23915        Los Angeles,
## 3        Russell Westbrook            1054    36352   23781         Long Beach,
## 4               Chris Paul            1178    40623   21235      Winston-Salem,
## 5            Stephen Curry             852    29256   20843              Akron,
## 6           DeMar DeRozan             993    34087   20812            Compton,
## 7          Damian Lillard             734    26648   18149            Oakland,
## 8                Rudy Gay            1089    34135   17463           New York,
## 9             Paul George             765    25711   15713           Palmdale,
## 10              Kyle Lowry            1055    33750   15556       Philadelphia,
## 11   Giannis Antetokounmpo             686    22365   15283        Athens, Gre
## 12           Anthony Davis             629    21641   15075            Chicago,
## 13            Bradley Beal             668    23211   14772           St Louis,
## 14            Kyrie Irving             637    21756   14770  Melbourne, Austra
```

Keys

- A key uniquely identifies an entry into the database.

- In the first table the player name can act as a key.

- Player birthplace could not be a key since both James and Curry are born in Akron.

- However, be careful with names as keys, since two players may have the same name.

Atomicity

- Beware of variables that combine multiple pieces of information.
- Birthplace can be broken down into city and country.
- This may be useful if we want to visualise points scored by players born in USA and players born outside USA.
- Variables should be *atomic*
- This is also known as *parsing* the data.

Split

```
new = NBA.Birthplace.str.split(', ',expand = True)
new
```

```
##                   0          1
## 0            Akron        USA
## 1    Washington DC        USA
## 2      Los Angeles        USA
## 3       Long Beach        USA
## 4    Winston-Salem        USA
## 5            Akron        USA
## 6          Compton        USA
## 7          Oakland        USA
## 8         New York        USA
## 9         Palmdale        USA
## 10    Philadelphia        USA
## 11          Athens     Greece
## 12         Chicago        USA
## 13        St Louis        USA
## 14       Melbourne  Australia
```

With original table

```
NBA["CityOfBirth"] = new[0]
NBA["CountryOfBirth"] = new[1]
NBA
```

```
##                        Player  Games Played  ...      CityOfBirth  CountryOfBirth
## 0            LeBron James          1394  ...            Akron             USA
## 1            Kevin Durant           974  ...    Washington DC             USA
## 2            James Harden           962  ...      Los Angeles             USA
## 3       Russell Westbrook          1054  ...       Long Beach             USA
## 4              Chris Paul          1178  ...    Winston-Salem             USA
## 5           Stephen Curry           852  ...            Akron             USA
## 6           DeMar DeRozan           993  ...          Compton             USA
## 7          Damian Lillard           734  ...          Oakland             USA
## 8               Rudy Gay          1089  ...         New York             USA
## 9             Paul George           765  ...         Palmdale             USA
## 10             Kyle Lowry          1055  ...     Philadelphia             USA
## 11  Giannis Antetokounmpo           686  ...           Athens          Greece
## 12          Anthony Davis           629  ...          Chicago             USA
## 13            Bradley Beal           668  ...         St Louis             USA
## 14            Kyrie Irving           637  ...        Melbourne       Australia
##
```

Another example

Player	Points
Steph Curry	20843 for Warriors
Kevin Durant	17566 for Thunder, 5374 for Warriors, 3625 for Nets

Split

Player	Points for Warriors	Points for Thunder	Points for Nets
Steph Curry	20843		
Kevin Durant	5374	17566	3625

Problems

- Empty cells for Steph Curry

  - Not such a big issue (see missing data later)

- What if Kevin Durant moves to another team?

- What if we want to include Giannis (only played for Bucks)?

  - Would need new columns and code that previously worked may break.

Solution

Player	Team	Points
Steph Curry	Warriors	20843
Kevin Durant	Thunder	17566
Kevin Durant	Warriors	5374
Kevin Durant	Nets	3625

First Normal Form

- New data entries can be added by adding rows only.
- Now the player and team combined form key.
- Overall the example so far is about getting the data into the *first normal form*.
- For the purposes of visualisation the lesson is to think carefully about how the data is structured.

Operations on Data Frames

The simple machines

- Hundreds of years ago, it was believed that all machines were made up of six simple machines
- These include: levers, wheels, pulleys, screws, etc.
- Nowadays machines are more complicated.
- But this is a good metaphor for data frames

Simple machines of data

- We will consider six "simple machines" of data frames.
  - Transforming
  - Sorting
  - Filtering
  - Group by/ aggregate
  - Reshaping (melting and casting)
  - Joining (merging)
- By some combination of these we can 'munge' data frames into almost any data frame we need.

Transform

- Create a new variable based on values of existing variables.
- For example, in the NBA data frame we have games played and points.
- Suppose we want to create a variable of points per game (PPG)

Transform in Python

```
NBA["PPG"]=NBA["Points"]/ NBA["Games Played"]
NBA
```

```
##                       Player  Games Played  ...  CountryOfBirth         PPG
## 0             LeBron James          1394  ...             USA   27.159254
## 1             Kevin Durant           974  ...             USA   27.274127
## 2             James Harden           962  ...             USA   24.859667
## 3        Russell Westbrook          1054  ...             USA   22.562619
## 4               Chris Paul          1178  ...             USA   18.026316
## 5            Stephen Curry           852  ...             USA   24.463615
## 6            DeMar DeRozan           993  ...             USA   20.958711
## 7           Damian Lillard           734  ...             USA   24.726158
## 8                 Rudy Gay          1089  ...             USA   16.035813
## 9              Paul George           765  ...             USA   20.539869
## 10              Kyle Lowry          1055  ...             USA   14.745024
## 11   Giannis Antetokounmpo           686  ...          Greece   22.278426
## 12            Anthony Davis           629  ...             USA   23.966614
## 13             Bradley Beal           668  ...             USA   22.113772
## 14            Kyrie Irving           637  ...       Australia   23.186813
##
```

Sort

- Suppose we want to sort the data according to one of the variables.
- We can use the `sort_values` function.
- Consider that we want to sort by minutes played from smallest to largest.

Players by minutes

```
NBAbymin = NBA.sort_values(by = 'Minutes')
NBAbymin
```

```
##                     Player  Games Played  ...  CountryOfBirth       PPG
## 12         Anthony Davis           629  ...             USA  23.966614
## 14          Kyrie Irving           637  ...       Australia  23.186813
## 11  Giannis Antetokounmpo         686  ...          Greece  22.278426
## 13          Bradley Beal           668  ...             USA  22.113772
## 9            Paul George           765  ...             USA  20.539869
## 7         Damian Lillard           734  ...             USA  24.726158
## 5          Stephen Curry           852  ...             USA  24.463615
## 2           James Harden           962  ...             USA  24.859667
## 10            Kyle Lowry          1055  ...             USA  14.745024
## 6           DeMar DeRozan          993  ...             USA  20.958711
## 8               Rudy Gay          1089  ...             USA  16.035813
## 1           Kevin Durant           974  ...             USA  27.274127
## 3       Russell Westbrook        1054  ...             USA  22.562619
## 4              Chris Paul         1178  ...             USA  18.026316
## 0            LeBron James        1394  ...             USA  27.159254
##
```

Filter

- Filtering involves selecting only some subset of the data.
- There are many ways to do this
  - Select rows
  - Select columns
- Select by a logical condition

Example

- Suppose we only want to consider
  - Players with points per game greater than 20
  - Players born in Akron
  - Players not born in the United States
- These are all examples of logical conditions (either true or false).

Players with PPG above 20

```
NBAppg20 = NBA.loc[NBA["PPG"]>20]
NBAppg20
```

```
##                       Player  Games Played  ...  CountryOfBirth         PPG
## 0            LeBron James          1394  ...             USA   27.159254
## 1            Kevin Durant           974  ...             USA   27.274127
## 2            James Harden           962  ...             USA   24.859667
## 3       Russell Westbrook          1054  ...             USA   22.562619
## 5           Stephen Curry           852  ...             USA   24.463615
## 6            DeMar DeRozan           993  ...             USA   20.958711
## 7          Damian Lillard           734  ...             USA   24.726158
## 9             Paul George           765  ...             USA   20.539869
## 11  Giannis Antetokounmpo           686  ...          Greece   22.278426
## 12           Anthony Davis           629  ...             USA   23.966614
## 13            Bradley Beal           668  ...             USA   22.113772
## 14            Kyrie Irving           637  ...       Australia   23.186813
##
## [12 rows x 8 columns]
```

Players born in Akron

```
NBAAkr = NBA.loc[NBA["CityOfBirth"] == 'Akron']
NBAAkr
```

```
##              Player  Games Played  Minutes  ...  CityOfBirth CountryOfBirth
## 0   LeBron James             1394    53142  ...        Akron            USA  2
## 5  Stephen Curry              852    29256  ...        Akron            USA  2
##
## [2 rows x 8 columns]
```

Note that a single = denotes assignment, a double == denotes 'equals' in a logical statement.

Players born outside USA

```
NBAnonUS = NBA.loc[NBA["CountryOfBirth"] != 'USA']
NBAnonUS
```

```
##                        Player  Games Played  ...  CountryOfBirth        PPG
## 11  Giannis Antetokounmpo               686  ...          Greece  22.278426
## 14           Kyrie Irving               637  ...       Australia  23.186813
##
## [2 rows x 8 columns]
```

In general we can read ! as 'not' in Python

Group by / aggregate

- Suppose we want to compare total points scored by players country of birth.
- This requires two functions
  - The `groupby` function tells us the variable to group on (in this case Country of Birth).
  - The `agg` function tells us which variable to aggregate (in this case points)

Groupby/aggregate in Python

```
NBAg = NBA.groupby('CountryOfBirth').agg({'Points': 'sum'})
NBAg
```

```
##               Points
## CountryOfBirth
## Australia       14770
## Greece          15283
## USA            271739
```

Other ways to aggregate include mean, min and max.

Reshape

- Often in order to produce the visualisation we want we need to reshape the data.
- This is done using two functions
  - The function `melt` converts the data from wide to long.
  - The function `pivot` converts the data from long to wide.

Melting

```
NBAlong = NBA.melt(id_vars=['Player'],value_vars=['Games Played', 'Minu
NBAlong
```

```
##                           Player      variable  value
## 0               LeBron James  Games Played   1394
## 1               Kevin Durant  Games Played    974
## 2               James Harden  Games Played    962
## 3          Russell Westbrook  Games Played   1054
## 4                  Chris Paul  Games Played   1178
## 5              Stephen Curry  Games Played    852
## 6               DeMar DeRozan  Games Played    993
## 7             Damian Lillard  Games Played    734
## 8                    Rudy Gay  Games Played   1089
## 9                 Paul George  Games Played    765
## 10                 Kyle Lowry  Games Played   1055
## 11     Giannis Antetokounmpo  Games Played    686
## 12              Anthony Davis  Games Played    629
## 13                Bradley Beal  Games Played    668
## 14                Kyrie Irving  Games Played    637
## 15               LeBron James       Minutes  53142
```

Pivoting

```
NBAwide = NBAlong.pivot(index='Player', columns = 'variable')
NBAwide
```

```
##                              value
## variable            Games Played Minutes Points
## Player
## Anthony Davis                629   21641  15075
## Bradley Beal                 668   23211  14772
## Chris Paul                  1178   40623  21235
## Damian Lillard               734   26648  18149
## DeMar DeRozan                993   34087  20812
## Giannis Antetokounmpo        686   22365  15283
## James Harden                 962   33378  23915
## Kevin Durant                 974   35776  26565
## Kyle Lowry                  1055   33750  15556
## Kyrie Irving                 637   21756  14770
## LeBron James                1394   53142  37860
## Paul George                  765   25711  15713
## Rudy Gay                    1089   34135  17463
## Russell Westbrook           1054   36352  23781
```

A better example

```
Sydney = pd.read_csv('../data/SydneyClimate.csv')
Sydney
```

```
##     Year    M01    M02    M03    M04    M05    M06    M07    M08    M09    M10    M11
## 0   2017    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN   24.9   24.8
## 1   2018   28.5   28.1   27.4   26.7   22.2   17.7   19.9   19.3   21.0   21.9   25.0
## 2   2019   29.6   27.7   26.9   25.1   22.7   18.6   19.8   19.5   22.0   24.7   27.0
## 3   2020   29.0   27.5   25.6   24.5   20.3   18.7   18.2   19.5   22.7   24.2   26.1
## 4   2021   27.4   26.7   25.5   24.3   20.9   18.1   18.2   20.7   22.5   24.2   23.2
## 5   2022   27.7   26.7   24.8   23.7   20.8   18.3   17.0   19.8   20.6   22.6   24.3
```

Retrieved from Bureau of Meteorology

Melting

```
Sydlong = Sydney.melt(id_vars='Year').sort_values(by = ['Year','variable
Sydlong
```

```
##       Year variable  value
## 0    2017      M01    NaN
## 6    2017      M02    NaN
## 12   2017      M03    NaN
## 18   2017      M04    NaN
## 24   2017      M05    NaN
## ..    ...      ...    ...
## 47   2022      M08   19.8
## 53   2022      M09   20.6
## 59   2022      M10   22.6
## 65   2022      M11   24.3
## 71   2022      M12    NaN
##
## [72 rows x 3 columns]
```

Plot

38
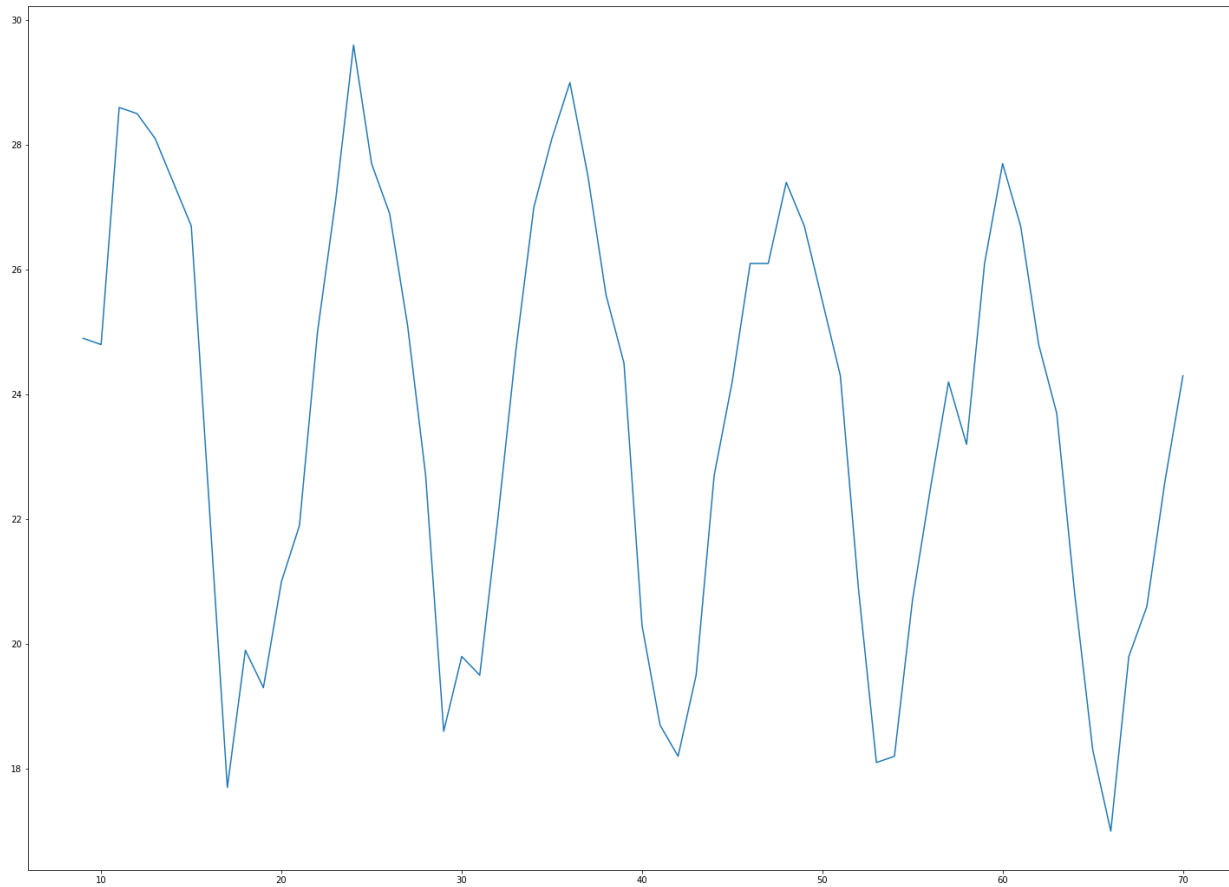
Merge

- Bring two data frames together
- Similar to a VLOOKUP type function in spreadsheet programs such as Excel.
- We can use `merge` to add information about State of birth for NBA players.

Merge

```
CitiesStates = pd.read_csv('../data/UScitiesstates.csv')
NBAmerge = pd.merge(NBA,CitiesStates, left_on = 'CityOfBirth', right_on
NBAmerge
```

```
##                    Player  Games Played  ...             City           State
## 0           LeBron James          1394  ...            Akron            Ohio
## 1          Stephen Curry           852  ...            Akron            Ohio
## 2            James Harden           962  ...      Los Angeles      California
## 3       Russell Westbrook          1054  ...       Long Beach      California
## 4              Chris Paul          1178  ...    Winston-Salem  North Carolina
## 5           DeMar DeRozan           993  ...          Compton      California
## 6          Damian Lillard           734  ...          Oakland      California
## 7               Rudy Gay          1089  ...         New York         New York
## 8            Paul George           765  ...         Palmdale      California
## 9             Kyle Lowry          1055  ...     Philadelphia     Pennsylvania
## 10          Anthony Davis           629  ...          Chicago         Illinois
## 11            Bradley Beal           668  ...         St Louis         Missouri
##
## [12 rows x 10 columns]
```

Different type of merge

- Left: Keep all entries from first data frame
- Right: Keep all entries from first data frame
- Inner: Keep all entries that appear in both data frames
- Outer: Keep all entries that appear in either data frame

Merge

```
CitiesStates = pd.read_csv('../data/UScitiesstates.csv')
NBAmerge = pd.merge(NBA,CitiesStates, how = 'outer', left_on = 'CityOfB
NBAmerge
```

```
##                       Player  Games Played  ...            City          State
## 0            LeBron James          1394  ...           Akron           Ohio
## 1           Stephen Curry           852  ...           Akron           Ohio
## 2            Kevin Durant           974  ...             NaN            NaN
## 3            James Harden           962  ...     Los Angeles     California
## 4       Russell Westbrook          1054  ...      Long Beach     California
## 5              Chris Paul          1178  ...   Winston-Salem  North Carolina
## 6            DeMar DeRozan           993  ...         Compton     California
## 7          Damian Lillard           734  ...         Oakland     California
## 8               Rudy Gay          1089  ...        New York       New York
## 9             Paul George           765  ...        Palmdale     California
## 10             Kyle Lowry          1055  ...    Philadelphia   Pennsylvania
## 11    Giannis Antetokounmpo           686  ...             NaN            NaN
## 12          Anthony Davis           629  ...         Chicago       Illinois
## 13            Bradley Beal           668  ...        St Louis       Missouri
## 14            Kyrie Irving           637  ...             NaN            NaN
```

Putting them together

- In your own time, construct data frames for the following following:
  - Each state and with the maximum points per minute (PPM) by a player from each state.
  - The same as above with states ranked from highest to lowest according to the maximum PPM.
  - The same as above but with an extra column with the player name of the player with the highest PPM in each state.
- There may be more than one correct answer.

Data types

Data Types

- Each variable measures a certain characteristic.

- Characteristics can be measured in different ways

- This leads to *data type* which are important for understanding

  - How we can transform data.

  - The correct visualisation to use.

Scales of measurement

- Any old (or new) statistics textbook will introduce four scales of measurement
    - Nominal
    - Ordinal
    - Interval
    - Ratio
- These are still useful (with some caveats).

Nominal data

- Tells us something about a characteristic but there is no notion of having more or less of a characteristic.
- Example: Country of birth.
- Can you think of other examples?
- Even if we assign numbers to nominal categories, it does not make sense to find means medians etc.
- The mode still makes sense.

Ordinal data

- Tells us whether we have more or less of a characteristic, but not how much more or less.
- Example: rate players as good, very good, excellent.
- If we assign numbers to nominal categories it still does not make sense to add subtract these numbers.
- However the median (and the mode) still make sense.

Interval/Ratio data

- All numerical data is either interval or ratio data.
- The differences between the two concern whether the zero point of the scale truly represents an absence of the characteristic being measured.
- Best understood with an example.

Points per game

- Suppose I constructed a new index for points per game (PPG) where a PPG of 20 now becomes a PPG of 0.
    - Paul George (PPG: 20.5) would have a "new" PPG of 0.5.
    - Steph Curry (PPG: 24.5) would have a "new" PPG of 4.5.
- Does this mean that Curry is scoring 9 times as much as George? No.

The textbook example

- The famous example is temperature.
- The Celsius scale attaches 0 and 100 to the freezing and boiling point of water.
  - This is arbitrary
- For the Kelvin scale, zero is true zero since it is a temperature where atoms have no energy (loosely speaking).

Does it matter?

- Not that much (outside of science).

- Most data we see in business are ratio data.

- In general, I will use *numeric data* and *ratio data* interchangeably.

- Just think carefully when dividing with numerical variables.

Summary

Operation	Nominal	Ordinal	Interval	Ratio
Equality	✔	✔	✔	✔
Order		✔	✔	✔
Add / subtract			✔	✔
Multiply / divide				✔
Mode	✔	✔	✔	✔
Median		✔	✔	✔
Arithmetic mean			✔	✔
Geometric mean				✔

Some exceptions

- Nominal data with two categories.

  - Born in US assigned 1, born outside US assigned 0.

  - Arithmetic mean is then the *proportion* born in US.

- Likert (customer satisfaction) scales:

  - Strongly disagree = 1, Disagree = 2, etc.

  - Using an arithmetic mean is controversial but common in practice.

- Time is very unusual since calendar effects are important in business.

Types in Pandas

```
NBA.dtypes
```

```
## Player              object
## Games Played         int64
## Minutes              int64
## Points               int64
## Birthplace          object
## CityOfBirth         object
## CountryOfBirth      object
## PPG                 float64
## dtype: object
```

Object is text, int64 is an integer and float64 is a real number.

Data Profiling

Some issues

- Duplicated entries
  - Can be removed using `drop_duplicates` function in pandas.
- Data entry errors
  - For example cannot score 20000 points in 3 minutes.
  - Requires domain knowledge.
  - These issues can be discovered during visualisation.

Standardisations

- Steph Curry may appear elsewhere in the data as
    - "Stephen Curry"
    - "Wardell Stephen Curry II" (his full name).
- Similar things happen with company names Facebook/ Meta, General Motors/ GM etc.
- Requires domain knowledge and some extra coding.

Missing Data

- There are often missing data

- Need to think of a good strategy to encode missing data.

- In Python there is `NaN` for this.

- Often will replace with a number but this is a bad strategy

Do not use "0" for "missing"

- Suppose if games played is missing and we replace with zero.

- The zeros will distort the mean.

- If we compute points per game there will be division by zero.

- Do not replace missing values by zeros.

Do not use "-999" for "missing"

- Sometimes a completely implausible number such as -999 is used to denote missing.
- This can lead to strange visualisations
- The following example is for the Sydney temperature data
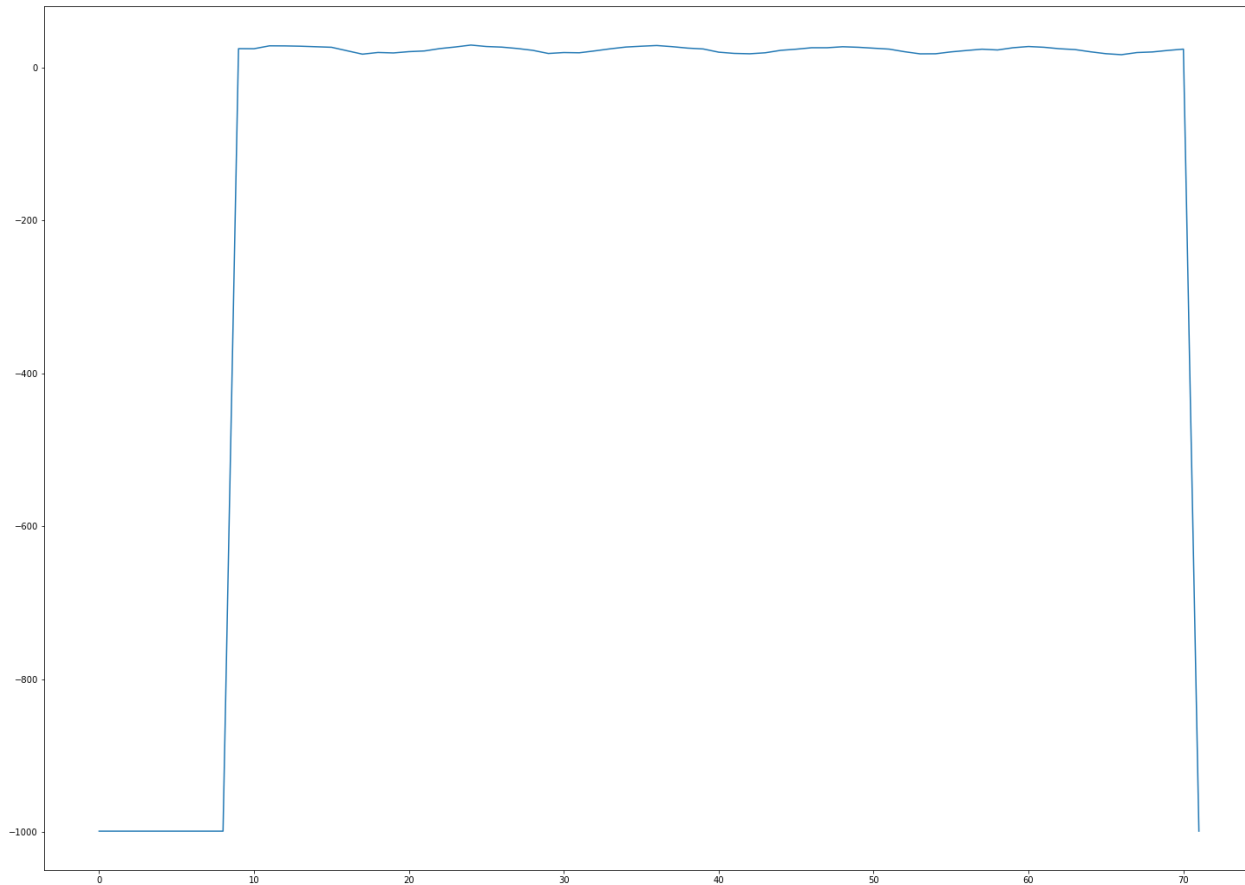
Dealing with missing data

- Only use complete cases

- Impute missing values

  - With a random value

  - With mean/median or mode

  - More complicated models

- Report/Visualise missing data

  - By reporting missing data this gives a better idea of uncertainty.

Report missing

Do you approve or disapprove of the job Anthony Albanese is doing as Prime Minister?

	TOTAL	Labor	TOTAL: Coalition	Greens	Minor parties/ Independents	
Strongly approve	18%	30%	10%	22%	9%	
Approve	42%	58%	31%	49%	29%	
Disapprove	16%	5%	26%	13%	23%	
Strongly disapprove	12%	0%	22%	2%	26%	
Don't know	13%	7%	11%	14%	13%	
TOTAL: Approve	60%	88%	41%	71%	39%	
TOTAL: Disapprove	28%	5%	48%	15%	49%	

Source: Guardian

Wrap-up

Conclusions

- Our focus from now on will be visualisation.

- Writing code to visualise messy data is hard.

- Spend the time to clean your data.

- Focus on the *principles* since these work for Excel, Tableau, R, Python, etc.

- After you appreciate the principles, practice your coding.

Questions