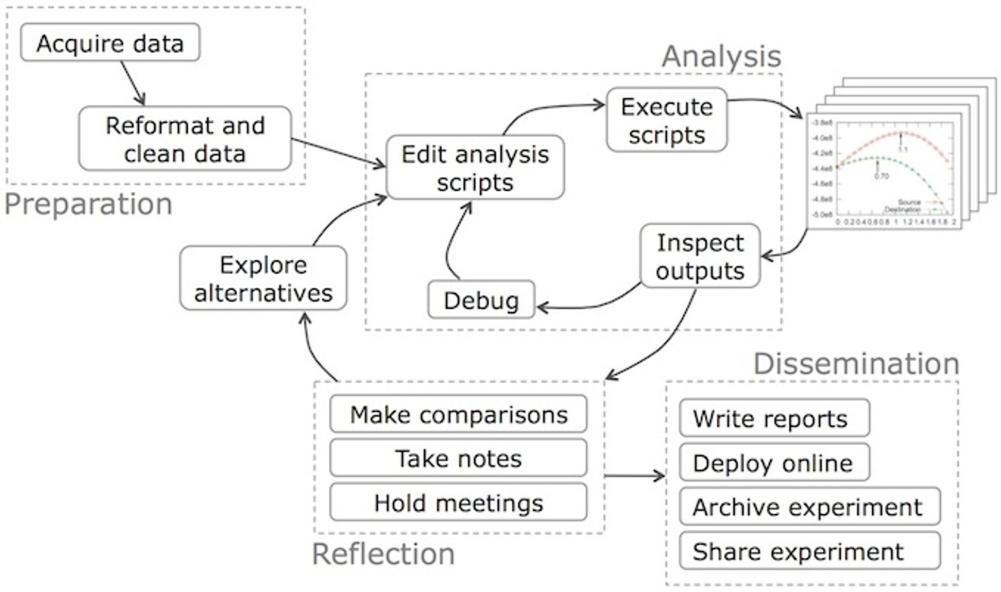
Week 01 - Summary

What is Data Science

- The workflow…



- Analysis requires a loop so we can understood the insights deeper

Types of Data and Levels of Measurement

Categorical Data

- A categorical variable is also known as discrete or qualitative variable and can have two or more categories
- It is further divided into two variants, nominal and ordinal
    - These variables are sometimes coded as numerical values, or as strings

Nominal Data

- This is an unordered category data
- This type of variable may be “label-coded” in numeric form but these numerical values have no mathematical interpretation and are just labelling to denote categories
- For example, colours: black, red and white can be coded as 1, 2 and 3

Dichotomous Data

- This is a type of nominal data that can only have two possible values, e.g. true or false, or presence or absence
- These are also sometimes referred as binary or Boolean variables

Ordinal Data

- This is ordered categorical data in which there is strict order for comparing the values, so a labelling as numbers is not completely arbitrary
- For example, human height (small, medium and high) can be coded into numbers small = 1, medium = 2, high = 3

Interval Data

- It is a variable in which the interval between values has meaning and there is not true zero value
    - Values encode differences
    - Equal intervals between values
    - No true zero
    - Addition is defined

Ratio Data

- It is a variable that might have a true value of zero and represents the total absence of the variable being measured

- For example, it makes sense to say Kelvin temperature of 100 is twice as hot as a Kelvin temperature of 50 because it represents twice as much the thermal energy

Data Acquisition and Data Cleaning

File Access

- Organisational data is ‘dirty’ with typical formats: csv, excel and XML

Programmatically

- HTML
- APIs of Web Services

Database Access or collect data yourself through survey

Approach 1: Spreadsheet Software

Approach 2: Specific Data Cleaning Tools

Exploratory Data Analysis (with Spreadsheets)

Summarising Nominal Data with Bar Charts

Measures of central tendency:

- mode
  - The most frequent value
  - Defined for nominal data, but spreadsheets might not compute
  - Can read from a bar chart

Measures of dispersion:

- counts / distribution

- of count frequency of each category

Visualisation: Bar Chart

Summarising Ordinal Data: Histograms, median, percentiles

Measures of central tendency:

- median, mode

Measures of dispersion:

- counts/distribution
- min/max/range
- percentiles

Visualisation: Histogram Chart

Summarising Ratio (and Interval) Data

Measures of central tendency:

- mean, median, mode

Measures of dispersion:

- counts/distribution
- min/max/range
- percentile
- stdev/variance

Visualisation: Scatter Plot

Exploratory Data Analysis with Python

Python Import System

```
import csv
for row in csv.reader(['one, apple, green', 'two, tomato, red']):
  print(row[1])
```

```
# alternatively...
import csv as X
from csv import DictReader
```

Data Acquisition and Cleaning with Python

Read Data into Python using csv

`pprint` module needed for pretty print complex data structure

- `pprint` formats a dictionary read by csv so it's easer to read

```
import csv
import pprint
data = list(csv.DictReader(open('MajorPowerStations.csv')))
pprint.pprint(data[0])
```

Pandas: Fix Missing Values During Import

```
import pandas as pd

missing_vaues = ["--", "<Null>"]
data = pd.read_csv('MajorPowerStations.csv', na_values = missing_values)
data.head()
```

Pandas - Missing Data Handling

- Pandas provides various functions for handling missing/wrong data
  - `csv_read()` automatically replaces missing values with NA/NaN
- Other examples:
  - `DataFrame.dropna()` removes rows with any missing values
  - `DataFrame.fillna()` fill NA/NaN values using a specified method
  - `DataFrame.replace()` replace values

```
data2 = data['numGen'].dropna()

data['numGen'].fillna(0, inplace = True)

data['numGen'].replace(to_replace = '<Null>', value = 0, inplace = True)
```

Cleaning Data: Convert to Correct Types

- The standard Python csv module reads everything as string types
- Pandas is a bit better, but still will fallback to string if it can't deduce the type from all values in a column
- Need to convert as appropriate:
  - int() creates integer objects, e.g., -1, 101
  - float() creates floating point object, e.g., 3.14, 2.71
  - datetime.strptime() creates date time objects from strings

Pandas Typing

```
import pandas as pd
data = pd.read_csv('MajorPowerStations.csv')

data['numGenerator'] = data['numGenerator'].astype(int)
data['powerOutput'] = data['powerOutput'].astype(float)
```

Tip:

Always a good idea to retain a copy of your work in Jupyter Notebook by using

`new_data = old_data.copy()`

Descriptive Statistics with Python/Pandas

Pandas - Data Structures

- Two main data structures:
  - **Series** (1-dimensional, labeled, homogeneous typed)
  - **DataFrame** (2-dimensional, labeled, (potentially) heterogeneous columns)

```
data.axes
data.columns
data.dtypes
data['name'].count()
```

Filtering

- You can filter entries in a DataFrame using `loc[]`
  - Difference between `loc[]` and `iloc[]`:
    - loc is typically used for label indexing and can access multiple columns, while iloc is used for integer indexing

```
# list of all photovoltaic solar power stations
solarStations = data.loc[data['type'] == 'Solar Photovoltaic']

# total power capacity of thermal solar stations
data.loc[data['type'] == 'Solar Terminal', 'power'].sum()

# list of all large (>100 MW) wind power stations
largeWindParks = data.loc[(data['type'] == 'Wind Turbine') & (data['power'] > 100)]
```

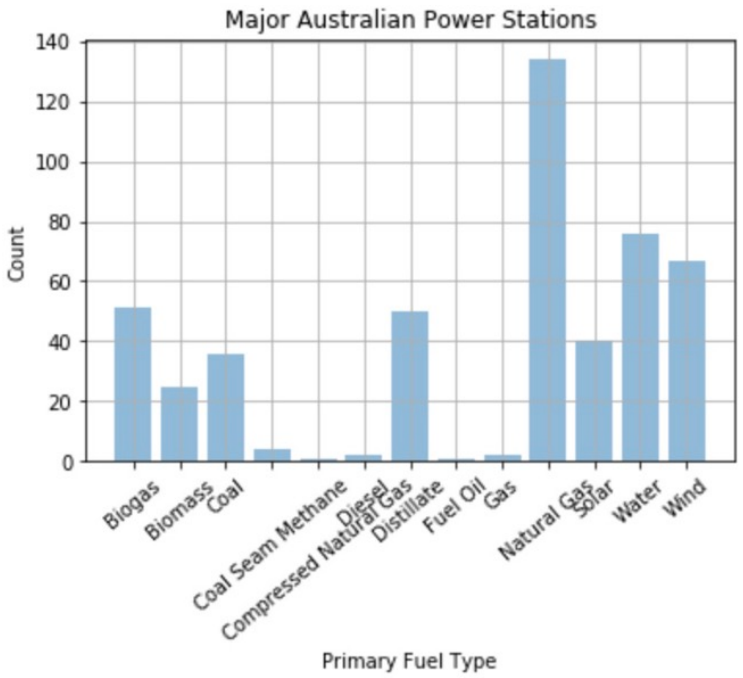
Frequency Distributions using groupby() and size()

```
classDistr = data.groupby('class').size()
print(classDistr)
```

Data Visualisation with Python

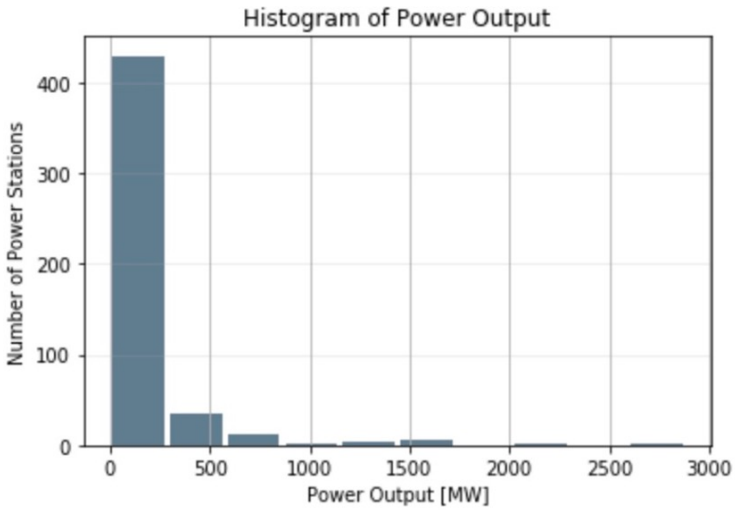
Creating a Bar Chart (for nominal/categorical data)

```
%matplotlib inline
fuelTypeDistr = wrkData.groupby('fueltype').size().reset_index(name = 'numStations')
# Use groupby() to get frequency distribution, rename resulting counts as 'numStations'

# Plot
plt.bar(fuelTypeDistr['fueltype'], fuelTypeDistr['numStations'], alpha = 0.5, align = 'center')
plt.xticks(rotation = 40)
plt.title('Major Australian Power Stations')
plt.xlabel('Primary Fuel Type')
plt.ylabel('Count')
plt.grid()
```



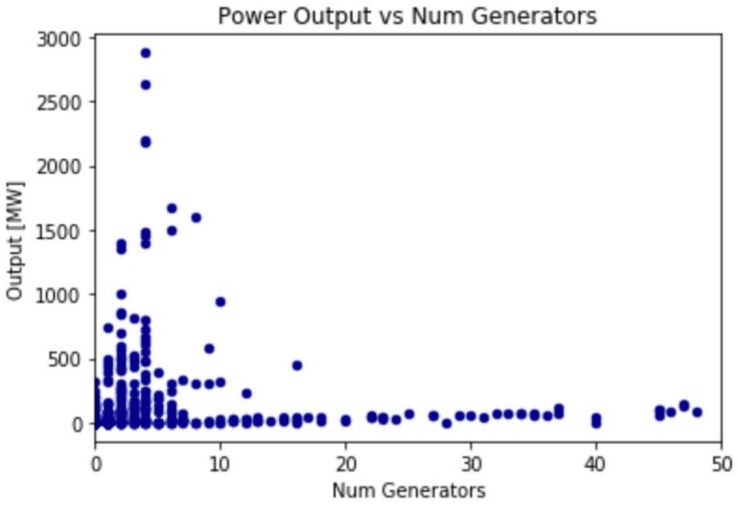
Plotting a Histogram (for continuous variables)

```
pyExpFreq = wrkData['power'].hist(bins = 10, width = 0.9, color = '#607c8e')
plt.title('Histogram of Power Output')
plt.xlabel('Power Output [MW]')
plt.ylabel('Number of Power Stations')
plt.grid(axis = 'y', alpha = 0.25) # alpha adjusts size of the grids
```



Creating a Scatter Plot

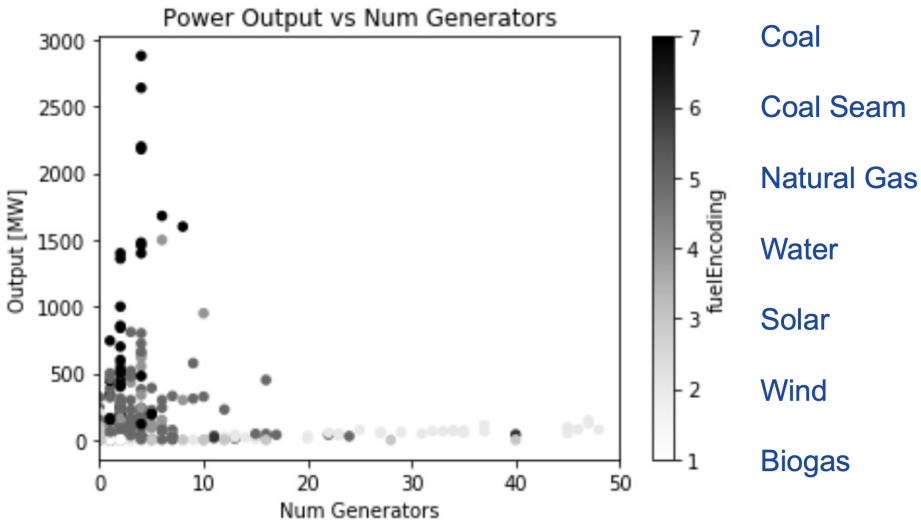
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure()
sub = plt.subplot()
wrkData.plot.scatter(x = 'numGen', y = 'power', c = 'DarkBlue', ax = sub)
sub.set_xlim(0, 50)
plt.title('Power Output vs Num Generators')
plt.xlabel('Num Generators')
plt.ylabel('Output [MW]')
```



Creating a Scatter Plot with variable colouring/grayscale

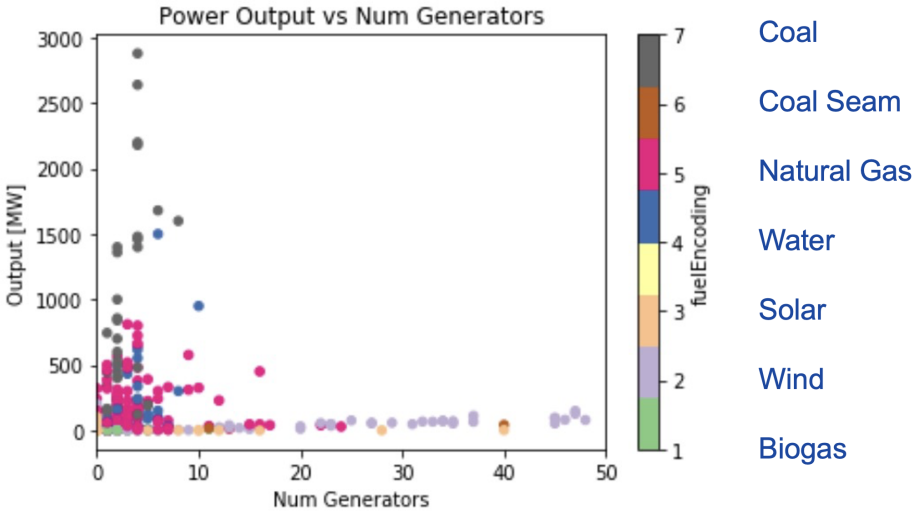
```
# assign colours to some selected fuel types
# the numbers and the order chosen are up-to-you
# we have chosen an order that works well with the colour schemes used in the subsequent plots

wrkData['fuelEncoding'] = wrkData['fueltype'].map({
  'Biogas':1,
  'Wind':2,
  'Solar':3,
  'Water':4,
  'Natural Gas':5,
  'Coal Seam Machine':6,
  'Coal':7
})
```

```
# Now we can use this encoding column to colour our plot
%matplotlib inline

fig = plt.figure()
sub = plt.subplot()
wrkData.plot.scatter(x = 'numGen', y = 'power', c = 'fuelEncoding', ax = sub)
# colour by encoding value
sub.set_xlim(0, 50)
plt.title('Power Output vs Num Generators')
plt.xlabel('Num Generators')
plt.ylabel('Output [MW]')
```



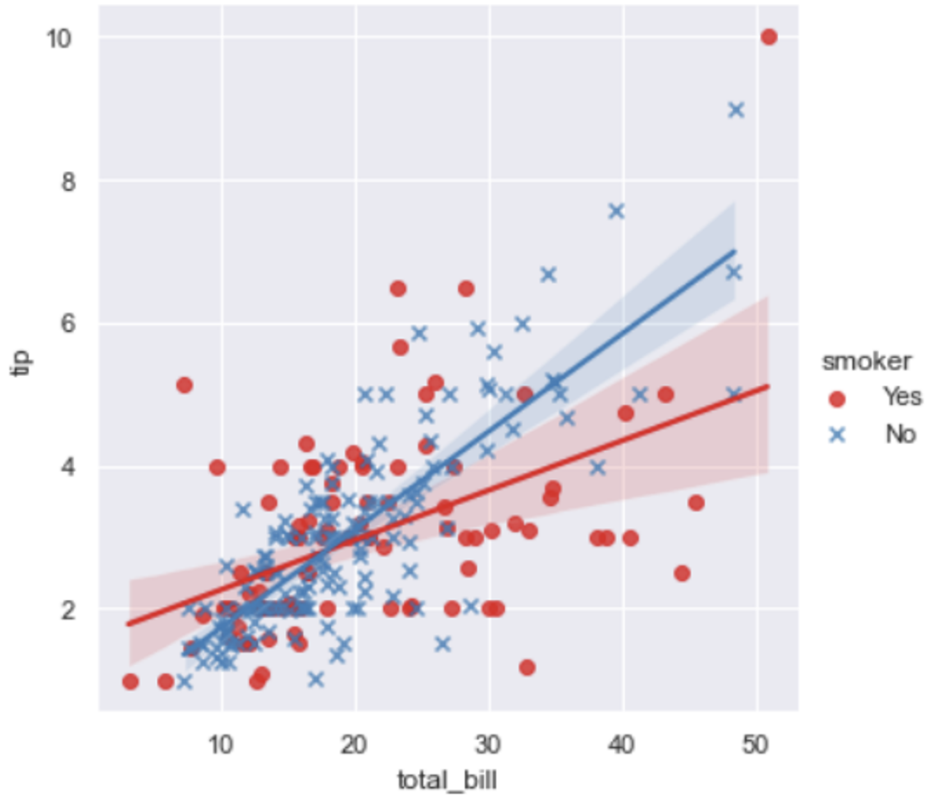
Creating a Scatter Plot with specific colormap

```
# the same plot as before, but use a more vivid colour scheme (colormap = 'Accent')
%matplotlib inline

fig = plt.figure()
sub = plt.subplot()
wrkData.plt.scatter(x = 'numGen', y = 'power', c = 'fuelEncoding', colormap = 'Accent',
ax = sub)
sub.set_xlim(0, 50)
plt.title('Power Output vs Num Generators')
plt.xlabel('Num Generators')
plt.ylabel('Output [MW]')
```



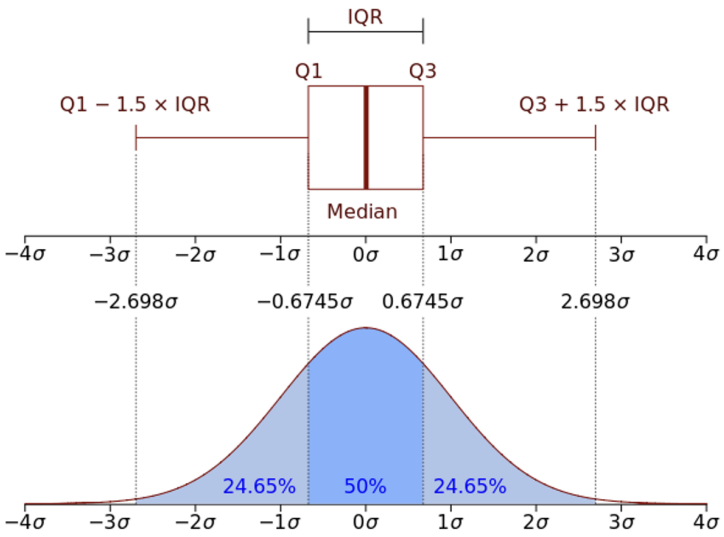
Improving Visualisation: Seaborn library

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(color_codes = True)
tips = sns.load_dataset("tips")
sns.lmplot(x = "total_bill", y = "tip", hue = "smoker", data = tips, markers =
["o", "x"], palette = "Set1");
```



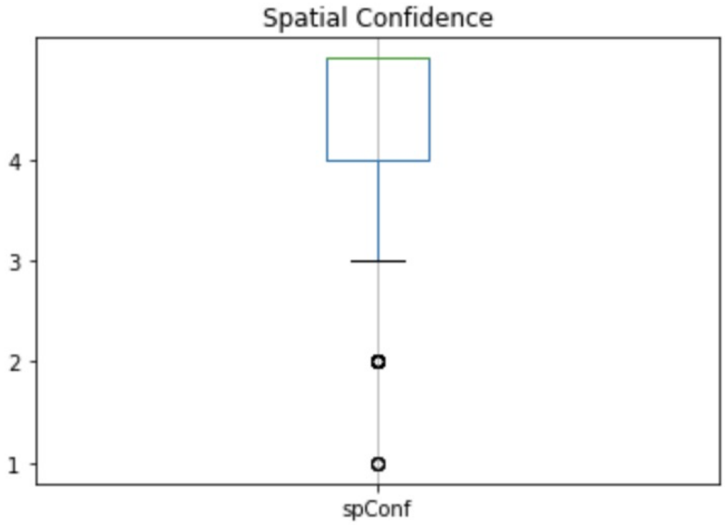
Bot Plots

- A boxplot graphically summarises quantitative data by means of the following five numbers:
  - the median value
  - the first and the third quartiles (Q1 and Q3)
  - and the minimum and the maximum values as lower and upper fence
  - Alternatively, we can express the value range by means of 1.5 IQR (inter-quartile range(IQR)) around the first and third quartiles
  - Values outside fences are outliers



Boxplots using Pandas

```
%matplotlib inline

plt.yticks(np.arange(1, 5, 1.0))
fig = wrkData.boxplot(['spConf']).set_title('Spatial Confidence')
plt.grid(axis = 'y', alpha = 0) # disable grid lines
```



- spatial confidence on a likert scale of 1 to 5; 5 representing highest confidence in location data of power station