Data Analysis with Python
Pandas Training v6

Trainer: Marcus Lee

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

About the Trainer

Marcus Lee has his degree in Computer Science and a background in Statistics from the University of Otago. Before returning to Singapore, he analysed vacation data provided by the New Zealand Board of Tourism to determine the favourite activities of Australian, Japanese, and German tourists in New Zealand. In addition to a vast number of other demographics statistics, he has been able to provide significant advice to the board on how to promote tourism in New Zealand. His core specialization skills are Java, R, Statistical Analysis, Machine Learning, NumPy, Scikit, and Network Management. He has also a fair amount of experience in C, C++, and Python

Ground Rules

- Set your mobile phone to silent mode
- Participate actively in the class. No question is stupid.
- Mutual respect. Agree to disagree.
- One conversation at one time.
- Be punctual. Back from breaks on time.
- Exit the class silently if you need to step out for phone call, toilet break etc.
- 75% attendance is required

Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.

Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

Prerequisite

This course assumes the following knowledge

- Basic Python

Agenda

Topic 1 Data Preparation

- Data Analytics with Pandas
- Pandas DataFrame and Series
- Import and Export Data
- Filter and Slice Data
- Clean Data

Topic 2 Data Transformation

- Join Data
- Transform Data
- Aggregate Data

Topic 3 Data Visualization

- Data Visualization with Matplotlib and Seaborn
- Visualize Statistical Relationships with Scatter Plot
- Visualize Categorical Data with Bar Plot
- Visualize Correlation with Pair Plot and Heatmap
- Visualize Linear Relationships with Regression

Agenda

Topic 4 Data Analysis

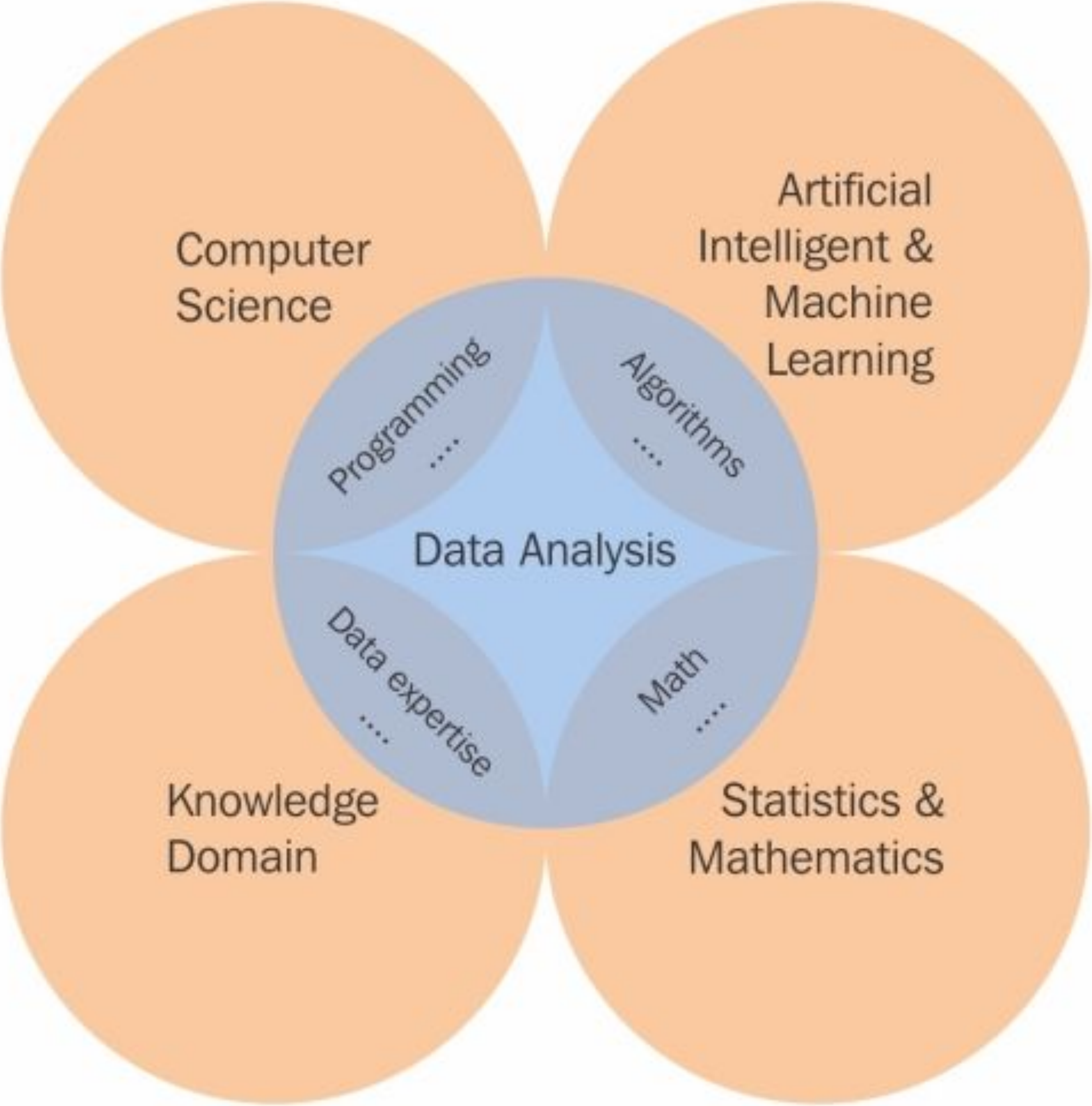
- Statistical Data Analysis
- Time Series Analysis

Google Classroom

- The resources can be found on the Google classroom
- Goto https://classroom.google.com and enter the class code below
- If you have problem to access the Google Classroom, please inform trainer or the staff

gdsag5n

Topic 1
Data Preparation


Computer Science
Artificial Intelligent & Machine Learning
Programming
....
Algorithms
....
Data Analysis
Data expertise
....
Math
....
Knowledge Domain
Statistics & Mathematics

Data Analysis Steps

- Data Preparation / Processing
  - Data Collection
  - Data Pre-Processing
- Data Analytics
  - Data Visualization
  - Data Analysis and Exploration
- Data Modeling
  - Create Model
  - Train (Fit) Model
  - Deploy Model

Python Libraries for Data Analysis

- Data Processing and Analysis
  - Numpy
  - Pandas
- Data Visualization
  - Matplotlib
  - Seaborn
- Data Modeling
  - Scikit Learn

Google Colab for Python

- Google Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud.
- With Colaboratory you can write and execute Python and Tensorflow code, save and share your analyses
- You can also access computing resources such as GPU and TPU for free
- You can access Google Colab by typing

  https://colab.research.google.com

Access Google Drive from Colab

To mount google drive locally, refer to this link https://colab.research.google.com/notebooks/io.ipynb

```
from google.colab import drive
drive.mount('/content/gdrive')
```

To access your Google Drive :

```
ls  '/content/gdrive/My Drive/'
```

Install Python Packages

```
pip install numpy
pip install matplotlib
pip install pandas
pip install scipy
pip install sklearn
```

For mac user, use pip3 instead of pip

Import Python Packages

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import pandas as pd
import sklearn
```

Pandas

- Data processing is important part of analyzing the data, because data is not always available in desired format.
- Various processing are required before analyzing the data such as cleaning, restructuring or merging etc.
- Pandas are built on the top of Numpy.
- Pandas provides rich set of functions to process various types of data
- Pandas integrates well with matplotlib library, which makes it very handy tool for analyzing the data.

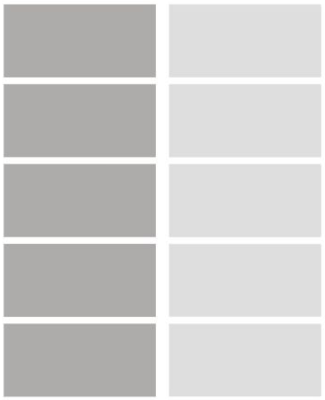
pandas

DataFrame

- DataFrame is the key data structure in Python, similar to the Dataframe in R
- DataFrame behave likes Excel spreadsheet
- It has rows index and columns name

Series

- Pandas provides two very useful data structures to process the data i.e. Series and DataFrame
- Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).
- The axis labels are collectively referred to as the index.
- The basic method to create a Series is to call:.
  s = pd.Series(data, index=index)
- Example
  data = np.array([10,20,30,40])
  s = pd.Series(data,index=['2011','2012','2013',



Create Series

- Series can be instantiated from dicts
- Example

```
data = {'2011':40,'2012':30,'2013':20,'2014':10}
s = pd.Series(data)
```

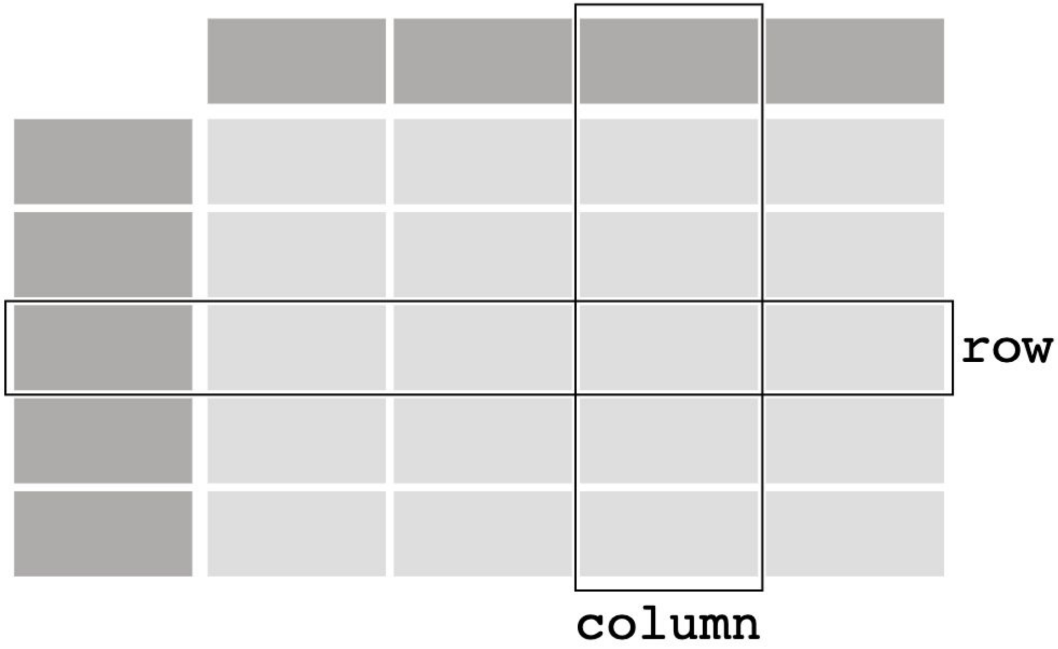
Retrieve Series Data

- A Series is like a fixed-size dict in that you can get and set values by index label.
- Example:

```
data = {'2011':40,'2012':30,'2013':20,'2014':10}
s = pd.Series(data)
s['2012']
```

Data Frame

- DataFrame is the widely used data structure of pandas.
- .DataFrame has two different index i.e. column-index and row-index.
- You can think of it as an SQL table or a spreadsheet data representation.



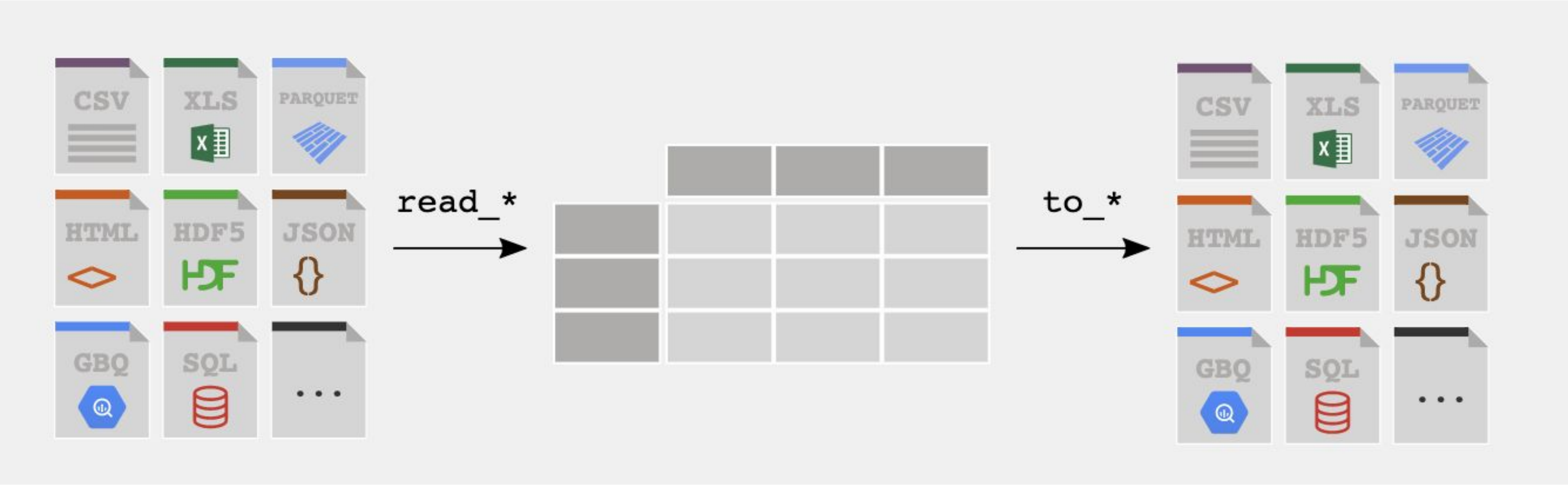
Create Dataframe

- The most common way to create a DataFrame is by using the dictionary of equal-length list as shown below:

```
data = {
    'Name' : ["Ally","Belinda","Jane","Steve"],
    'Height' : [160,165,155,180],
    'Gender' : ['F','F','F','M']
}
df = pd.DataFrame(data)
```

Import and Export Data

- Pandas supports the integration with many file formats or data sources out of the box (csv, excel, sql, json, parquet,…).
- Importing data from each of these data sources is provided by function with the prefix read_*. Similarly, the to_* methods are used to store data.



Import and Export CSV Data

- Pandas provides the read_csv() function to read data stored as a csv file into a pandas DataFrame. Eg

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

- To create the index and select data from imported data, Eg

  ```
  df = pd.read_csv('mtcars.csv',
  index_col = 'car_names',
  usecols = ['car_names','mpg','cyl','hp'])
  ```

- Whereas read_* functions are used to read data to pandas, the to_* methods are used to store data. Eg

  ```
  mtcars_sample.to_csv('cars_sample.csv')
  ```

Import and Export Excel Data

- The to_excel() method stores the data as an excel file.
- By setting index=False the row index labels are not saved in the spreadsheet. Eg

```
    mtcars_sample.to_excel('cars_sample.xlsx',
sheet_name='cars', index=False)
```

- The equivalent read function read_excel() will reload the data to a DataFrame. Eg

```
    mtcars_sample2 = pd.read_excel('cars_sample.xlsx',
sheet_name='cars')
```

DataFrame Attributes

mtcars.info() : Information of the dataframe

df.shape : Shape of a dataframe

df.columns : columns of a dataframe

df.index : index of a a dataframe

df['col]'.values : values of a particular column

Activity: Import Data

- Import the Singapore Health Expenditure dataset from

https://raw.githubusercontent.com/tertiarycourses/datasets/master/government-health-expenditure.csv

- Use financial year as index
- Import only operating, development and government health expenditure
- Export the data to csv format

Head and Tail

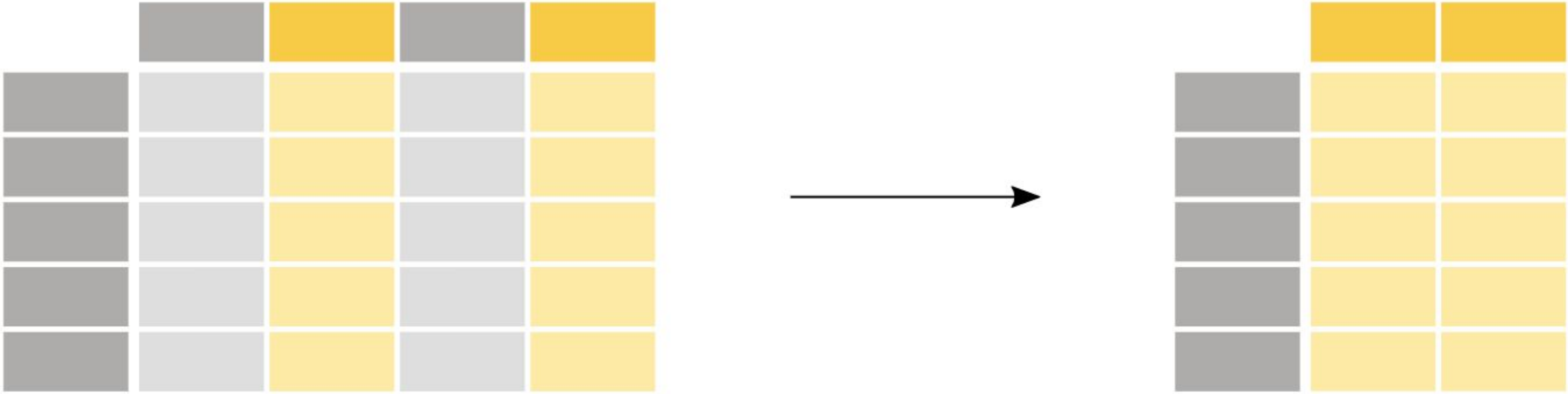
- To view a small sample of a Series or the DataFrame object, use the head() and the tail() methods.
- head() returns the first n rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.
- tail() returns the last n rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.
- Example

  ```
  mtcars.head(10)
  mtcars.tail()
  ```

Select Column

- To select a single column, use square brackets [] with the column name of the column of interest.T he returned data type is a pandas Series Eg cars_sample['mpg']

- To select multiple columns, use a list of column names within the selection brackets []. The returned data type is a pandas DataFrame. Eg cars_sample[['mpg','cyl']]



Select Row

- The Python and NumPy indexing operators "[ ]" and attribute operator "." provide quick and easy access to Pandas data structures across a wide range of use cases.
- Pandas now supports three types of Multi-axes indexing; the three types
  - .loc() - Label based
  - .iloc() - Integer based
- Example:
  - mtcars_sample.loc['Fiat 128']
  - mtcars_sample.loc[['Fiat 128','Lotus Europa']]
  - mtcars_sample.iloc[3]
  - mtcars_sample.iloc[[3,5]]

Slicing Data

- iloc can be used to slice out a subset of the data. Example

  mtcars_sample.iloc[3:6]

  mtcars_sample.iloc[:5]

Activity: Selecting and Slicing Data

- Import the Singapore Health Expenditure dataset from

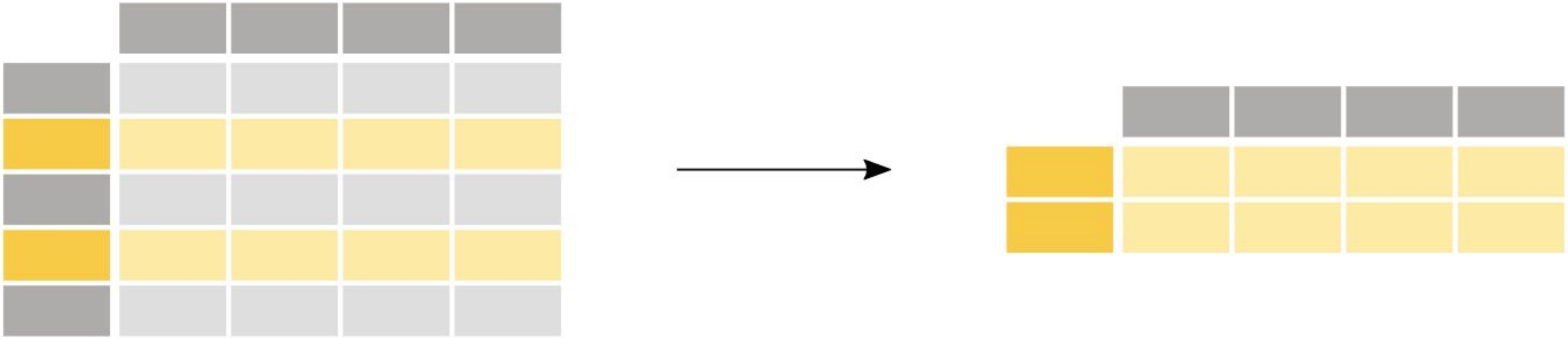
https://raw.githubusercontent.com/tertiarycourses/datasets/master/government-health-expenditure.csv

- Retrieve the operating and development expenditure data from the year 2016 and 2017
- Slice out the operating and development expenditure data from 2009 to 2013

Filtering Data

- To select rows based on a conditional expression, use a condition inside the selection brackets [].
- Some examples are as follows:

```
mtcars_sample[mtcars_sample['cyl']>4]
mtcars_sample[(mtcars_sample["mpg"] > 20) | (mtcars_sample["cyl"] < 6)]
mtcars_sample[mtcars_sample["am"] == 1]
mtcars_sample.loc[["Mazda RX4", "Fiat 128"], :]
mtcars_sample[mtcars_sample['cyl'].isin([6,8])]
```



Activity: Filtering Data

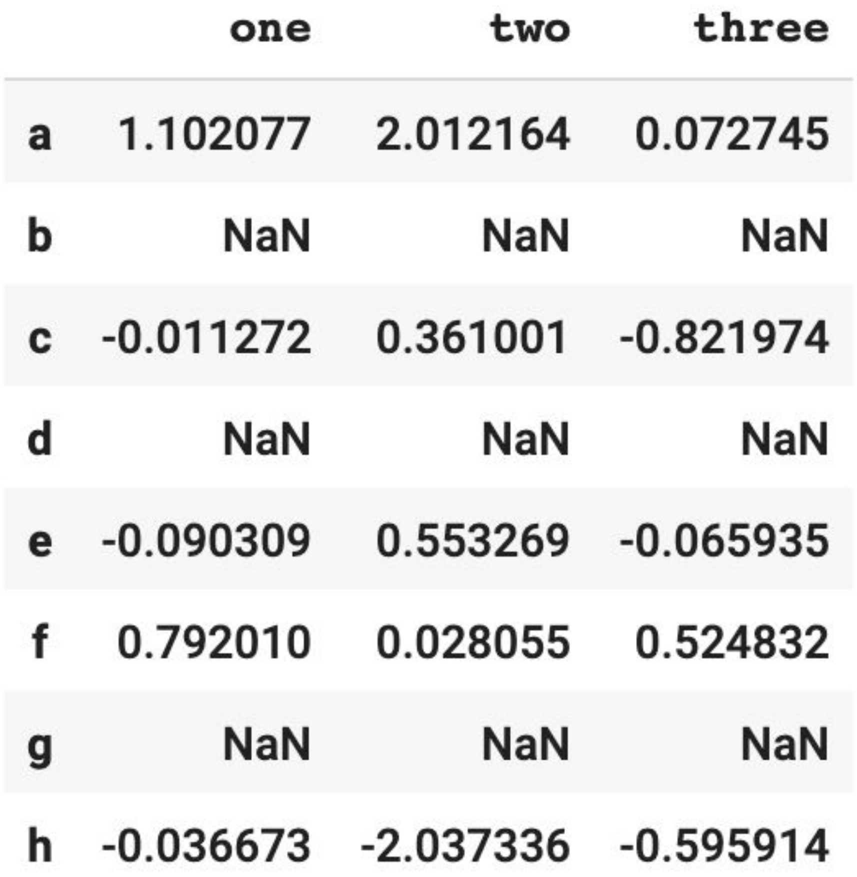
- Import the Singapore Health Expenditure dataset from

https://raw.githubusercontent.com/tertiarycourses/datasets/master/government-health-expenditure.csv

- Retrieve the all the data where operating expenditure data >5000
- Retrieve the all the data where operating expenditure data is between 5000 and 8000

Missing Values

- Missing values are represented with NaN in Pandas, for example,



	one	two	three
a	1.102077	2.012164	0.072745
b	NaN	NaN	NaN
c	-0.011272	0.361001	-0.821974
d	NaN	NaN	NaN
e	-0.090309	0.553269	-0.065935
f	0.792010	0.028055	0.524832
g	NaN	NaN	NaN
h	-0.036673	-2.037336	-0.595914

Remove Missing Data

- Use isnull() to check any missing data
  df['one'].isnull()
- Use dropna() to remove the missing
  df.dropna()

Impute Missing Data

- You can impute missing data with a fixed number, or forward fill or backfill

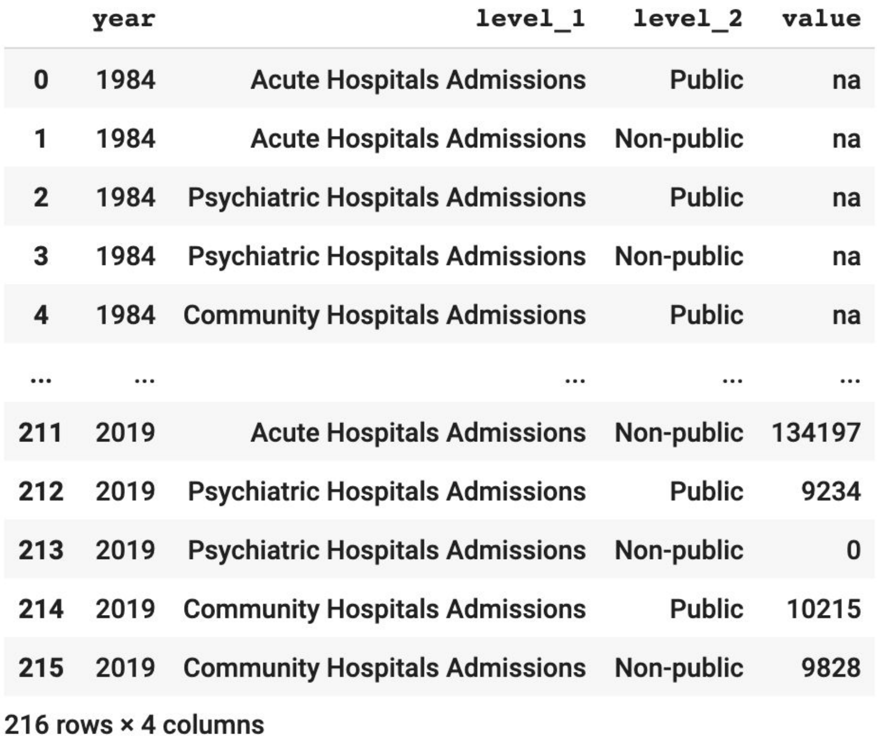
```
df.fillna(0)
df.fillna(method='pad')
df.fillna(method='backfill')
```

Activity: Filtering Data

- Import the Singapore Hospital Admission dataset from

https://raw.githubusercontent.com/tertiarycourses/datasets/master/hospital-admissions-by-sector-annual.csv

- Remove the missing data with 'na'



	year	level_1	level_2	value
0	1984	Acute Hospitals Admissions	Public	na
1	1984	Acute Hospitals Admissions	Non-public	na
2	1984	Psychiatric Hospitals Admissions	Public	na
3	1984	Psychiatric Hospitals Admissions	Non-public	na
4	1984	Community Hospitals Admissions	Public	na
...	...	...	...	...
211	2019	Acute Hospitals Admissions	Non-public	134197
212	2019	Psychiatric Hospitals Admissions	Public	9234
213	2019	Psychiatric Hospitals Admissions	Non-public	0
214	2019	Community Hospitals Admissions	Public	10215
215	2019	Community Hospitals Admissions	Non-public	9828

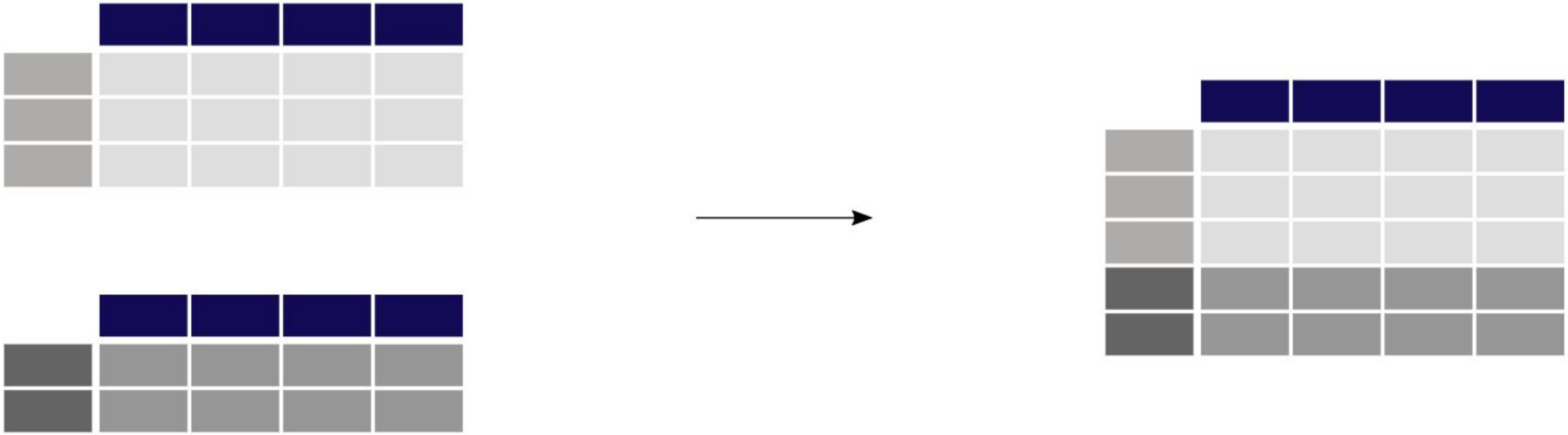
216 rows × 4 columns

Topic 2
Data Transformation

Concatenating Data

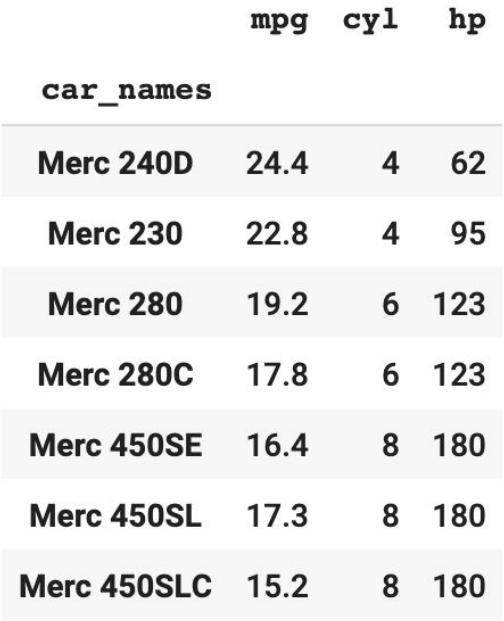
- The concat() function performs concatenation operations of multiple tables along one of the axis (0:row-wise or 1:column-wise).
- By default concatenation is along axis 0, so the resulting table combines the rows of the input tables.
- The syntax is

```
pd.concat(objs, axis=0, join='outer', ignore_index=False, keys=None,
        levels=None, names=None, verify_integrity=False, copy=True)
```



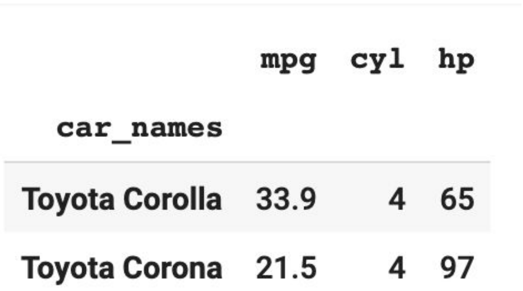
Concatenating Data Demo

```
merc = [c for c in mtcars_sample.index if 'Merc' in c]
merc_cars = mtcars_sample.loc[merc]
toyota = [c for c in mtcars_sample.index if 'Toyota' in c]
toyota_cars = mtcars_sample.loc[toyota]
merc_toyota_cars = pd.concat([merc_cars,
toyota_cars], axis=0)
```



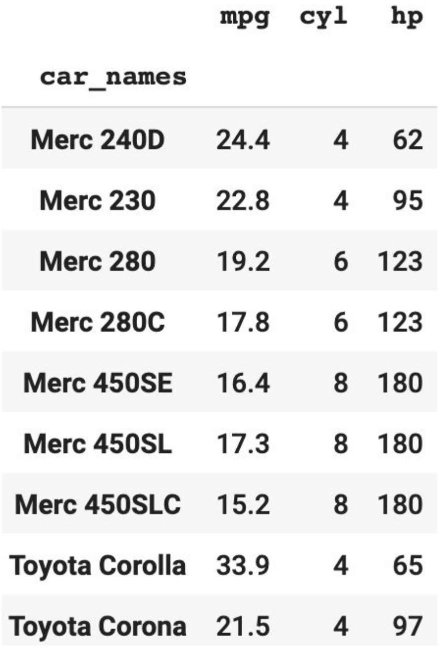
car_names	mpg	cyl	hp
Merc 240D	24.4	4	62
Merc 230	22.8	4	95
Merc 280	19.2	6	123
Merc 280C	17.8	6	123
Merc 450SE	16.4	8	180
Merc 450SL	17.3	8	180
Merc 450SLC	15.2	8	180

+



car_names	mpg	cyl	hp
Toyota Corolla	33.9	4	65
Toyota Corona	21.5	4	97

=



car_names	mpg	cyl	hp
Merc 240D	24.4	4	62
Merc 230	22.8	4	95
Merc 280	19.2	6	123
Merc 280C	17.8	6	123
Merc 450SE	16.4	8	180
Merc 450SL	17.3	8	180
Merc 450SLC	15.2	8	180
Toyota Corolla	33.9	4	65
Toyota Corona	21.5	4	97

Activity: Selecting and Slicing Data

- Import the air quality and pm2.5 data as follows:

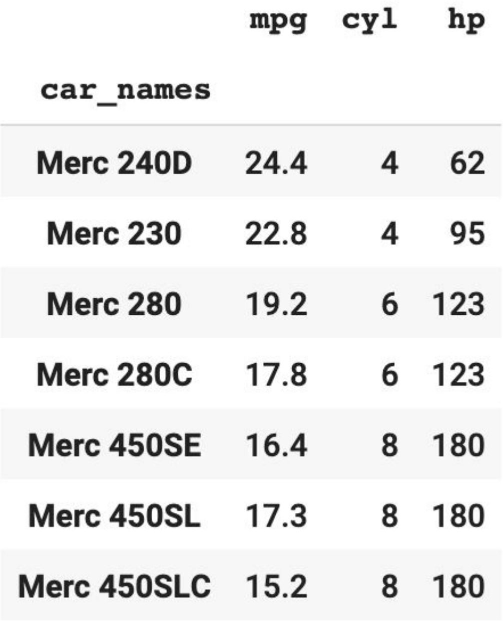
```
air_quality_no2 = pd.read_csv("https://raw.githubusercontent.com/pandas-dev/pandas/master/doc/data/air_quality_no2_long.csv")
air_quality_no2 = air_quality_no2[["date.utc", "location","parameter", "value"]]

air_quality_pm25 = pd.read_csv("https://raw.githubusercontent.com/pandas-dev/pandas/master/doc/data/air_quality_pm25_long.csv")
air_quality_pm25 = air_quality_pm25[["date.utc", "location","parameter", "value"]]
```

- Join the two datasets row wise.

Appending Data

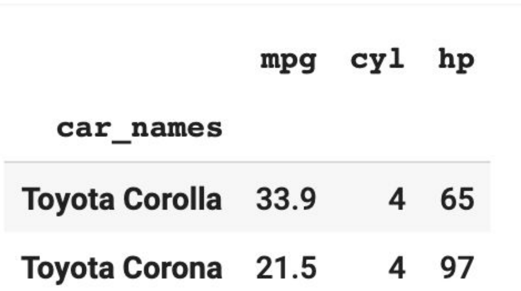
Alternate way to join the data is to use the append function. The syntax is as follows:

```
toyota_merc_cars2= toyota_cars.append(merc_cars)
toyota_merc_cars2
```



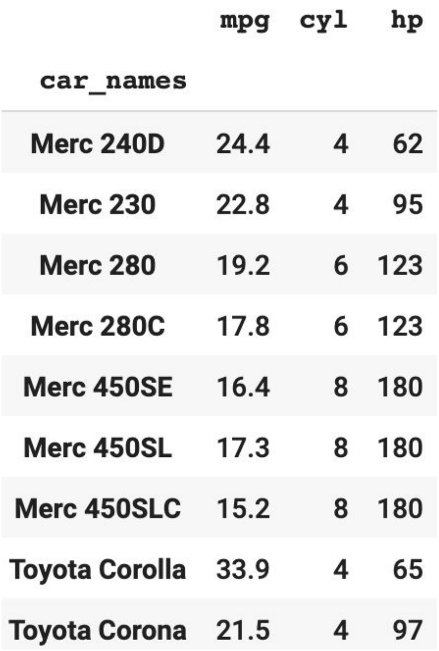
	mpg	cyl	hp
car_names			
Merc 240D	24.4	4	62
Merc 230	22.8	4	95
Merc 280	19.2	6	123
Merc 280C	17.8	6	123
Merc 450SE	16.4	8	180
Merc 450SL	17.3	8	180
Merc 450SLC	15.2	8	180

+



	mpg	cyl	hp
car_names			
Toyota Corolla	33.9	4	65
Toyota Corona	21.5	4	97

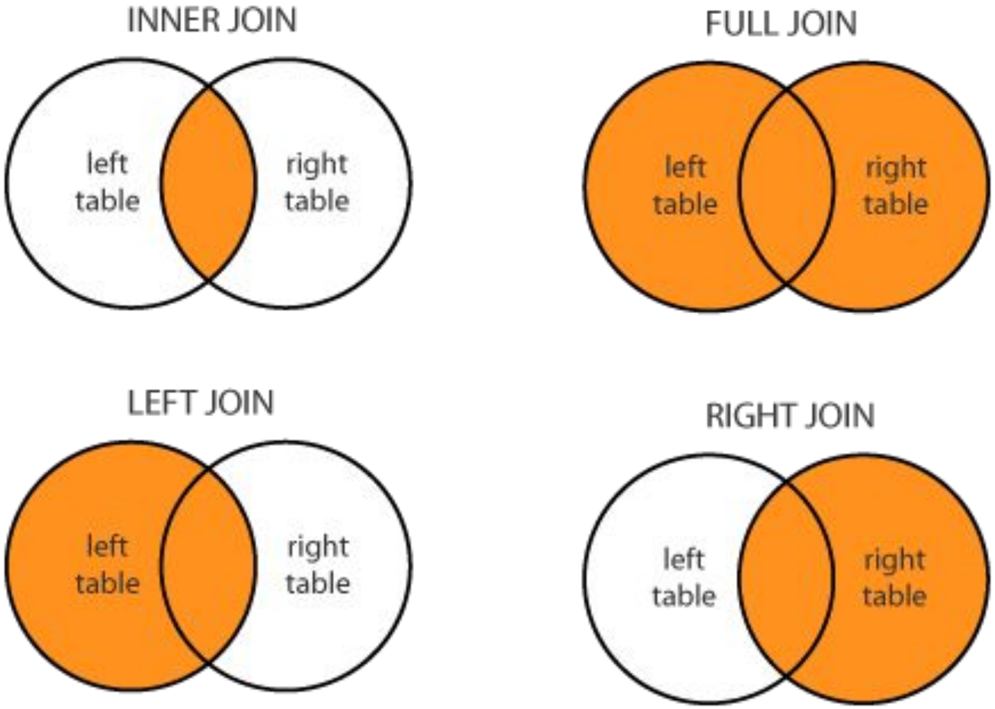
=



	mpg	cyl	hp
car_names			
Merc 240D	24.4	4	62
Merc 230	22.8	4	95
Merc 280	19.2	6	123
Merc 280C	17.8	6	123
Merc 450SE	16.4	8	180
Merc 450SL	17.3	8	180
Merc 450SLC	15.2	8	180
Toyota Corolla	33.9	4	65
Toyota Corona	21.5	4	97

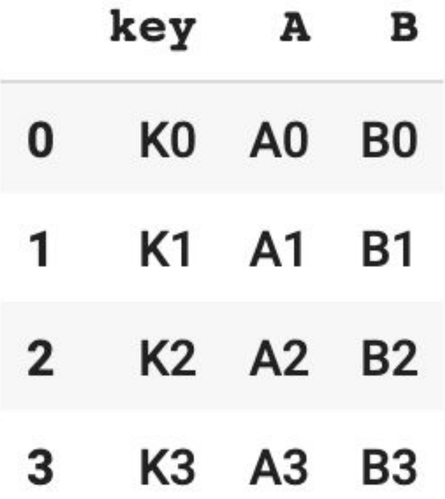
Merging Data

- Pandas has full-featured, high performance in-memory join operations idiomatically very similar to relational databases like SQL.
- The syntax is
  pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=True, suffixes=('_x', '_y'), copy=True, indicator=False, validate=None)



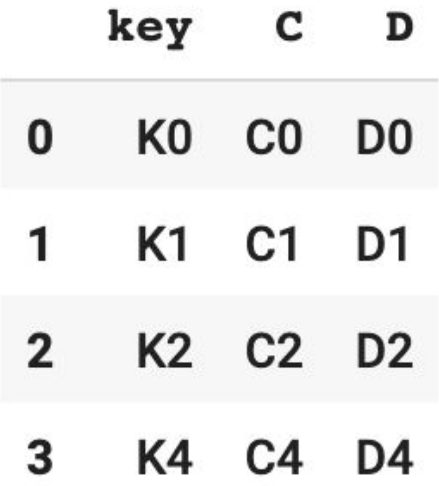
Merging Data Demo

```
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K4'],
                      'C': ['C0', 'C1', 'C2', 'C4'],
                      'D': ['D0', 'D1', 'D2', 'D4']})
result = pd.merge(left, right, on='key',how='inner')
```



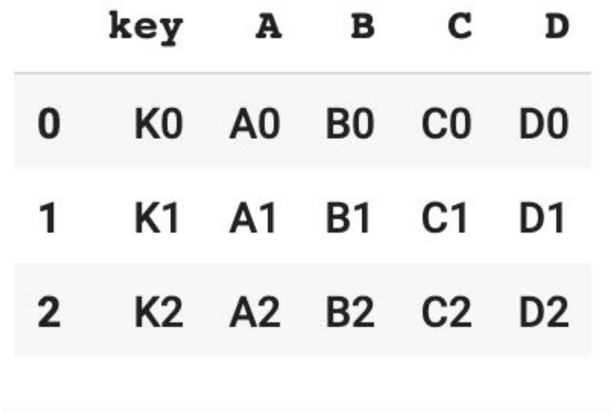
	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

+



	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K4	C4	D4

=



	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2

Activity: Merging Data

Merge air quality and pm 2.5 data using inner join

- based on location
- based on date

Sorting Data

- With sort_values(), the rows in the table are sorted according to the defined column(s). The index will follow the row order.
- Example mtcars_sample.sort_values(by="cyl",ascending=False)

Group and Aggregate Data

Index	Fruit
1	Apple
2	Apple
3	Orange
4	Apple
5	Orange

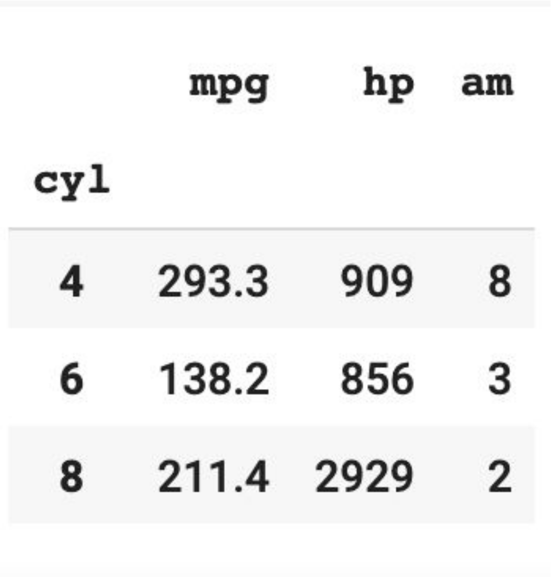
Fruit	
Apple	3
Orange	2

- To compare subsets
- To deduce reasons why subgroups differ
- To subset your data for your analysis

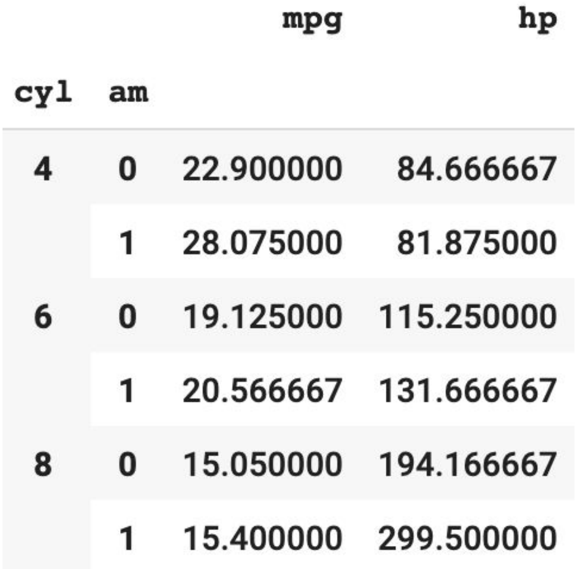
Groupby

- A groupby operation involves some combination of splitting the object, applying a function, and combining the results.

```
mtcars_sample.groupby(['cyl']).mean()
mtcars_sample.groupby('cyl').hp.mean()
mtcars_sample.groupby(['cyl']).sum()
mtcars_sample.groupby(['cyl']).agg(['mean', 'count'])
mtcars_sample.groupby(['cyl','am']).mean()
mtcars_sample.groupby('cyl').agg(lambda x:max(x)-min(x))
mtcars_sample.groupby(['cyl', 'am']).agg(['mean', 'count'])
```



	mpg	hp	am
cyl			
4	293.3	909	8
6	138.2	856	3
8	211.4	2929	2



		mpg	hp
cyl	**am**		
4	**0**	22.900000	84.666667
	1	28.075000	81.875000
6	**0**	19.125000	115.250000
	1	20.566667	131.666667
8	**0**	15.050000	194.166667
	1	15.400000	299.500000

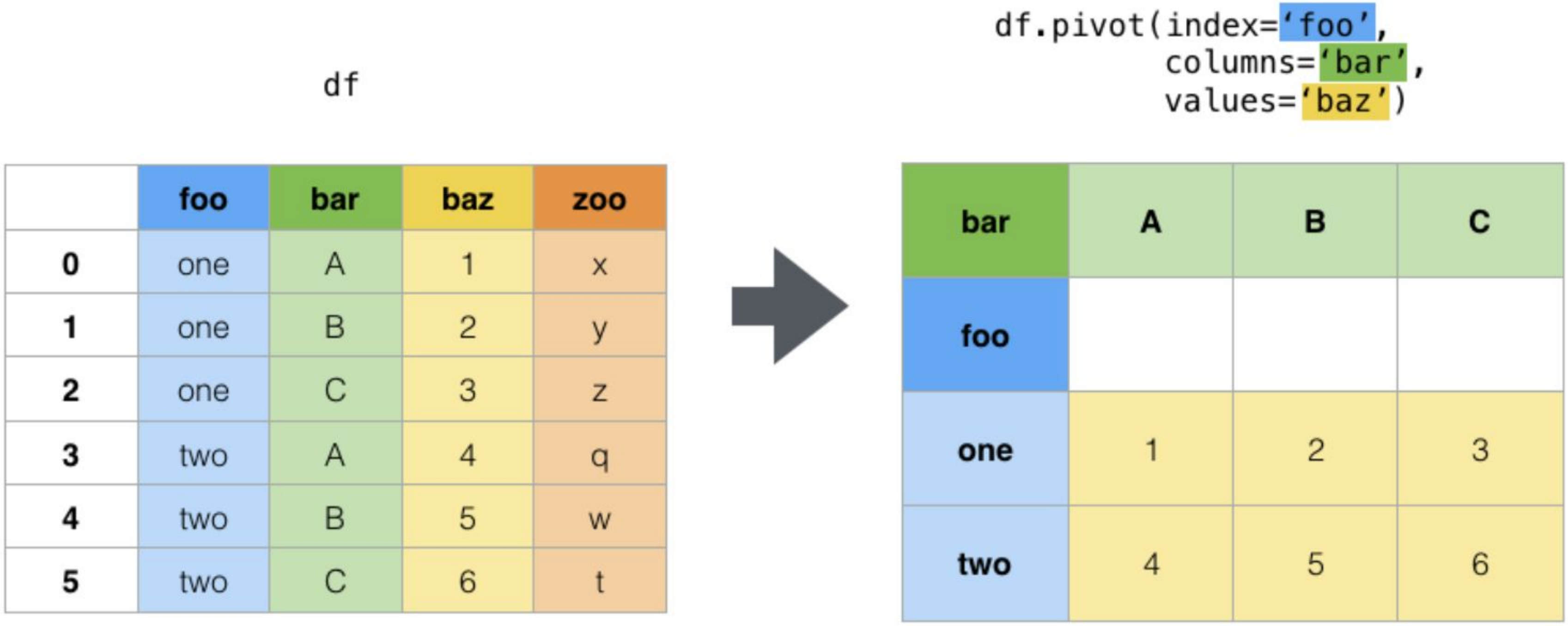
Activity: Groupby

- Import the Singapore long term care facilities data https://raw.githubusercontent.com/tertiarycourses/datasets/master/number-of-residential-long-term-care-facilities-sector-breakdown.csv
- Compute the total number of long term care facilities breakdown by year and sector using Group By method

Time: 10 mins

Pivot Table

- Pivot table is a well known concept in spreadsheet to reshape the data.
- pivot() can be used to rearrange the data
- pivot_table() can be used, providing an aggregation function (e.g. mean) on how to combine these values.



df

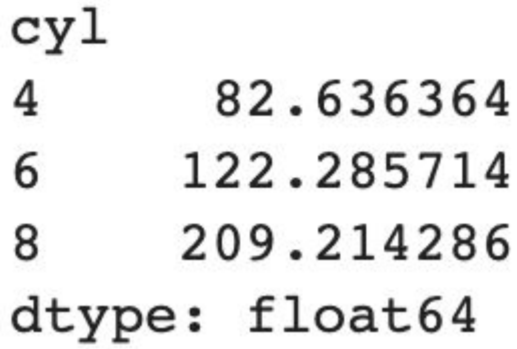
	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

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

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

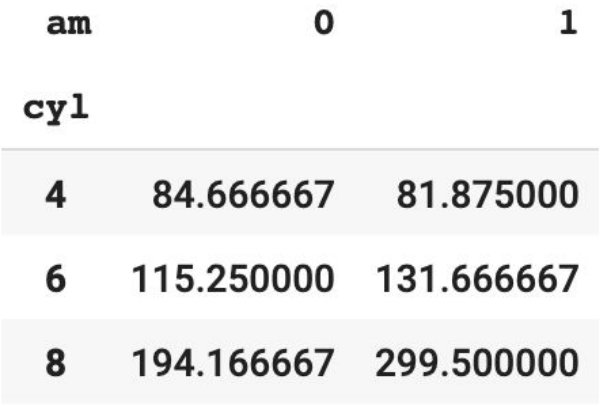
Pivot Table

```
mtcars_sample.pivot(columns='cyl',values='hp')
mtcars_sample.pivot(columns='cyl',values='hp').mean()
```



```
cyl
4       82.636364
6      122.285714
8      209.214286
dtype: float64
```

```
mtcars_sample.pivot_table(index='cyl',columns='am',
values='hp',aggfunc='mean')
```



am	0	1
cyl		
4	84.666667	81.875000
6	115.250000	131.666667
8	194.166667	299.500000

Activity: Pivot Table

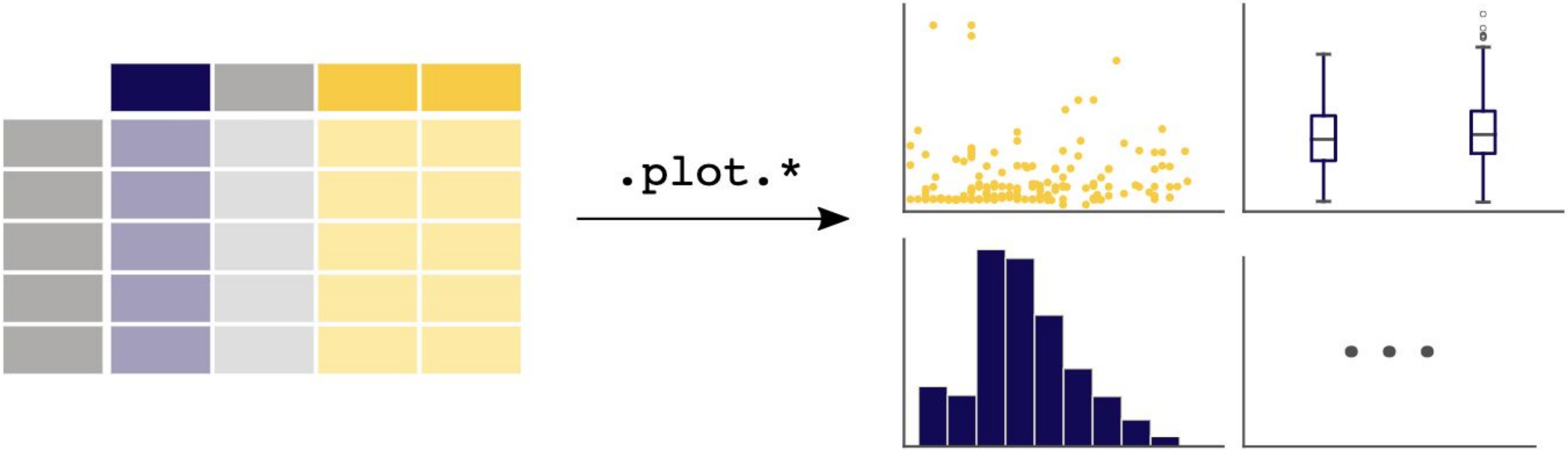
- Import the Singapore long term care facilities data https://raw.githubusercontent.com/tertiarycourses/datasets/master/number-of-residential-long-term-care-facilities-sector-breakdown.csv
- Compute the total number of long term care facilities breakdown by year and sector using Pivot Table method

Time: 10 mins

Topic 3
Data Visualization

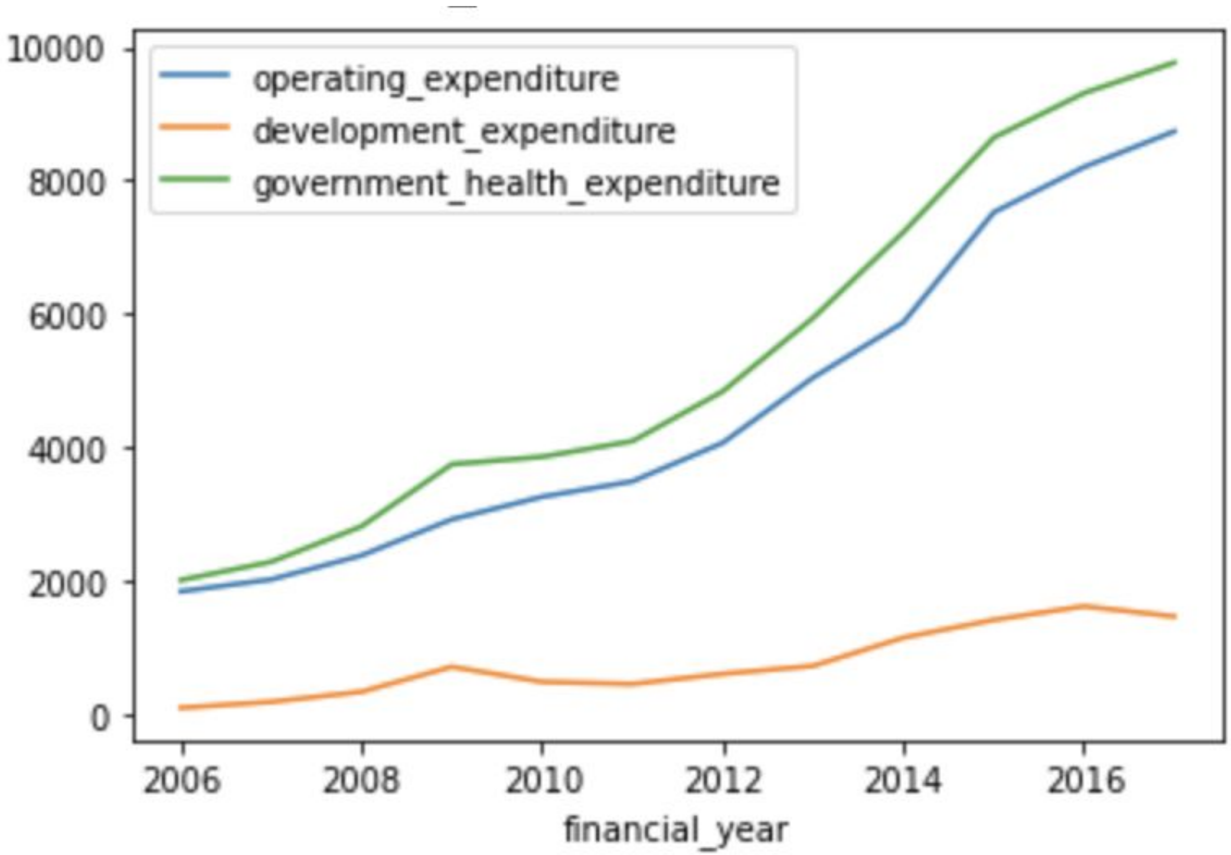
Create Plots in Pandas

- You can quickly plot the dataframe data using the plot method
- By default, plot will yield a line plot
- Example:
  mtcars_sample.plot()



Activity: Line Plot

Plot the Singapore healthcare expenditure over the year



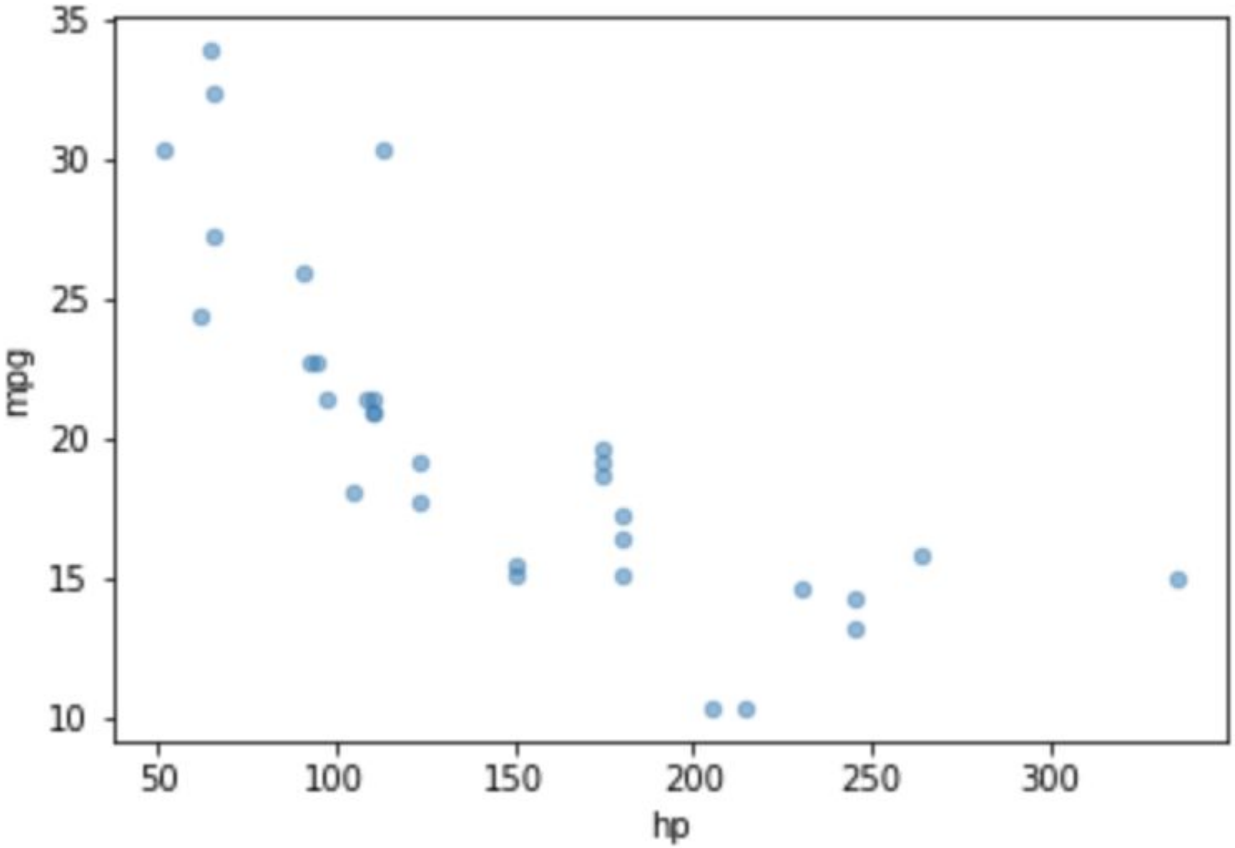
Pandas Plots

Pandas support the following data visualizations:

- Area plot: area
- Bar plot: bar
- Horizontal Bar plot: barh
- Boxplot: box
- Density plot: density
- Histogram: hist
- Line plot: line
- Pie plot: pie
- Scatter plot: scatter

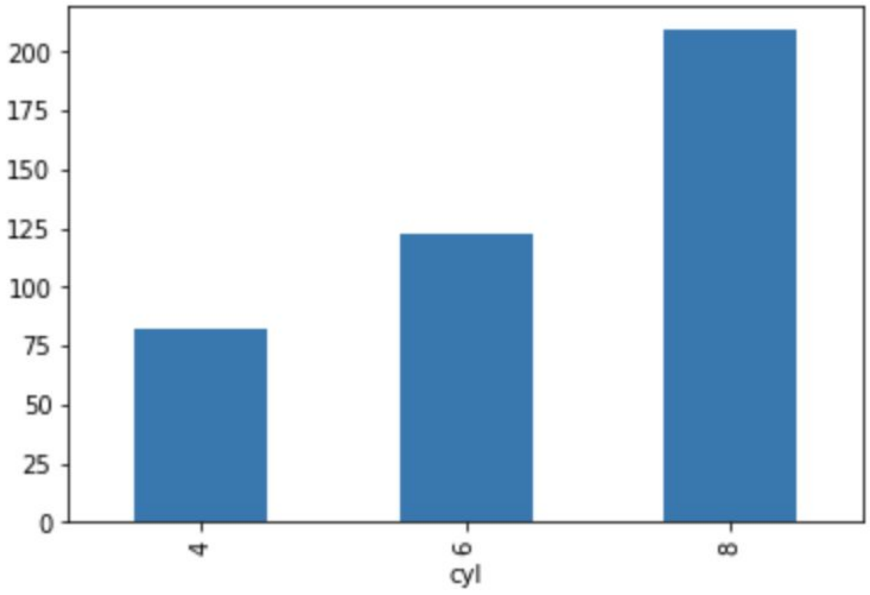
Scatter Plot

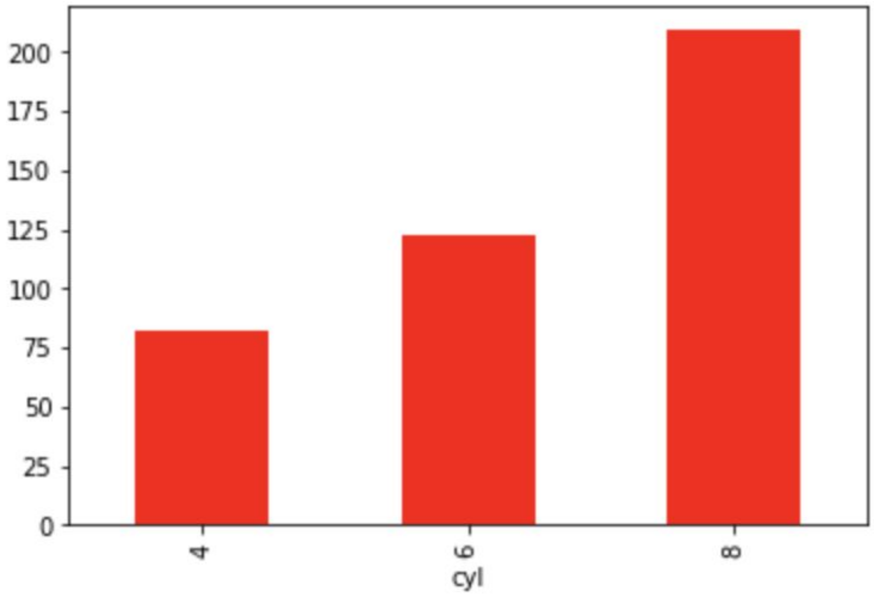
```
mtcars_sample.plot.scatter(x="hp", y="mpg",alpha=0.5)
```



Bar Plot

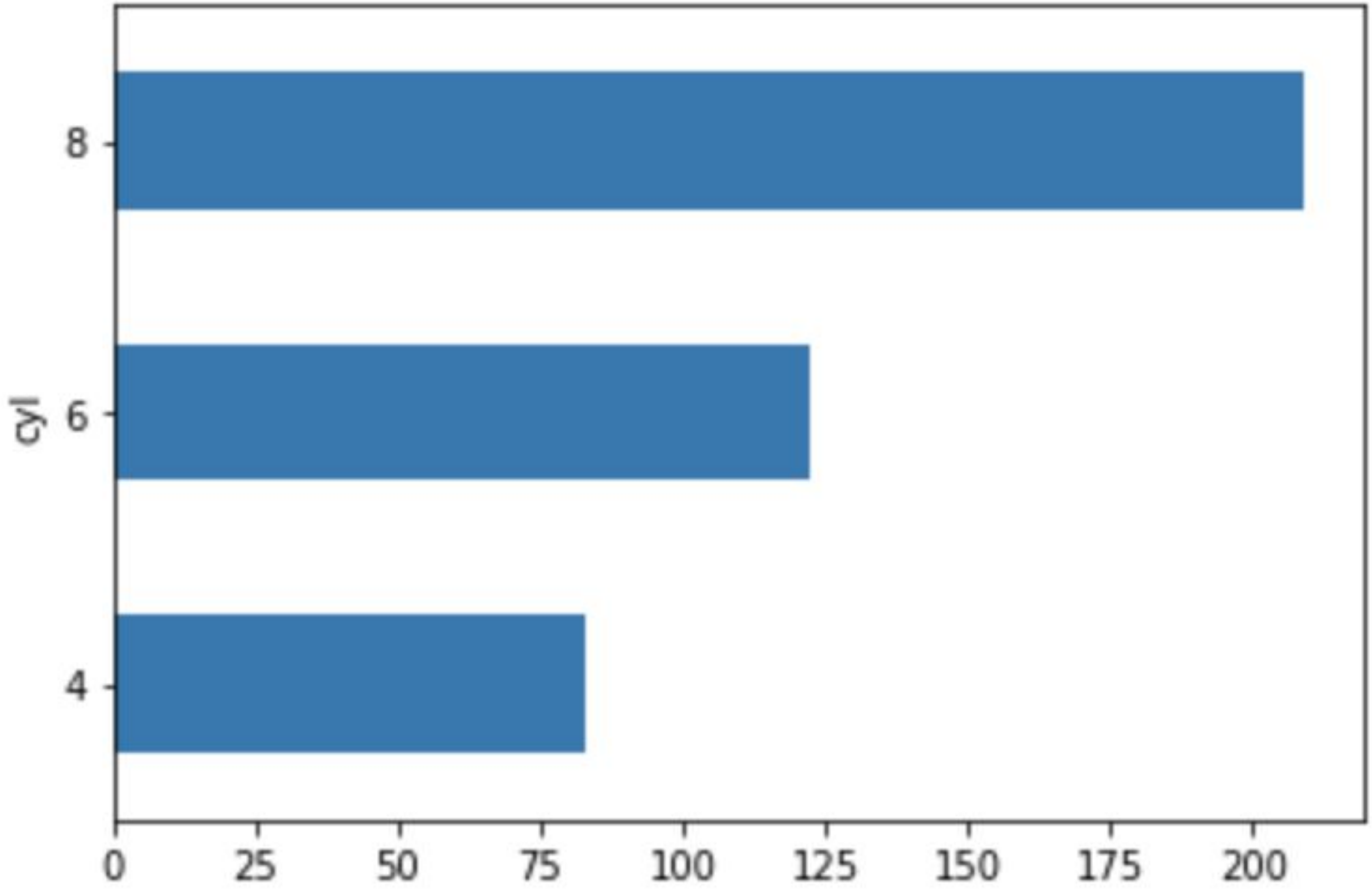
```
mtcars_sample.pivot(columns='cyl',values='hp').mean().plot.bar()

mtcars_sample.pivot(columns='cyl',values='hp').mean().plot.bar(color='red')

mtcars_cyl=
mtcars_sample.pivot(columns='cyl',values='hp').mean().plot(kind='bar',colo
r='red')
```





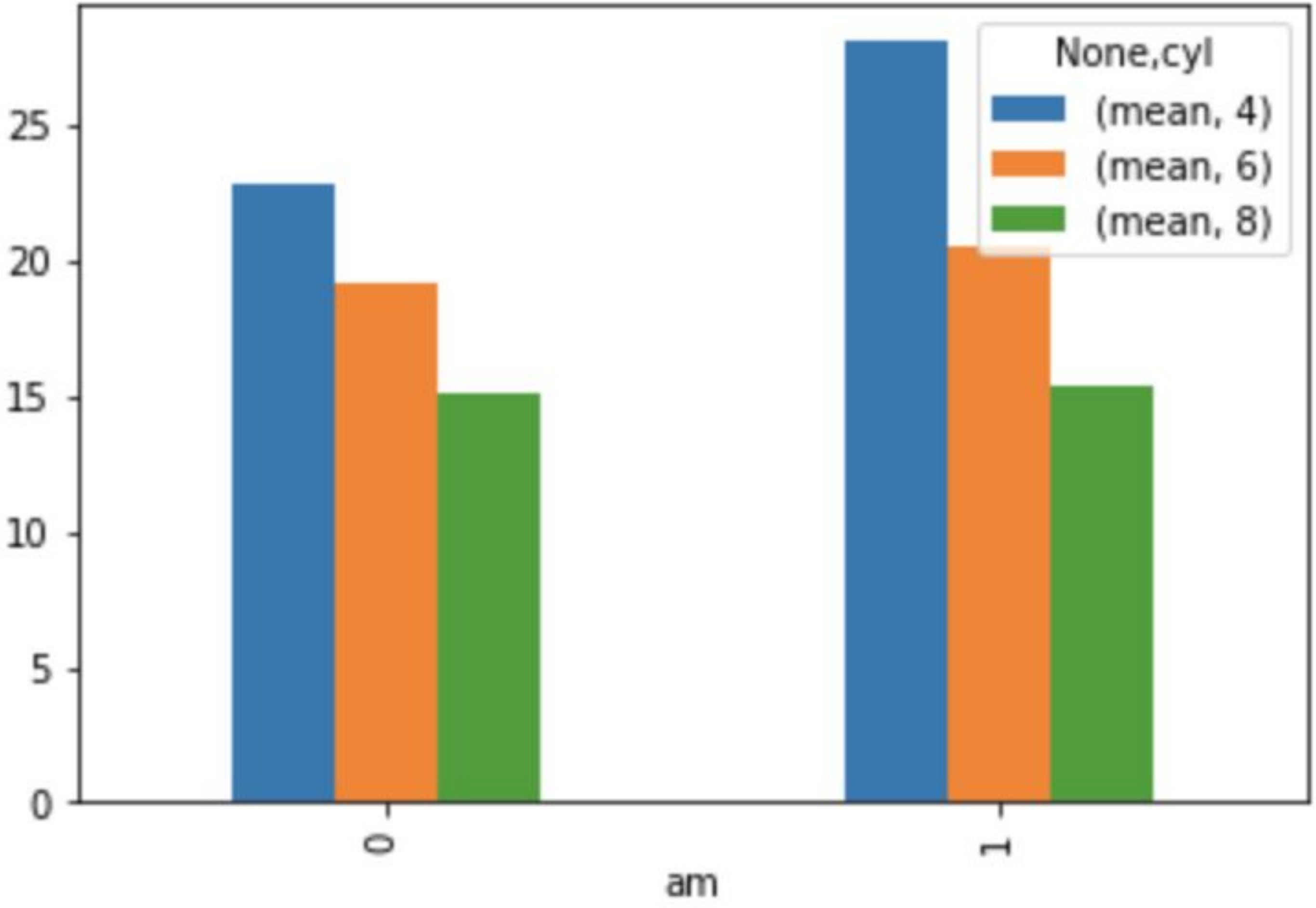
Horizontal Bar Plot

```
mtcars_cyl=
mtcars_sample.pivot(columns='cyl',values='hp').mean().plot.barh()
```



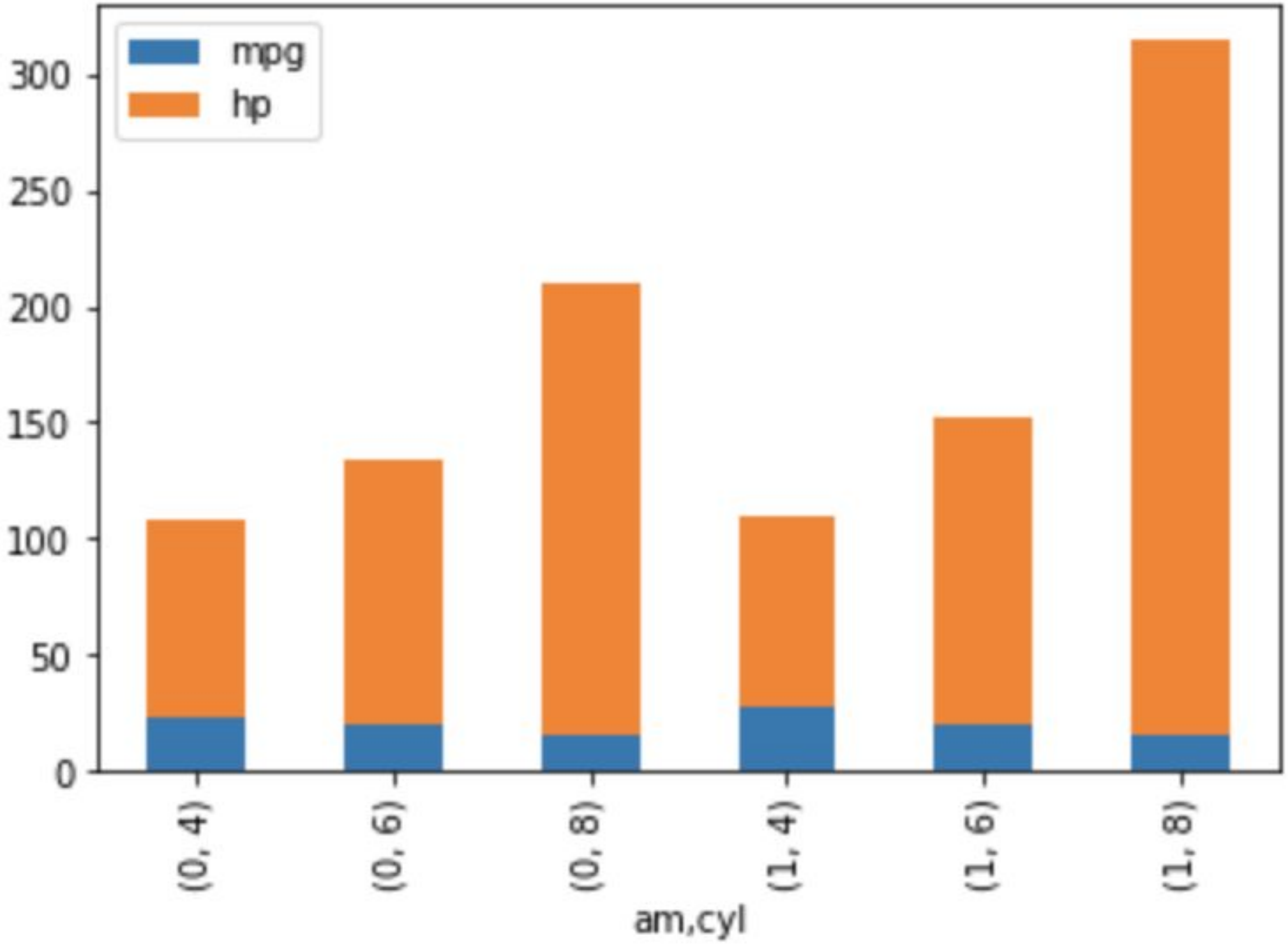
Stacked Bar Plot

```
mtcars_sample.pivot_table(index='am',columns='cyl',values='mpg',aggfu
nc=['mean']).plot.bar()
```



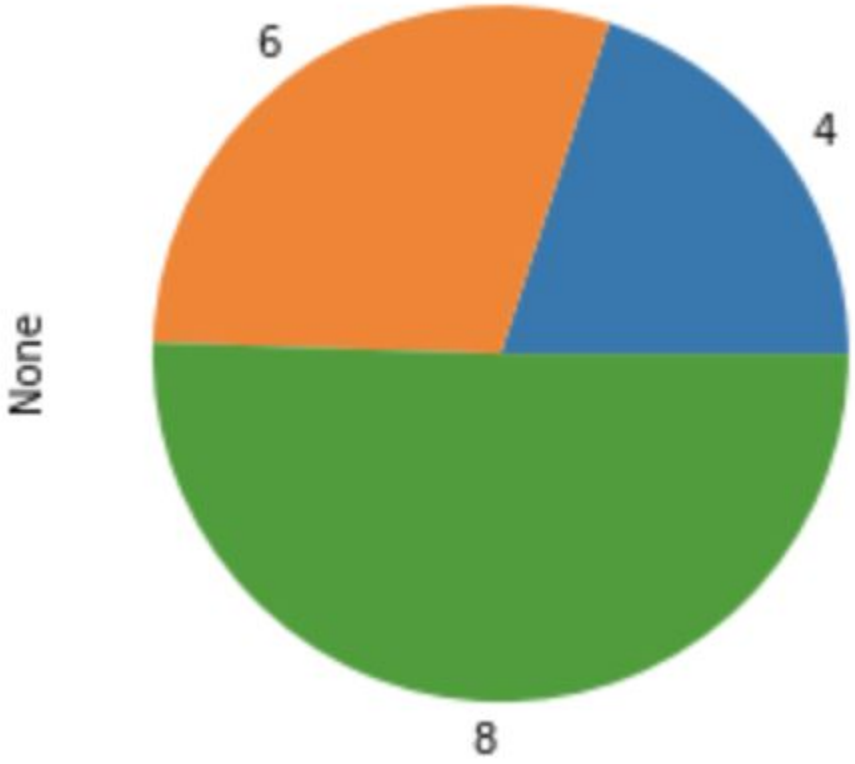
Stacked Bar Plot

```
mtcars_sample.groupby(['am','cyl']).mean().plot.bar(stacked=True)
```



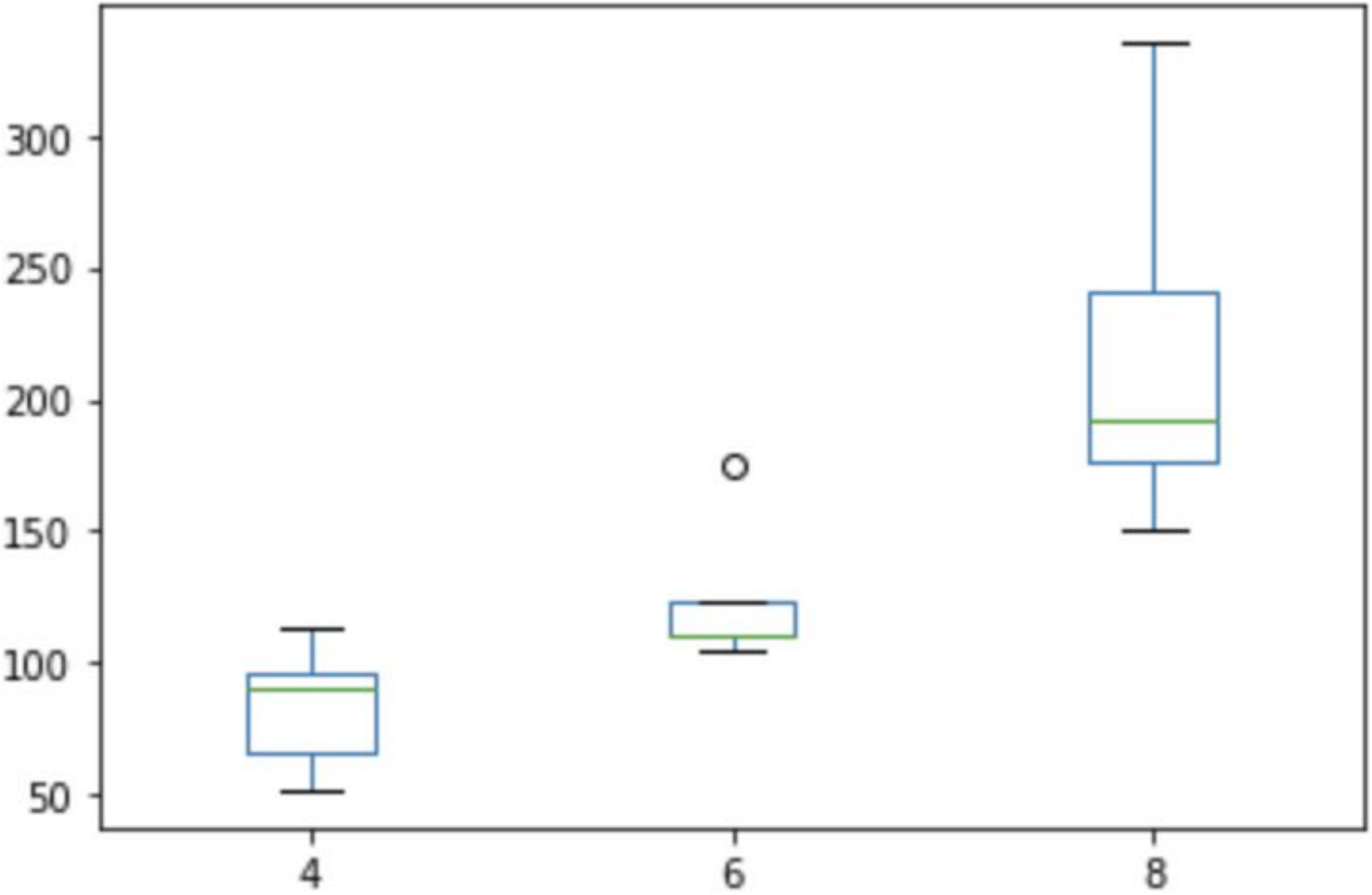
Pie Plot

```
mtcars_cyl=
mtcars_sample.pivot(columns='cyl',values='hp').mean().plot.pie()
```



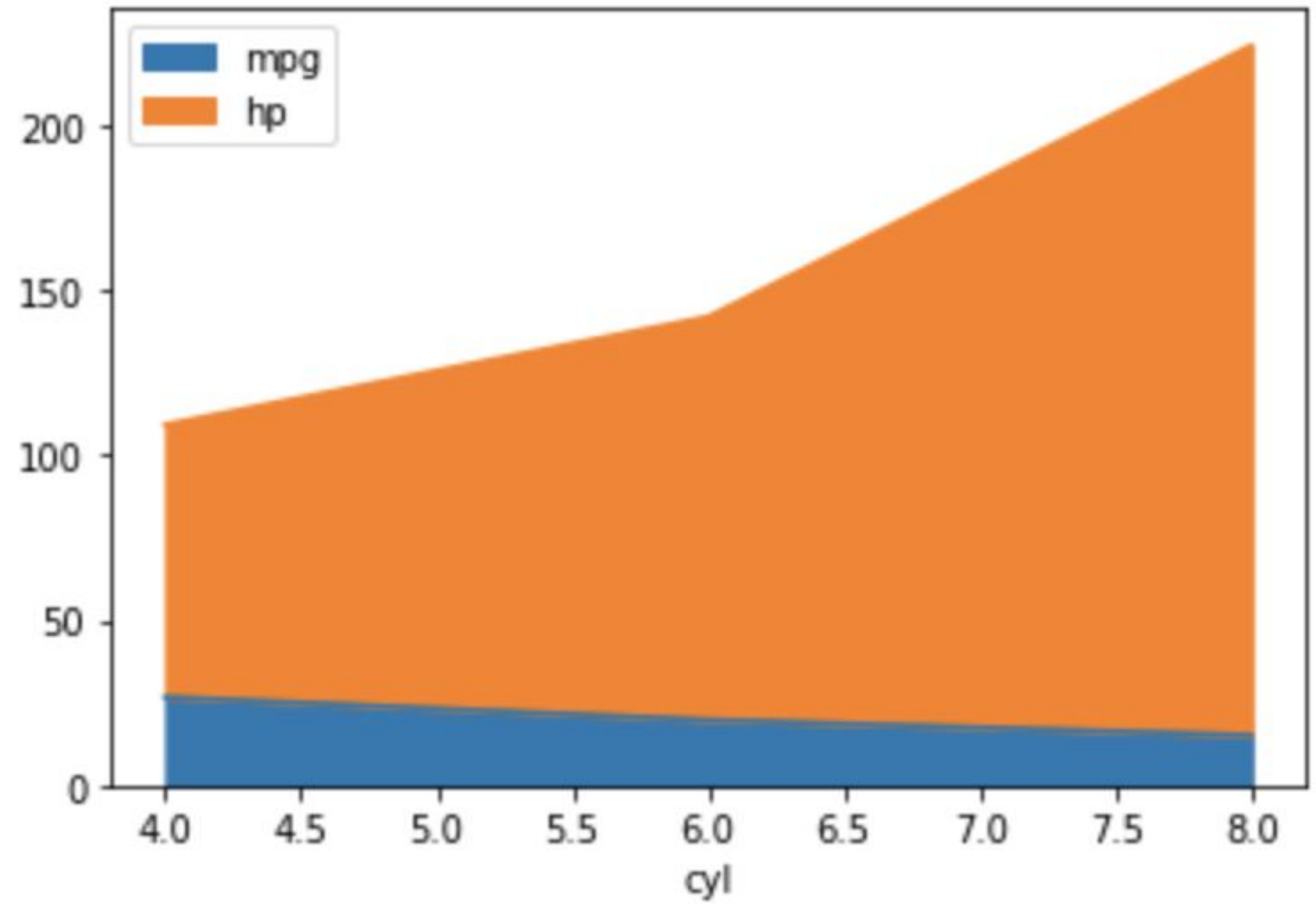
Box Plot

```
mtcars_cyl= mtcars_sample.pivot(columns='cyl',values='hp').plot.box()
```



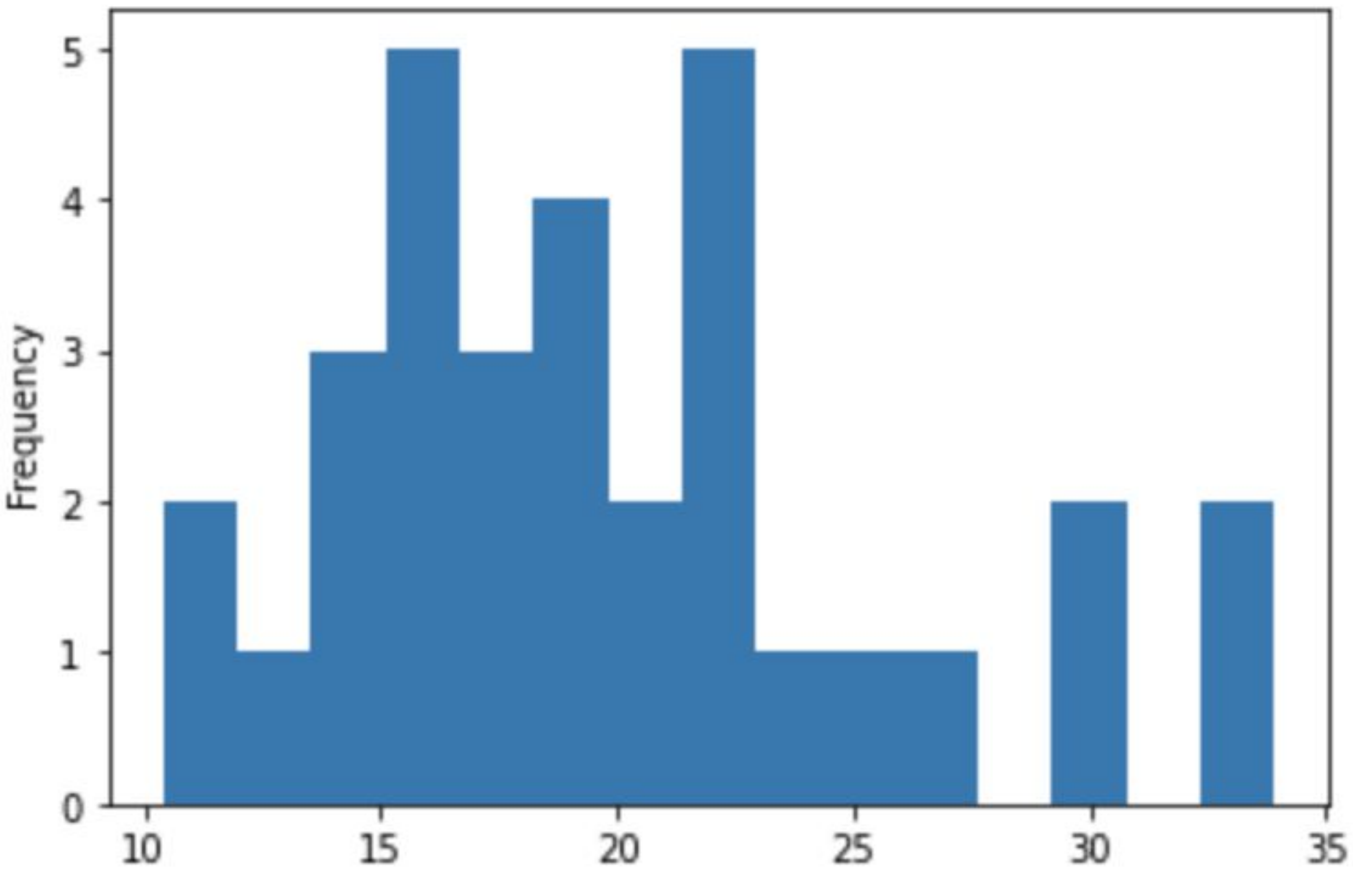
Area Plot

```
mtcars_sample2 = mtcars_sample[['cyl','mpg','hp']]
mtcars_sample2.groupby(['cyl']).mean().plot.area()
```



Histogram

```
mtcars_sample.mpg.plot.hist(bins=15)
```



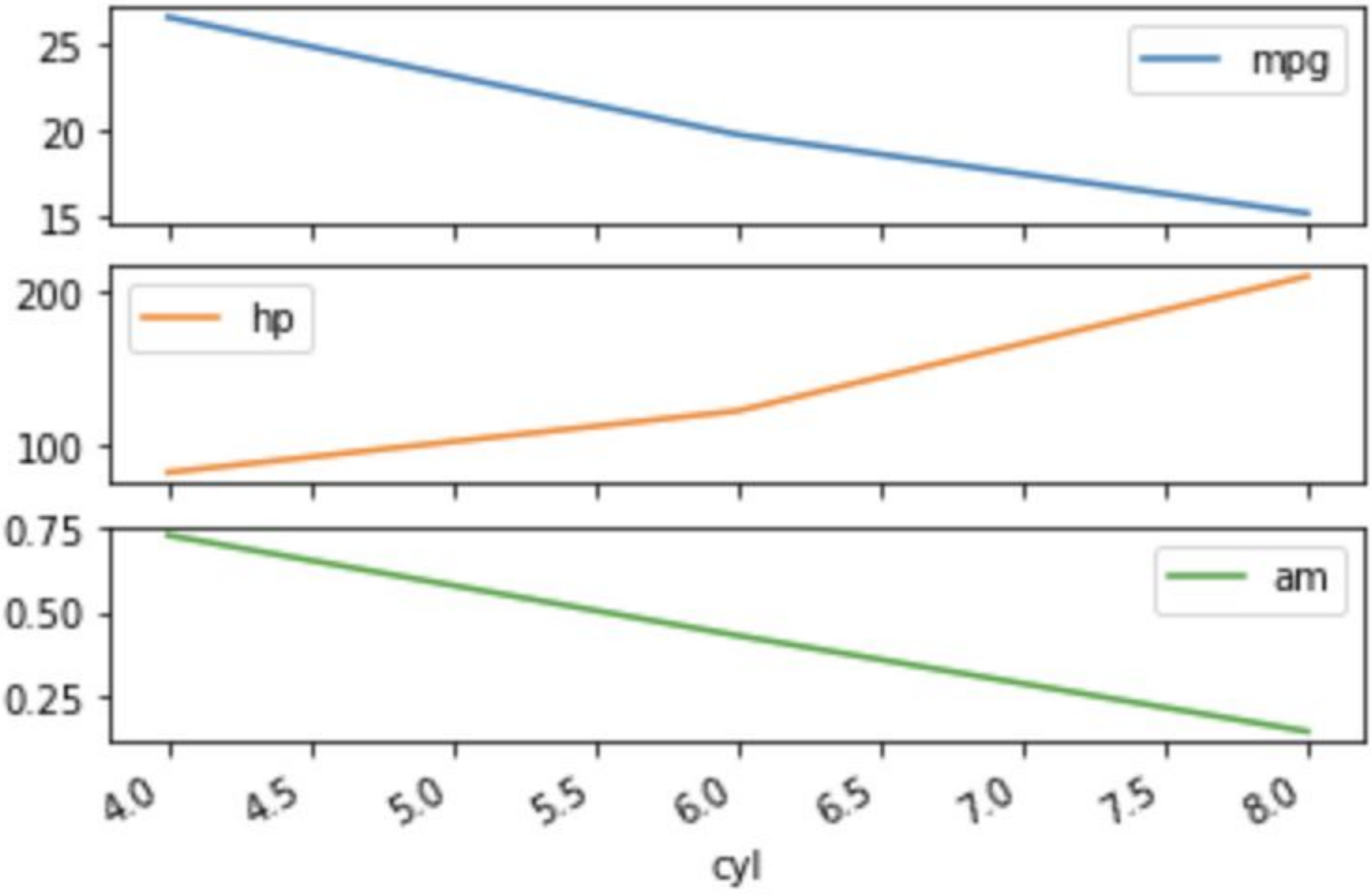
Activity: Data Visualization

- Import the Singapore long term care facilities data https://raw.githubusercontent.com/tertiarycourses/datasets/master/number-of-residential-long-term-care-facilities-sector-breakdown.csv
- Create a horizontal bar plot of the total long term care facilities by the sector

Time: 10 mins

Subplot

```
mtcars_cyl= mtcars_sample.groupby('cyl').mean()
mtcars_cyl.plot(subplots=True)
```

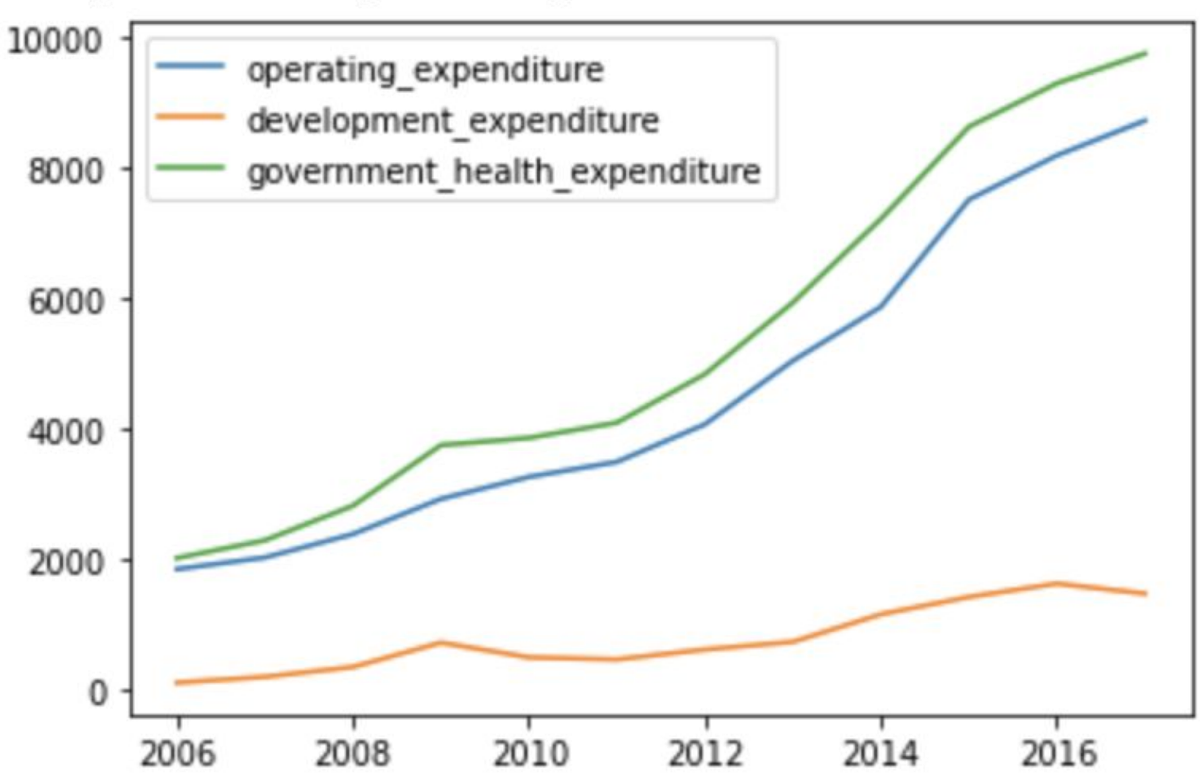


Activity: Data Visualization

- Import the Singapore Health Expenditure dataset from

https://raw.githubusercontent.com/tertiarycourses/datasets/master/government-health-expenditure.csv

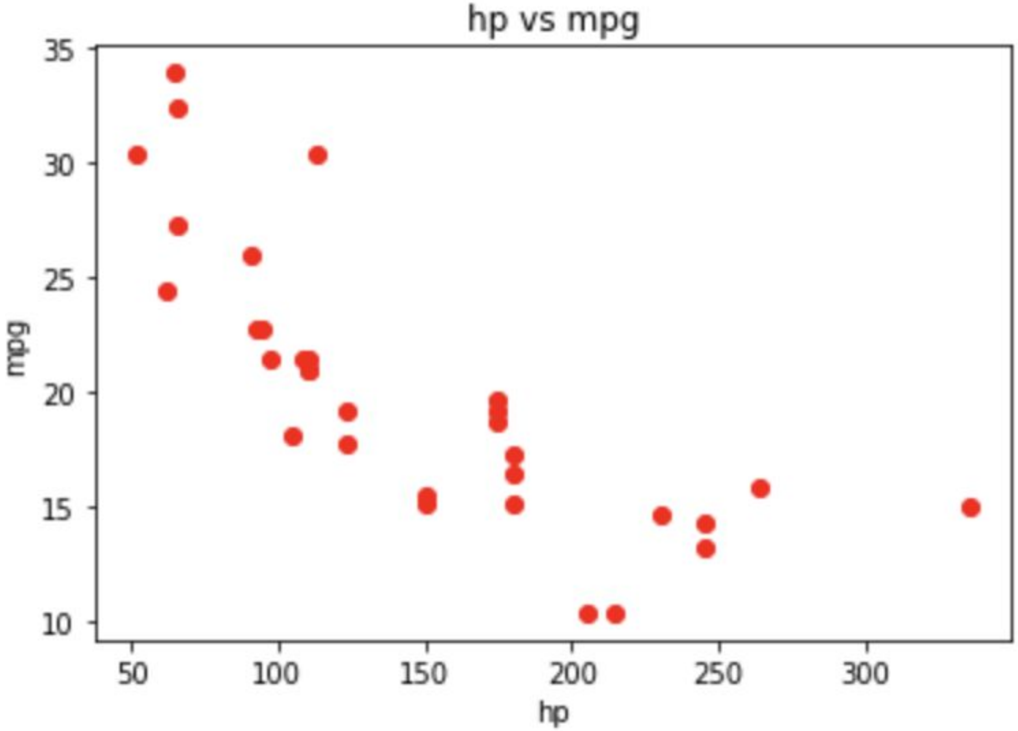
- Create 3 subplots for each expenditure.



Matplotlib

- Panda Series and DataFrame data structures work seamlessly with more advanced data visualization tools such as Matplotlib or Seaborn.
- For example, you can do a scatter plot using Matplotlib as follows

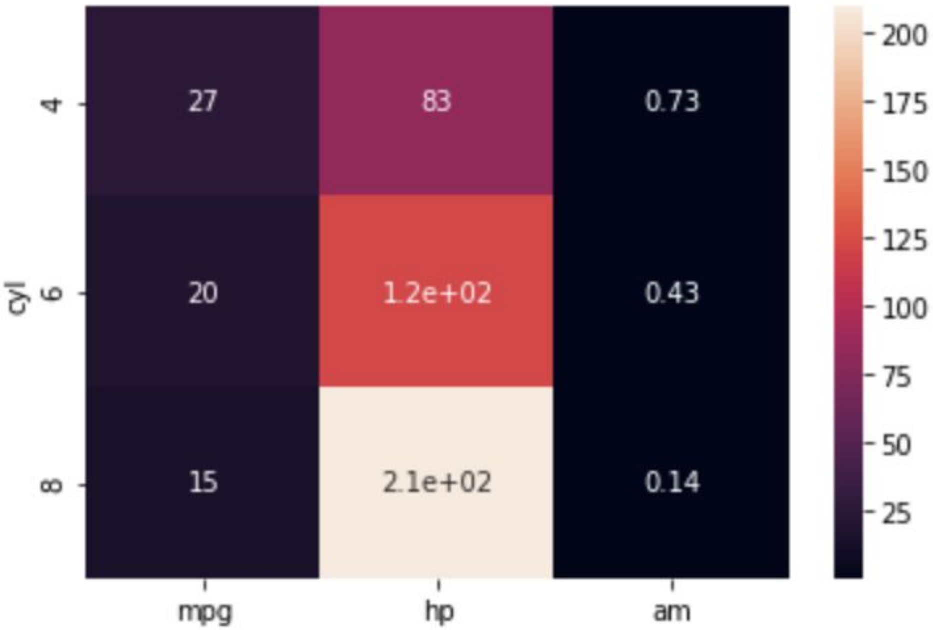
```
plt.scatter(x="hp", y="mpg", data = mtcars_sample,color='red')
plt.xlabel('hp')
plt.ylabel('mpg')
plt.title('hp vs mpg')
```

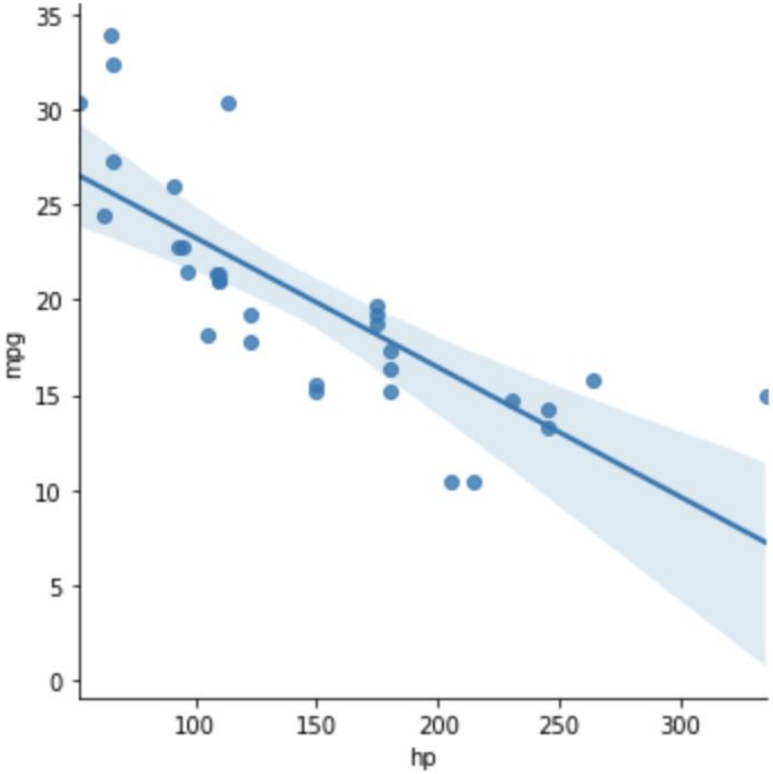


Seaborn

- You can create heatline and regression plots readily with Seaborn.

```
mtcars_cyl= mtcars_sample.groupby('cyl').mean()
sb.heatmap(mtcars_cyl, annot=True)
sb.lmplot(x="hp", y="mpg",data=mtcars_sample,fit_reg=True)
```





Topic 4
Data Analysis

What is Statistics?

Statistics is a discipline which is concerned with:

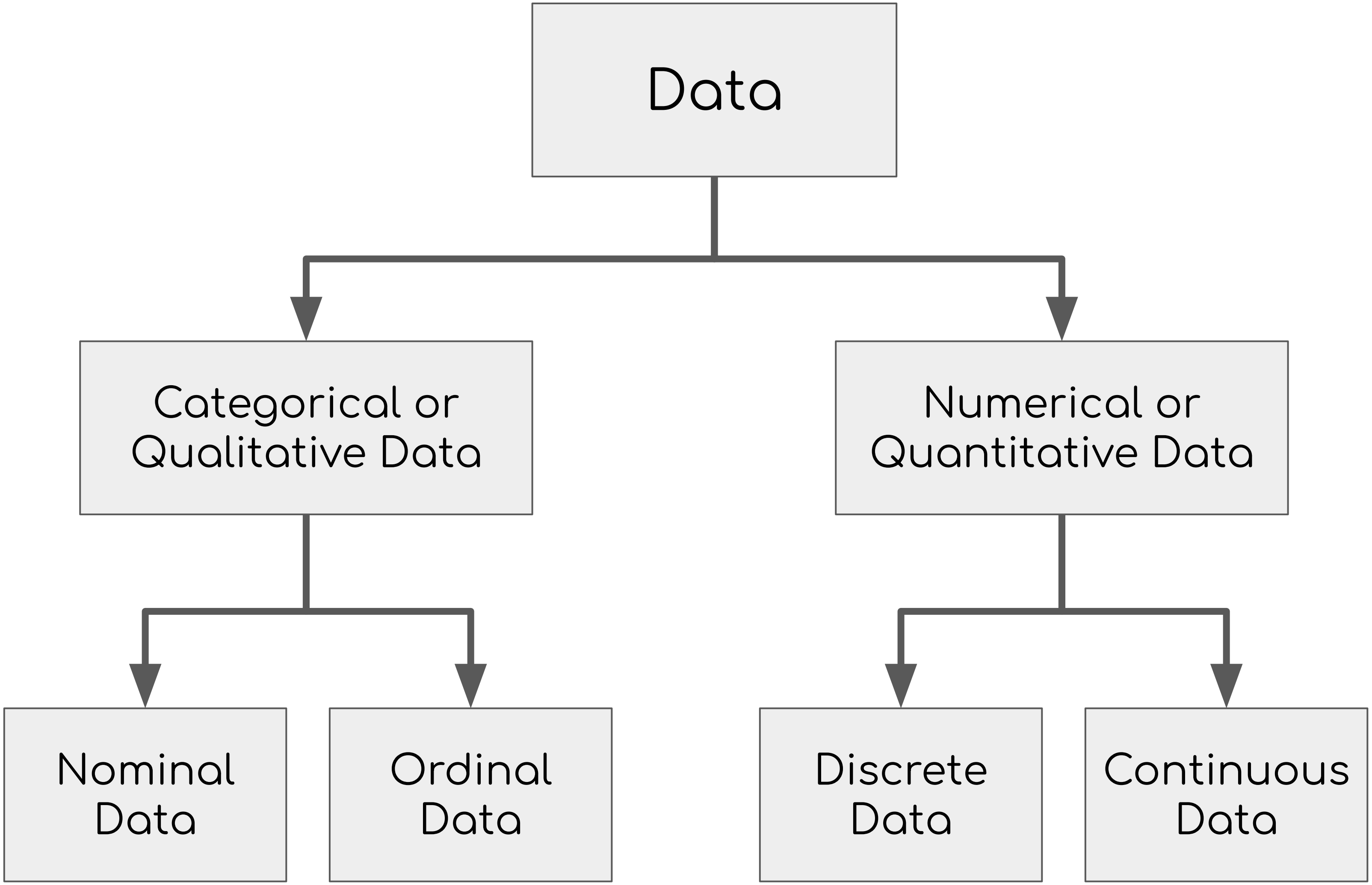
- designing experiments and other data collection,
- summarizing information to aid understanding,
- drawing conclusions from data, and
- estimating the present or predicting the future.

Statistical statements:
"I sleep for about eight hours per night on average"
"You are more likely to pass the exam if you start preparing earlier"

Why Statistics Matter?

- Environmental Study
  - Is Singapore getting hotter over last 10 years?
- Policy Study
  - Is more people using green transport such as Bicycles, Buses, Carpool, CNG Cars, Electric Cars, Electric Scooters
- Market Analysis
  - Is more people likely to take green transport if they've seen a recent TV advertisement for green transport?
- Public Transport
  - Is more people likely to commute by MRT if we have more MRT stations in the neighborhood?
- Health Care
  - Does air pollution from vehicles cause any health concern?
- Data Science
  - Statistics is fundamental for understanding Artificial Intelligence and Machine Learning.

Types of Data



Data

Categorical or Qualitative Data

Numerical or Quantitative Data

Nominal Data

Ordinal Data

Discrete Data

Continuous Data

Categorical and Quantitative Data

- **Categorical (Qualitative) Data** - each observation belongs to one of a set of categories. Examples:
  - Weather (Rainy /Sunny)
  - Air Pollutants (Ozone/Nitrogen Dioxide)
  - Gender (Male or Female)
  - Place of residence (HDB, Condo, ...)
  - Marital status (Married, Single,...)
- **Quantitative (Numerical) Data** - observations take numerical values. Examples:
  - Surface Air temperature
  - Weekly number of dengue cases
  - No. of days with rainfall in a month
  - Age
  - Number of cars
  - Weight

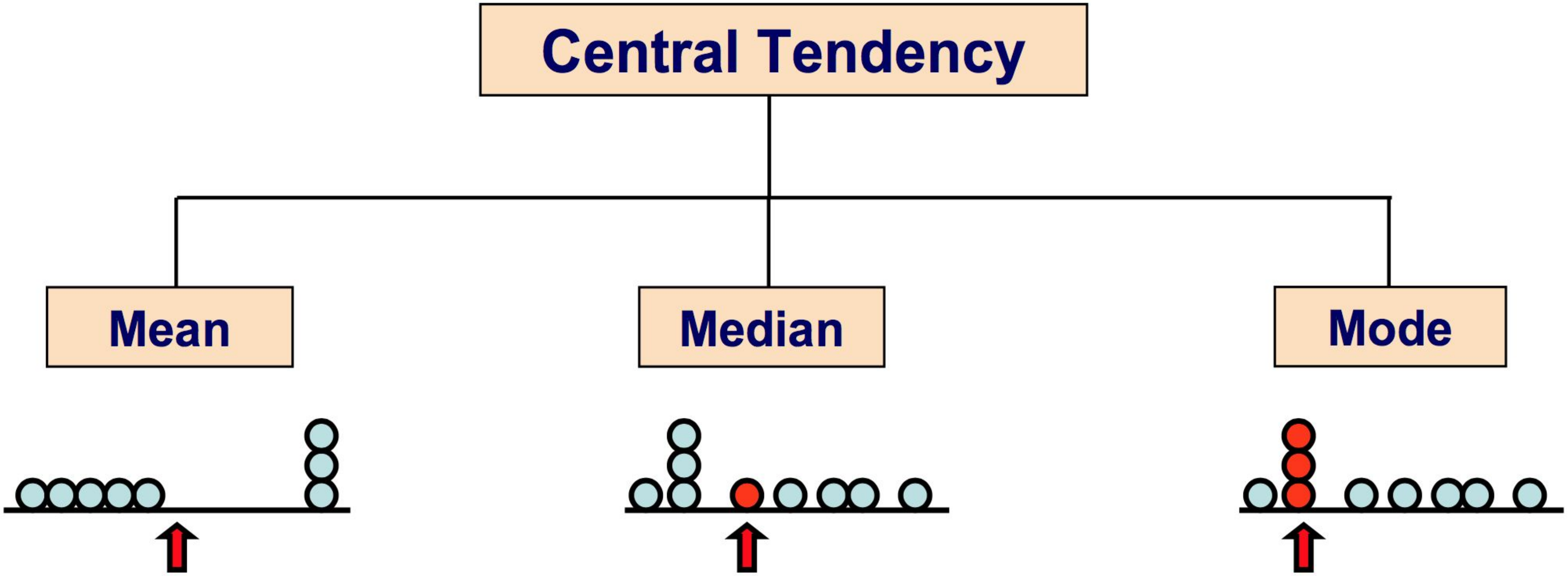
Nominal and Ordinal Data

- **Nominal Data** is defined as data that is used for naming or labelling variables, without any quantitative value. It is sometimes called “labels” data Eg
  - Male/Female
  - Red/Green/Blue
- **Ordinal Data** is a type of categorical data with an order. The variables in ordinal data are listed in an ordered manner.
  - Disagree/Neutral/Agree/Strongly Agree
  - Very Bad/Bad/Good/Very Good

Discrete and Continuous Data

- **Discrete Data is** a set of countable numbers such as 0, 1, 2, 3,……Examples:
  - No. of days with rainfall in a month
  - Weekly no. of dengue cases
  - Number of children in a family
  - Number of foreign languages spoken
- **Continuous Data** are continuous numbers from an interval. Examples:
  - Surface Air temperature
  - Amount of rainfall in a month
  - Height
  - Weight

Measures of Central Tendency



Central Tendency

Mean

Median

Mode

The center of gravity or the balance point

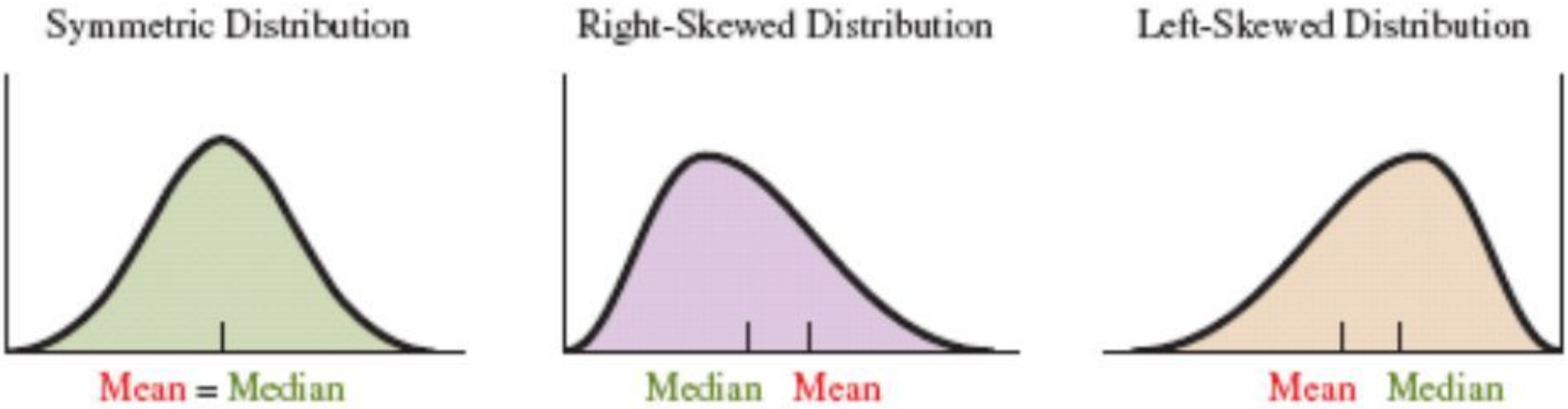
Midpoint of ranked values

Most frequently observed value

- Mean - add up all the values and divide by how many there are
- Median - Arrange all the numbers from smallest to largest:
  - odd number of points: Median = middle value
  - even number of points: Median= mean of the middle two values

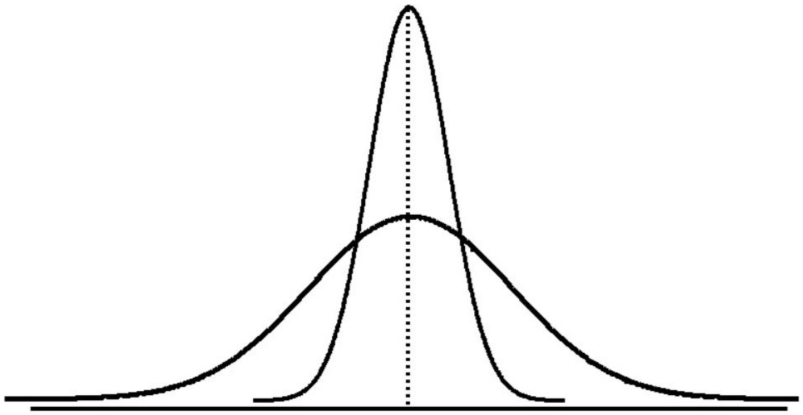
Mean vs Median

- Mean
  - Useful for roughly symmetric quantitative data
  - Sensitive to outlier data
- Median
  - Splits the data into halves
  - Useful for highly skewed quantitative data
  - Insensitive to outlier data



Measures of Dispersion

- The measures of dispersion measure the differences between how far “spread out” the data values are.
- Two commonly used measures for dispersion are: **range** and **standard deviation.**



Same center, different variation

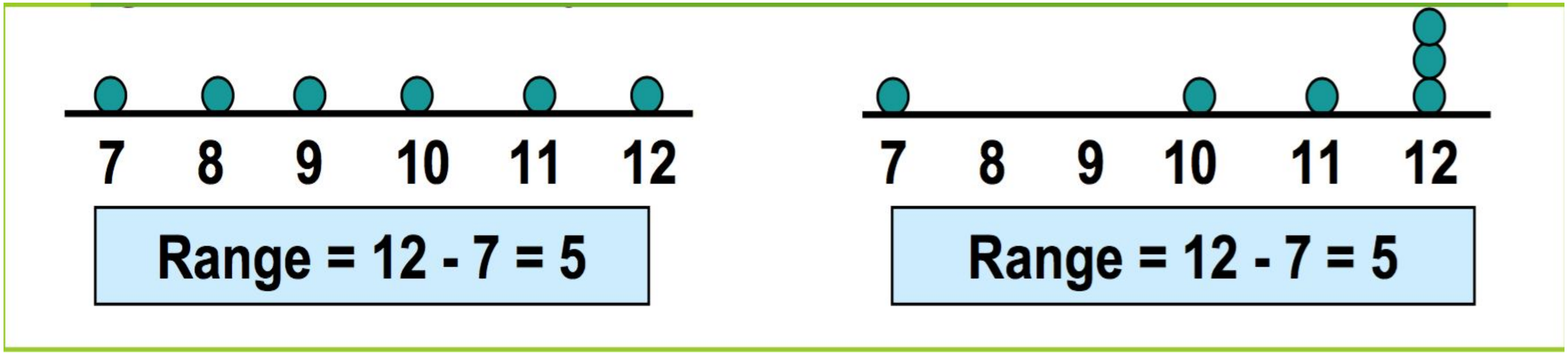
Standard Deviation

- The standard deviation measures the dispersion of a dataset relative to its mean and is calculated as the square root of the variance
- Larger standard deviation means greater variability of the data.

$$\mathrm{SD} = \sqrt{\frac{\sum |x - \bar{x}|^2}{n}}$$

Range

- Range is the difference between the highest and lowest values.
- Since it uses only the extreme values, it is greatly affected by extreme values.
- Range ignores the way in which data are distributed



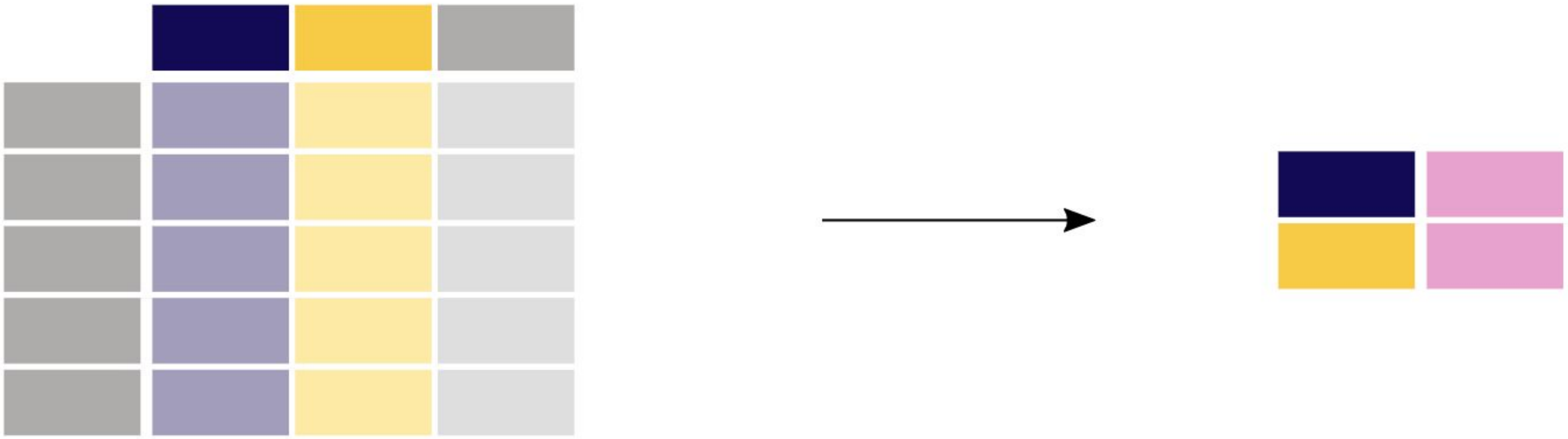
Statistical Functions in Pandas

Pandas offer the following built in statistical functions:

count()	Number of non-null observations
sum()	Sum of values
mean()	Mean of Values
median()	Median of Values
mode()	Mode of values
std()	Standard Deviation of the Values
min()	Minimum Value
max()	Maximum Value
abs()	Absolute Value
prod()	Product of Values
cumsum()	Cumulative Sum
cumprod()	Cumulative Product

Aggregating Statistics

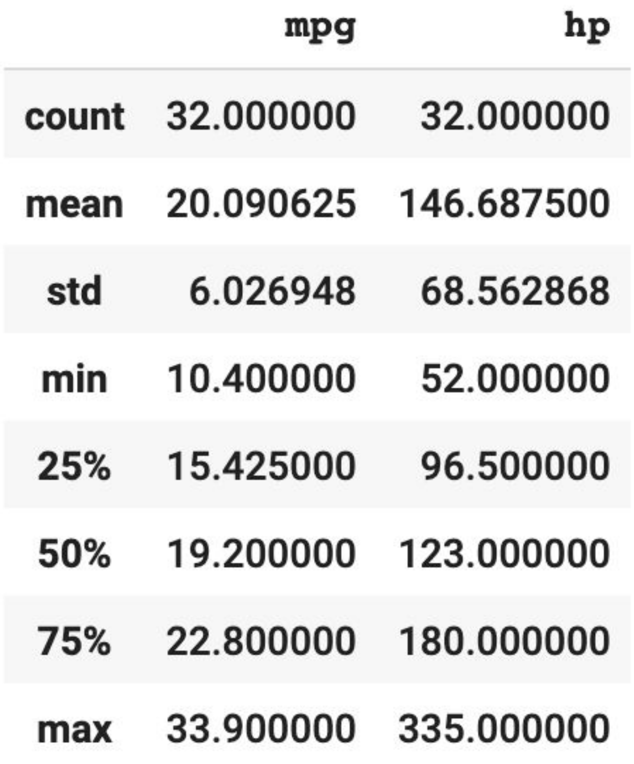
- Different statistics are available and can be applied to columns with numerical data.
- Aggregating statistical operations in general exclude missing data and operate across rows by default
- For example, mtcars_sample[["mpg", "hp"]].median()



Descriptive Statistics

- You can get the descriptive statistics of the data using the describe function
- For example,

```
mtcars_sample[["mpg", "hp"]].median()
mtcars_sample[["mpg", "hp"]].describe()
```

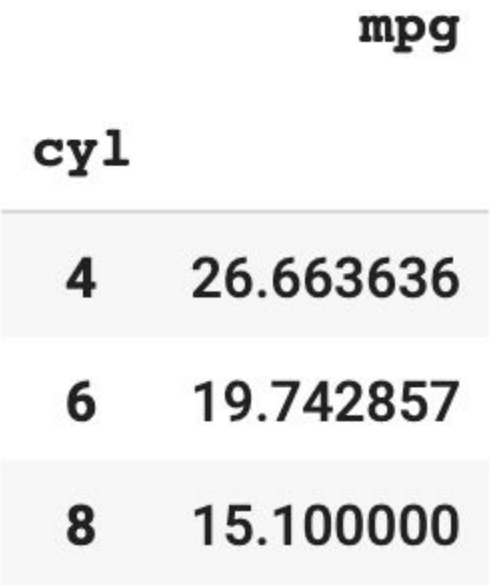


	mpg	hp
count	32.000000	32.000000
mean	20.090625	146.687500
std	6.026948	68.562868
min	10.400000	52.000000
25%	15.425000	96.500000
50%	19.200000	123.000000
75%	22.800000	180.000000
max	33.900000	335.000000

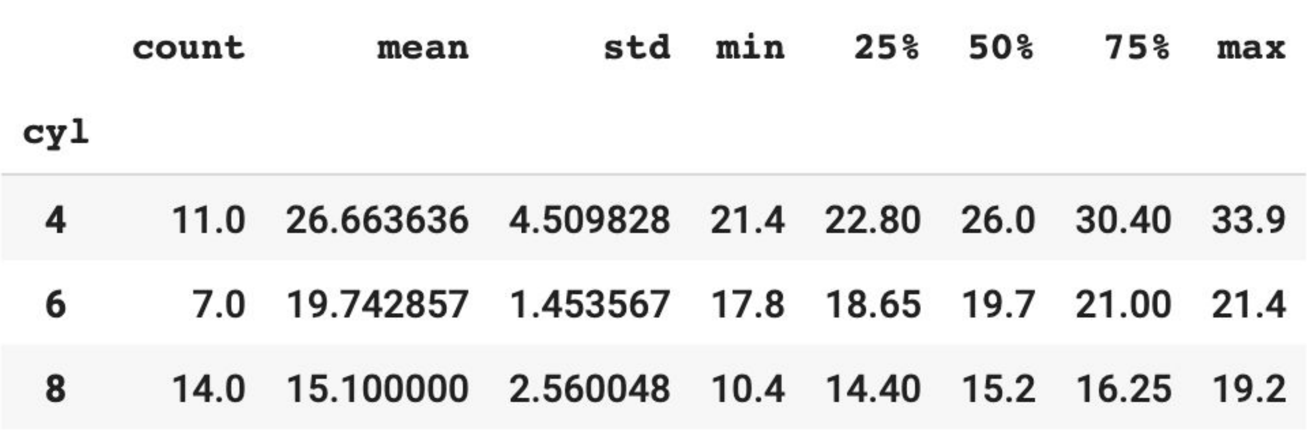
Statistics for Categorical Data

- You can use groupby function to compute the statistics for categorical data
- For example,

```
mtcars_sample.groupby('cyl').mpg.describe()
mtcars_sample.groupby('cyl').mpg.agg(['mean', 'median', 'max'])
mtcars_sample[["cyl", "mpg"]].groupby("cyl").mean()
```



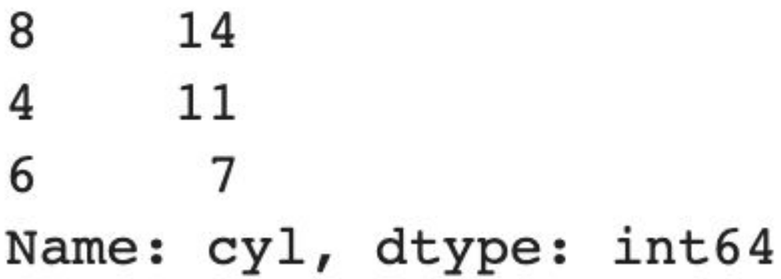
	mpg
cyl	
4	26.663636
6	19.742857
8	15.100000



	count	mean	std	min	25%	50%	75%	max
cyl								
4	11.0	26.663636	4.509828	21.4	22.80	26.0	30.40	33.9
6	7.0	19.742857	1.453567	17.8	18.65	19.7	21.00	21.4
8	14.0	15.100000	2.560048	10.4	14.40	15.2	16.25	19.2

Count

- You can use value_count() to count the number of records in each category
- For example mtcars_sample["cyl"].value_counts()



```
8    14
4    11
6     7
Name: cyl, dtype: int64
```

Activity: Descriptive Statistics

- Import the Singapore long term care facilities data https://raw.githubusercontent.com/tertiarycourses/datasets/master/number-of-residential-long-term-care-facilities-sector-breakdown.csv
- Compute the number of facilities in each ector using the value_counts() method

Time: 10 mins

What is Covariance

- Variance is a measure of the variability or spread in a set of data
- We use the following formula to compute variance for population and sample respectively.

$$Var(x) = \frac{\sum(x-\overline{x})^2}{N} \qquad Var(x) = \frac{\sum(x-\overline{x})^2}{N-1}$$

- **Covariance** is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction.
- We use the following formula to compute covariance for population and sample respectively

$$Cov(x,y) = \frac{\sum(x-\overline{x})(y-\overline{y})}{N} \qquad Cov(x,y) = \frac{\sum(x-\overline{x})(y-\overline{y})}{N-1}$$

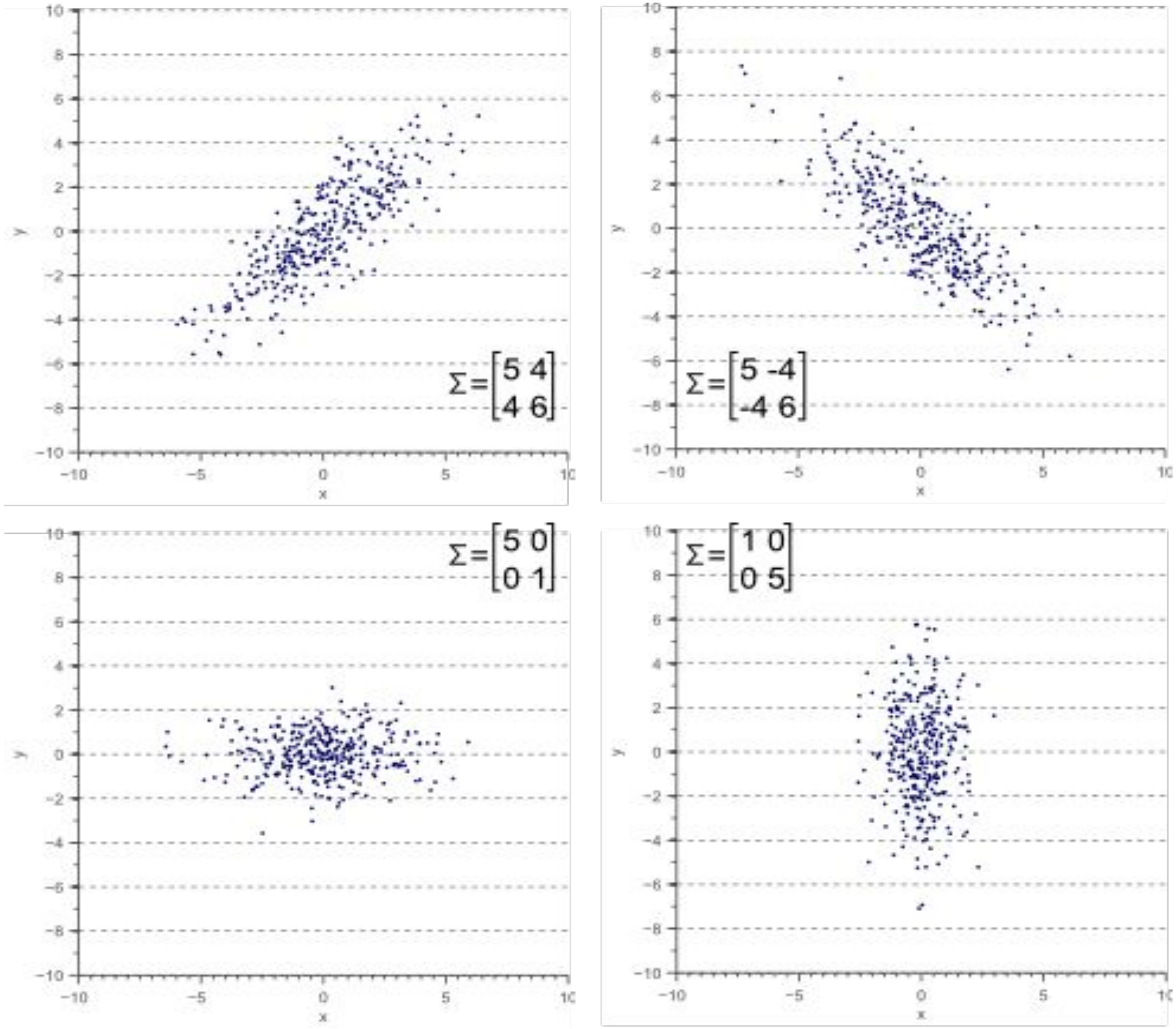
Covariance Matrix

- Variance and covariance are often displayed together in a covariance matrix given as follows:

$$\operatorname{Cov}(A) = \begin{bmatrix} \dfrac{\sum (x_i - \bar{X})(x_i - \bar{X})}{N} & \dfrac{\sum (x_i - \bar{X})(y_i - \bar{Y})}{N} \\ \dfrac{\sum (x_i - \bar{X})(y_i - \bar{Y})}{N} & \dfrac{\sum (y_i - \bar{Y})(y_i - \bar{Y})}{N} \end{bmatrix}$$

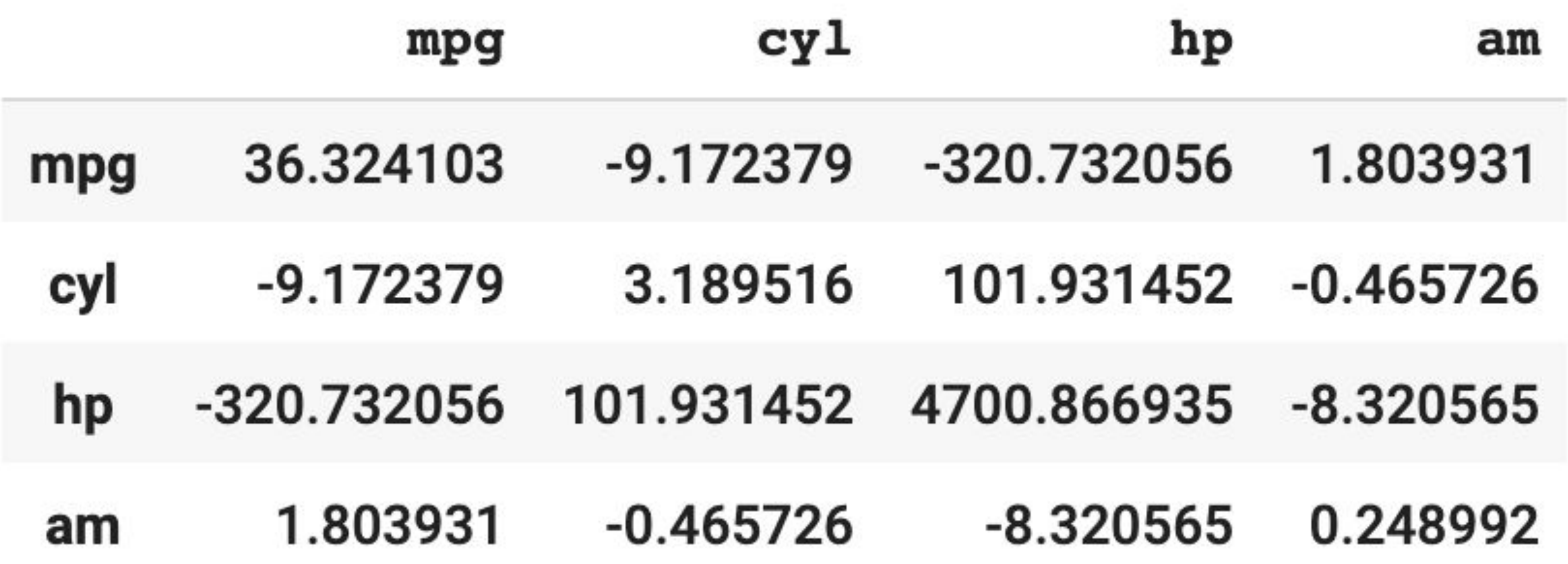
$$= \begin{bmatrix} \operatorname{Cov}(X, X) & \operatorname{Cov}(Y, X) \\ \operatorname{Cov}(X, Y) & \operatorname{Cov}(Y, Y) \end{bmatrix}$$

Covariance Matrix Visualization



Covariance on Pandas

- You can use cov function to compute the covariance matrix
- For example, mtcars_sample.cov()



	mpg	cyl	hp	am
mpg	36.324103	-9.172379	-320.732056	1.803931
cyl	-9.172379	3.189516	101.931452	-0.465726
hp	-320.732056	101.931452	4700.866935	-8.320565
am	1.803931	-0.465726	-8.320565	0.248992

What is Correlation

- The correlation coefficient is also known as the Pearson product-moment correlation coefficient, or Pearson’s correlation coefficient.
- It is obtained by dividing the covariance of the two variables by the product of their standard deviations.

$$Corr(x, y) = \frac{Cov(x, y)}{\sigma_x \sigma_y}$$

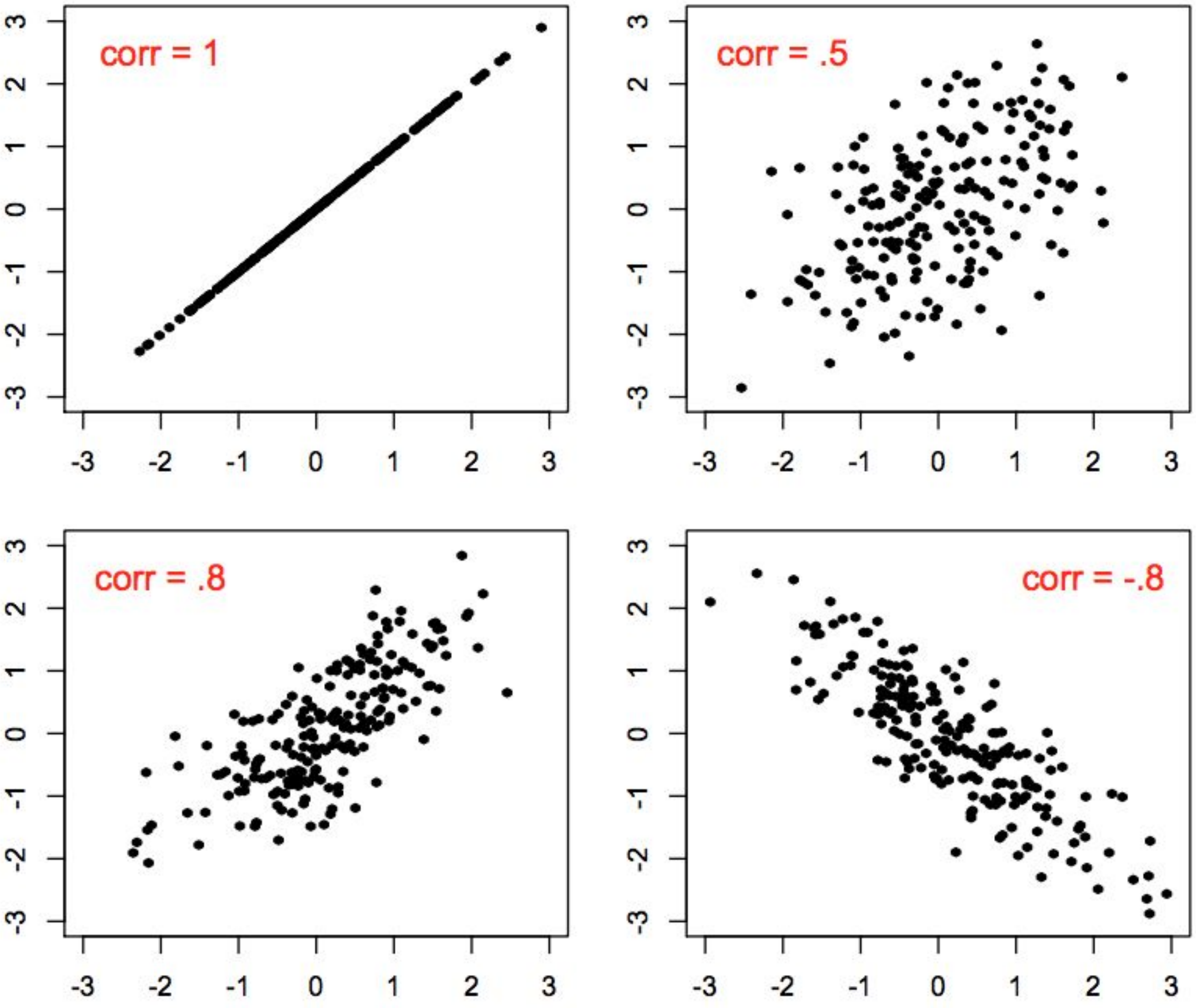
- The values of the correlation coefficient can range from -1 to +1. The closer it is to +1 or -1, the more closely are the two variables are related.
- The positive sign signifies the direction of the correlation i.e. if one of the variables increases, the other variable is also supposed to increase.

Correlation Matrix

- For multiple variables, we can display all the correlation coefficients in the matrix form as below:

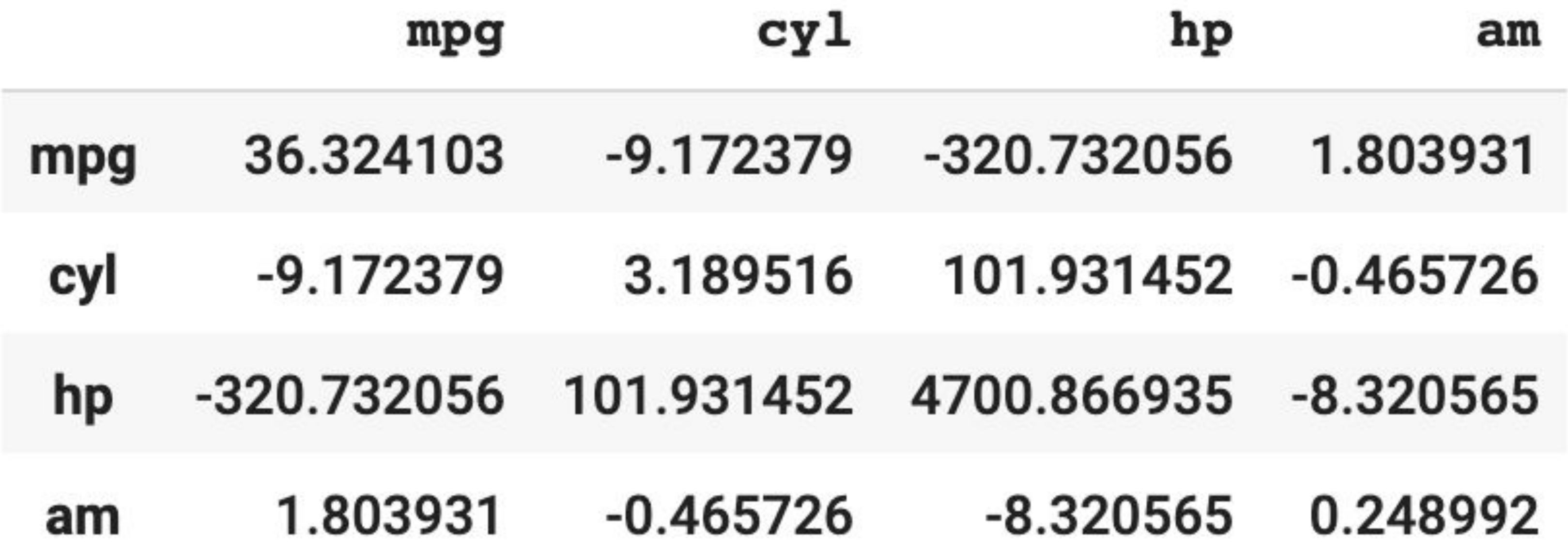
$$\begin{bmatrix} 1 & \mathrm{Corr}(X,Y) & \mathrm{Corr}(X,Z) \\ \mathrm{Corr}(X,Y) & 1 & \mathrm{Corr}(Y,Z) \\ \mathrm{Corr}(X,Z) & \mathrm{Corr}(Y,Z) & 1 \end{bmatrix}$$

Correlation Coefficient



Correlation Matrix on Pandas

- You can use cov function to compute the covariance matrix
- For example, mtcars_sample.cov()



	mpg	cyl	hp	am
mpg	36.324103	-9.172379	-320.732056	1.803931
cyl	-9.172379	3.189516	101.931452	-0.465726
hp	-320.732056	101.931452	4700.866935	-8.320565
am	1.803931	-0.465726	-8.320565	0.248992

Pandas Datetime

- By applying the to_datetime function, pandas interprets the strings and convert these to datetime (i.e. datetime64[ns, UTC]) objects.
- In pandas we call these datetime objects similar to datetime.datetime from the standard library as pandas.Timestamp

```
air_quality = pd.read_csv("https://raw.githubusercontent.com/pandas-dev/pandas/master/doc/data/air_quality_no2_long.csv",parse_dates=["date.utc"])
```

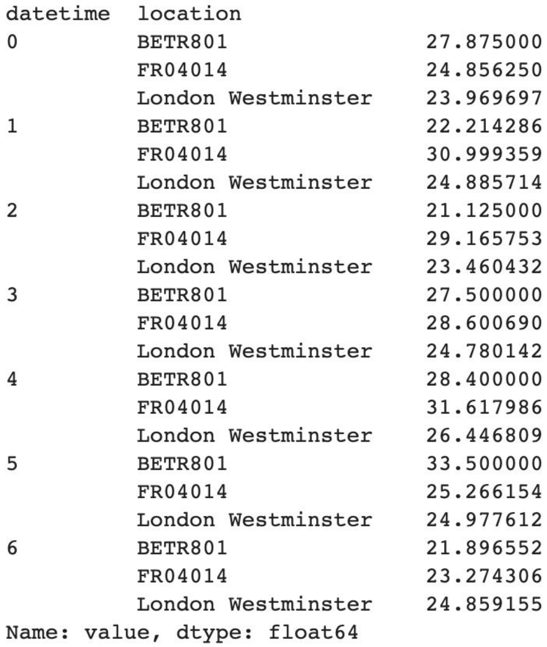
- Using pandas.Timestamp for datetimes enables us to calculate with date information and make them comparable. Hence, we can use this to get the length of our time series Eg

```
air_quality["datetime"].max() - air_quality["datetime"].min()
```

Datetime Properties

- By using Timestamp objects for dates, a lot of time-related properties are provided by pandas. For example the month, but also year, weekofyear, quarter
- All of these properties are accessible by the dt accessor. For example

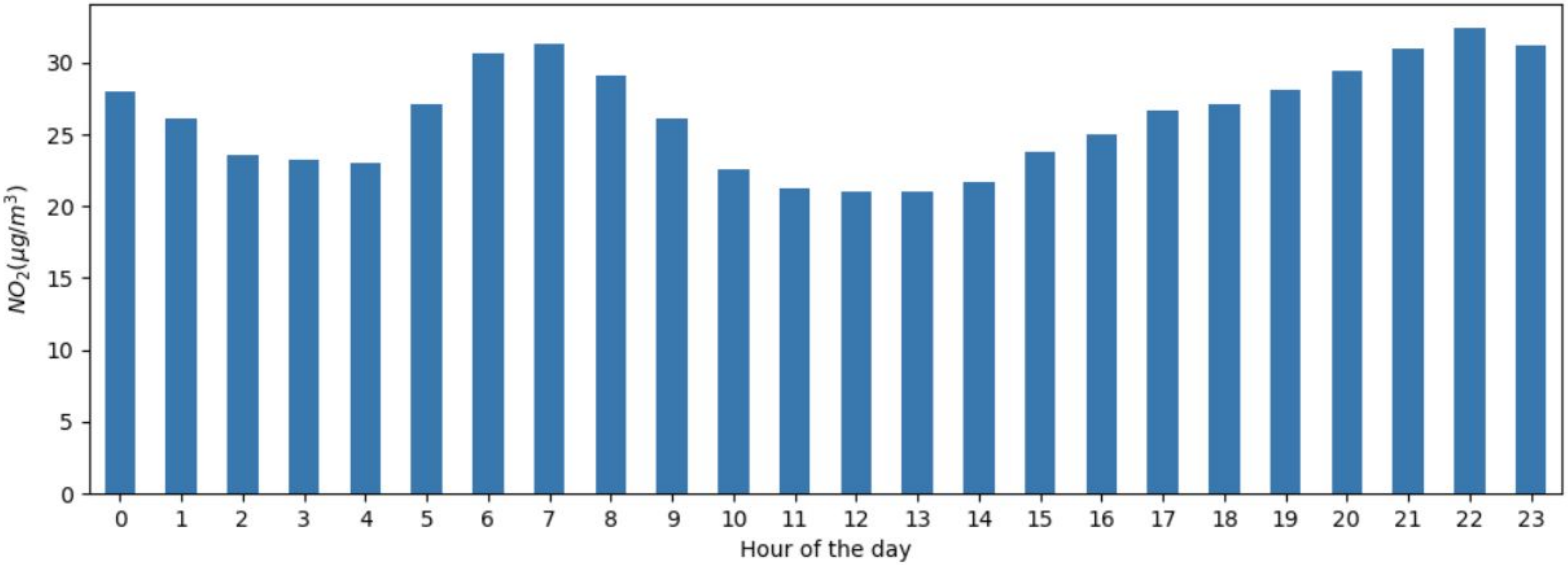
```
air_quality["month"] = air_quality["datetime"].dt.month
air_quality.groupby([air_quality["datetime"].dt.weekday,"location"])
["value"].mean()
```



```
datetime  location
0         BETR801               27.875000
          FR04014               24.856250
          London Westminster    23.969697
1         BETR801               22.214286
          FR04014               30.999359
          London Westminster    24.885714
2         BETR801               21.125000
          FR04014               29.165753
          London Westminster    23.460432
3         BETR801               27.500000
          FR04014               28.600690
          London Westminster    24.780142
4         BETR801               28.400000
          FR04014               31.617986
          London Westminster    26.446809
5         BETR801               33.500000
          FR04014               25.266154
          London Westminster    24.977612
6         BETR801               21.896552
          FR04014               23.274306
          London Westminster    24.859155
Name: value, dtype: float64
```

Time Series Plot

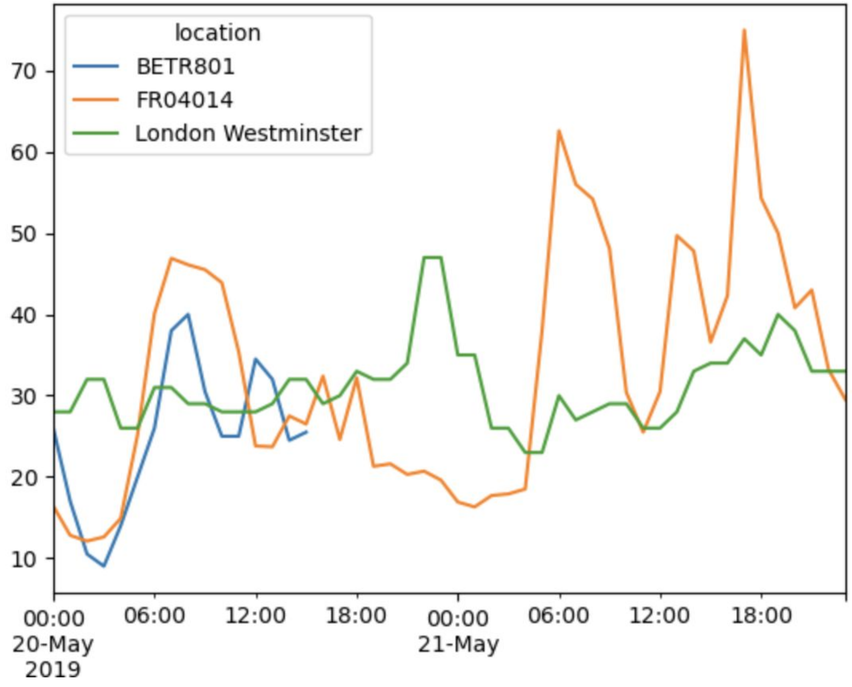
- We can calculate a given statistic (e.g. mean) for each hour of the day
- We can use the groupby and datetime property hour of pandas Timestamp, which is also accessible by the dt accessor.



Datetime as Index

- Working with a datetime index (i.e. DatetimeIndex) provides powerful functionalities.
- For example, we do not need the dt accessor to get the time series properties, but have these properties available on the index directly

```
no_2 = air_quality.pivot(index="datetime", columns="location",
values="value")
no_2["2019-05-20":"2019-05-21"].plot()
```

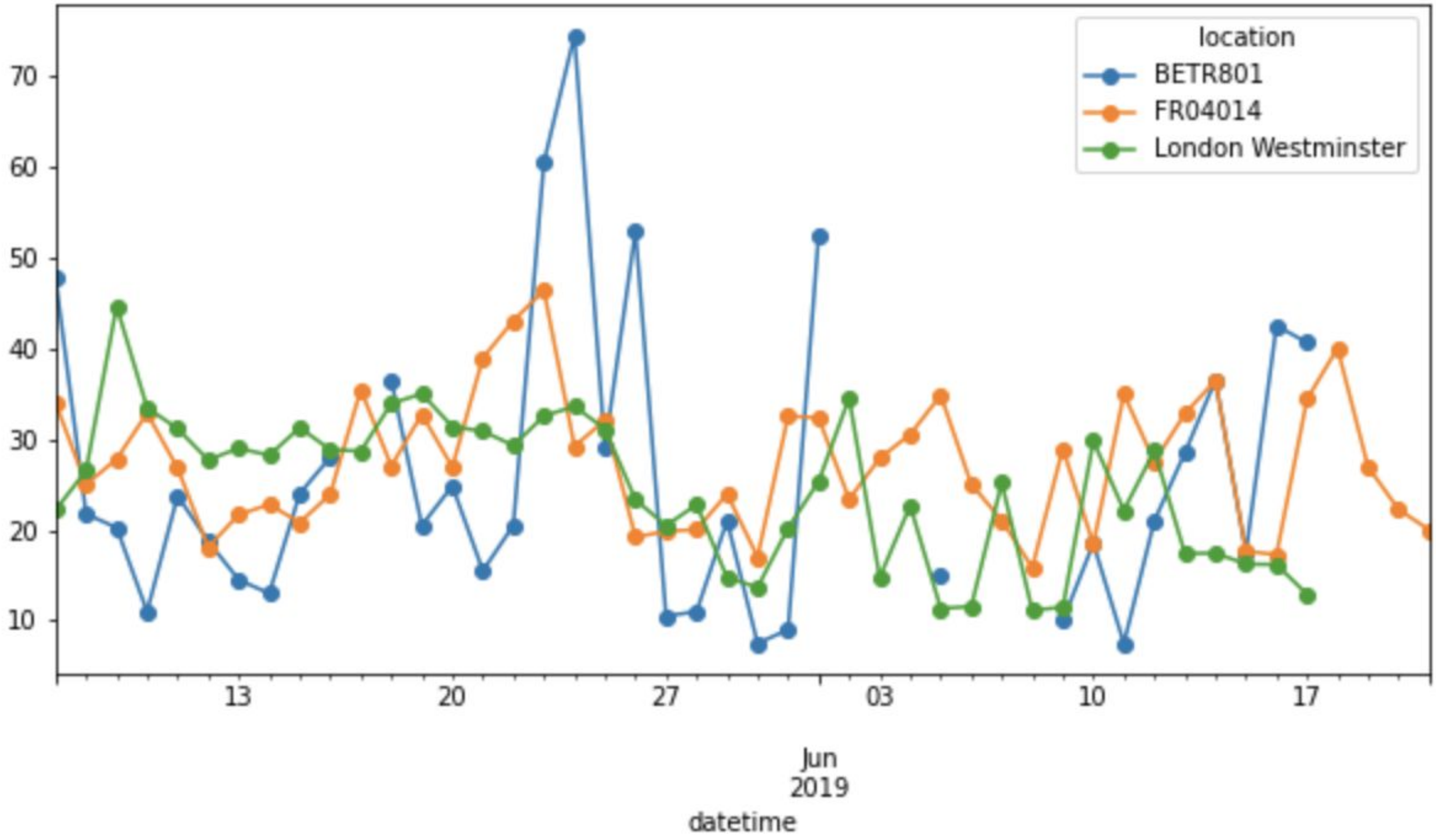


Resample a Time Series

- A very powerful method on time series data with a datetime index, is the ability to resample() time series to another frequency (e.g., converting secondly data into 5-minutely data).
- The resample() method is similar to a groupby operation:
  - it provides a time-based grouping, by using a string (e.g. M, 5H,…) that defines the target frequency
  - it requires an aggregation function such as mean, max,…

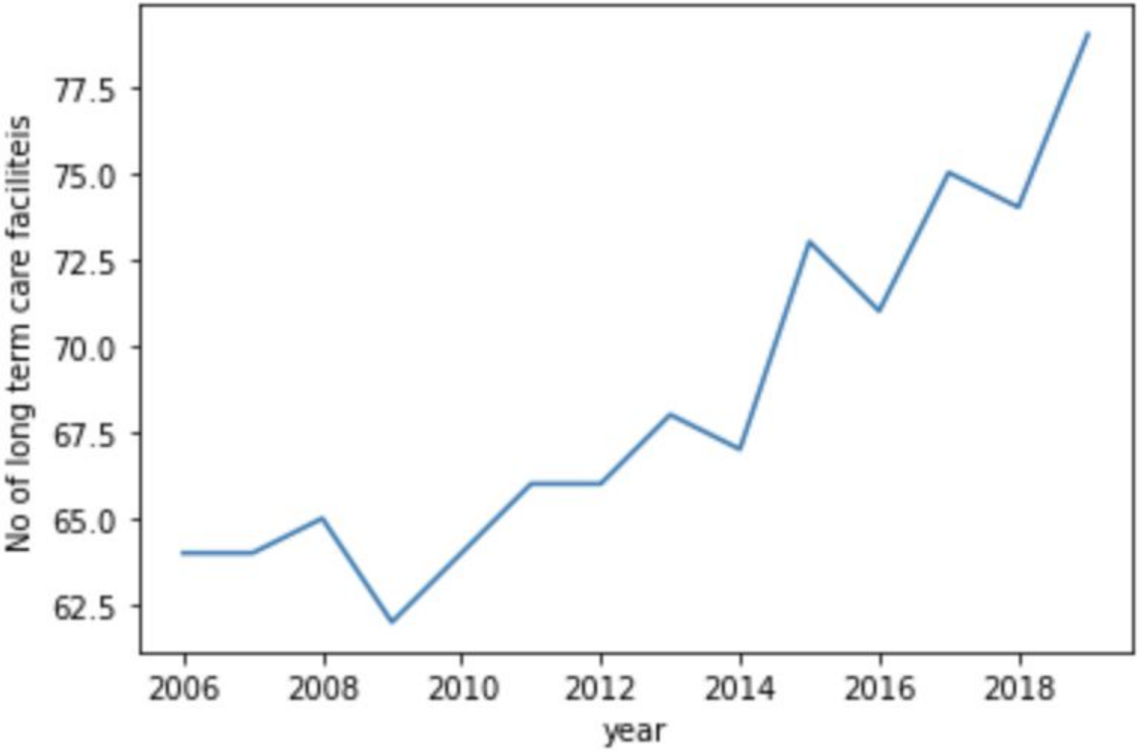
Resampled Time Series Plot

```
no_2.resample("D").mean().plot(style="-o", figsize=(10, 5));
```



Activity: Time Series Analysis

- Import the Singapore long term care facilities data https://raw.githubusercontent.com/tertiarycourses/datasets/master/number-of-residential-long-term-care-facilities-sector-breakdown.csv
- Plot the total no of long term care facilities vs year



Summary

Q&A

Feedback

https://goo.gl/R2eumq

Thank You!

Marcus Lee Yi Qing
87119800
makasulee@gmail.com