Programming with Python

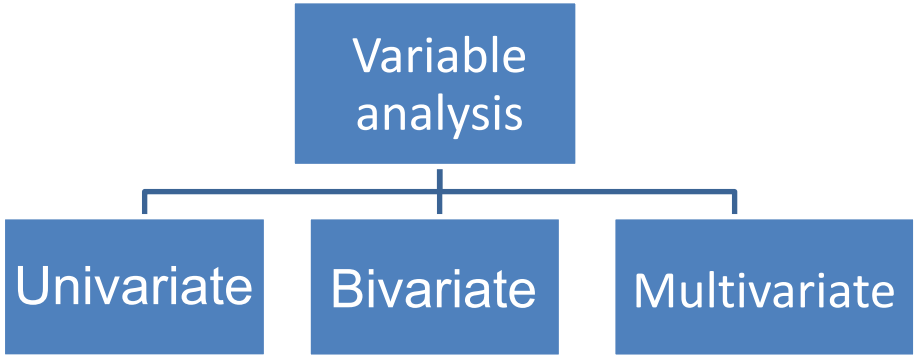
Data analysis and preprocessing

Agenda

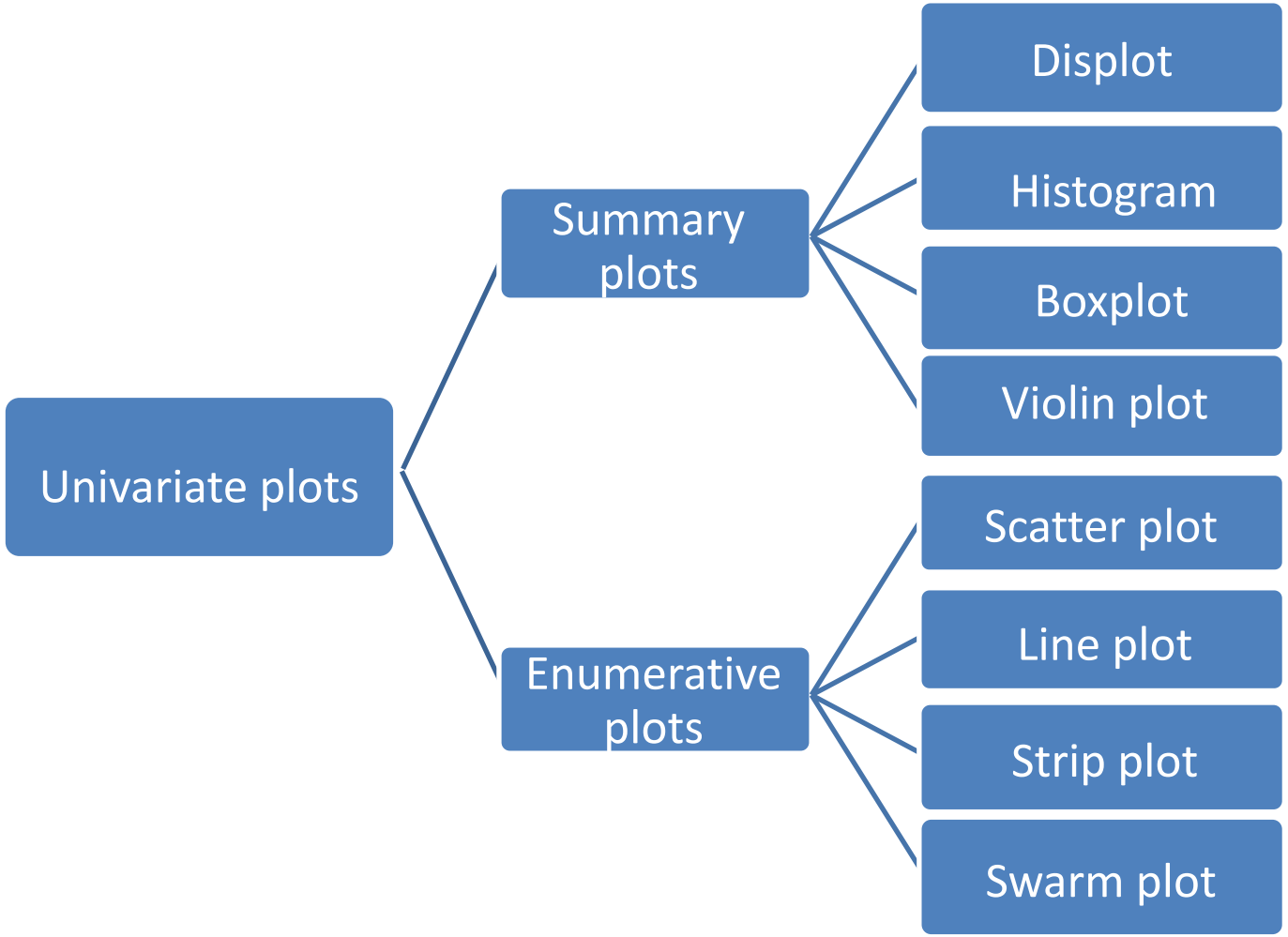
- Data analysis in Python
- Univariate, bivariate and multivariate analysis
- Data normalization and scaling
- One hot encoding
- Missing values imputation
- Working with outliers
- Summary

Data analysis in Python

- Visualization - Fastest way of analyzing data
- Helps in understanding distribution of variables in the data
- Correlation – helps in analyzing correlation between each variable in the dataset
- Python visualization libraries – Matplotlib, Seaborn, Plotly
- Variables category : Univariate, Bivariate, Multivariate



Univariate analysis

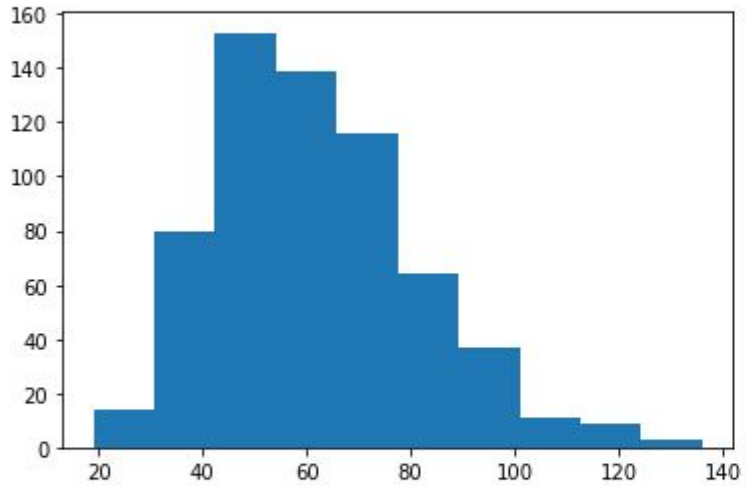


- Simplest form of analyzing data
- Helps to understand visible and descriptive details of a “single” variable involved
- Mainly describes pattern in data

Univariate example

Through histogram plot, we can see that the max. honey produced for a price of 50 pounds was around 150

```
plt.hist(df["yieldpercol"])
plt.show()
```



Also, we can calculate the average honey yield per colony from 1998 to 2012

```
df["yieldpercol"].mean()
```

```
62.00958466453674
```

The total stocks(in pounds) held by producers are:

```
df["stocks"].sum()
```

```
825606000.0
```

The total honey production in pounds is:

```
df["totalprod"].sum()
```

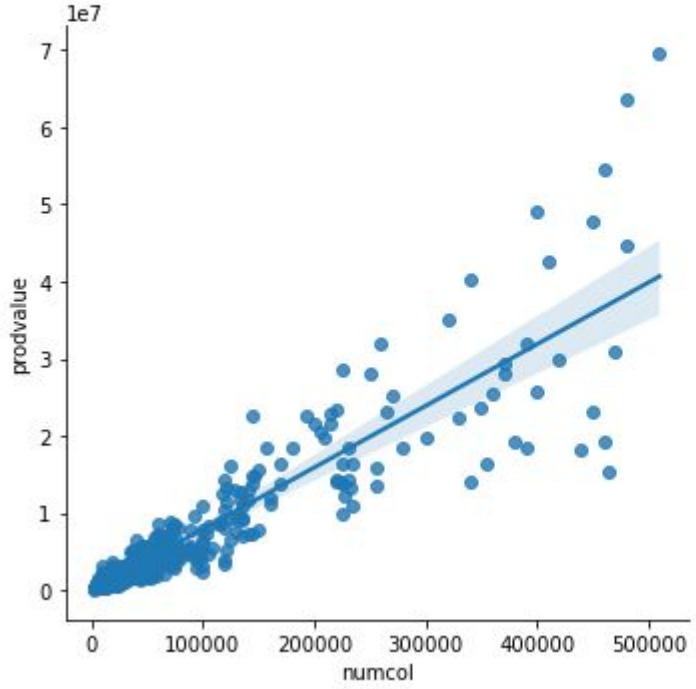
```
2609848000.0
```

Bivariate analysis

- Helps to understand relationship between two variables
- Visualization/Tabular based methods:
  - Scatter plot - analysis of two numerical variables
  - Crosstab - analysis of two categorical variables
  - Boxplot - analysis of a numerical variable and a categorical variable
- Statistical tests:
  - Z-test and t-test – analysis of numerical and categorical variables
  - Chi – square test - analysis of two categorical variables

We can visualize relationship between the Number of Colonies & Value in Production :

```
sns.lmplot(x="numcol", y="prodvalue", data=df)
plt.show()
```

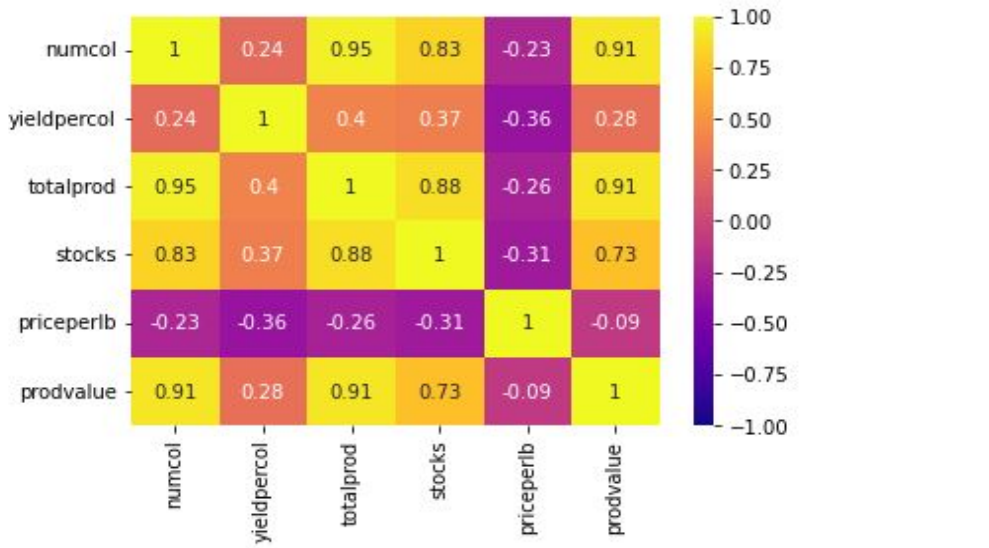


Correlation

- To find if variables are related in any way
- Pandas corr() method - generates correlation matrix
- Seaborn heatmap() - helps to visualize correlation

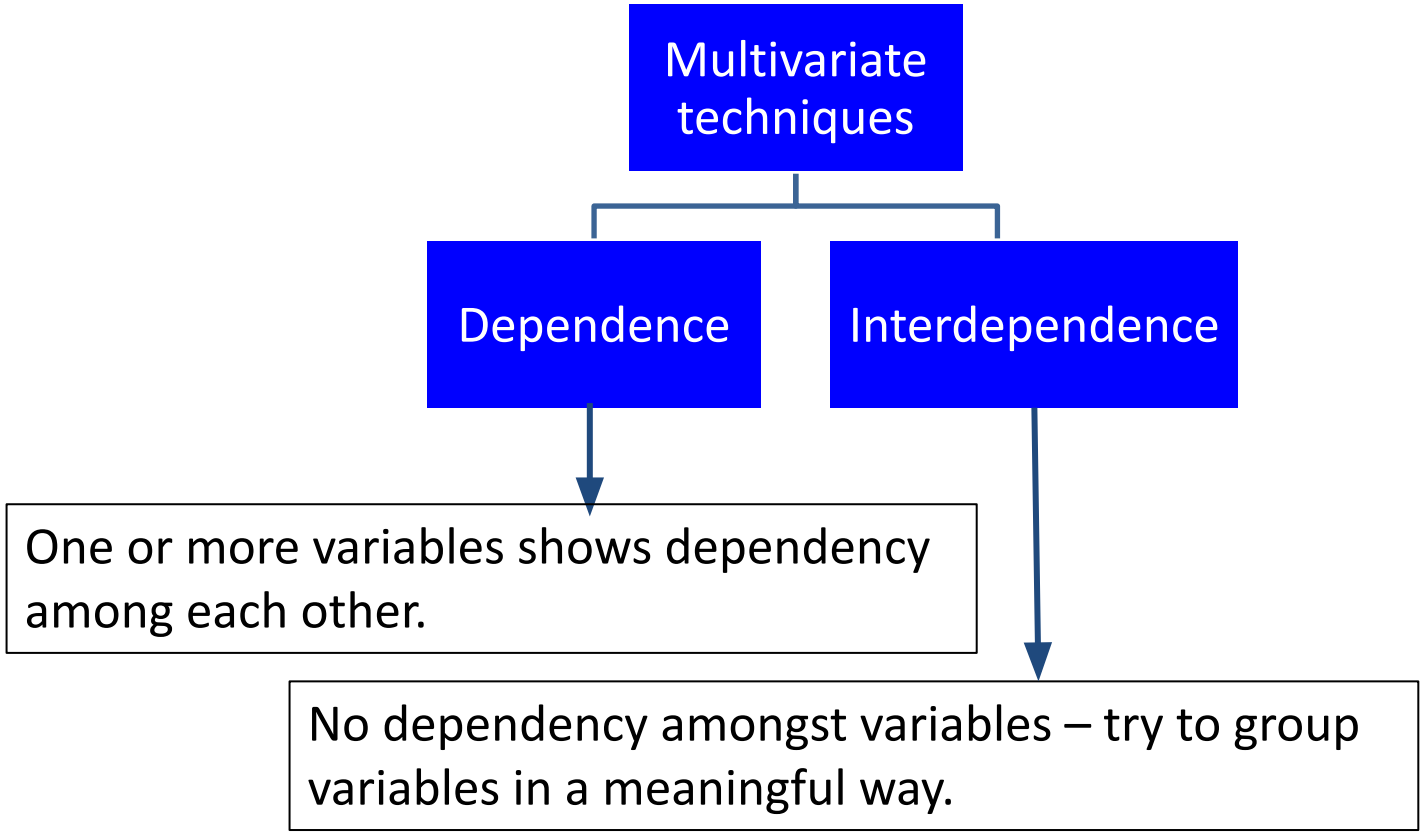
	numcol	yieldpercol	totalprod	stocks	priceperlb	prodvalue
numcol	1.000000	0.243515	0.953594	0.825929	-0.232701	0.912796
yieldpercol	0.243515	1.000000	0.396252	0.367812	-0.358646	0.278977
totalprod	0.953594	0.396252	1.000000	0.878830	-0.264499	0.907236
stocks	0.825929	0.367812	0.878830	1.000000	-0.305867	0.728560
priceperlb	-0.232701	-0.358646	-0.264499	-0.305867	1.000000	-0.089567
prodvalue	0.912796	0.278977	0.907236	0.728560	-0.089567	1.000000

```
# heatmap helps in visualizing correlation amongst features
sns.heatmap(corr_matrix,annot=True,cmap='plasma',vmin=-1,vmax=1)
plt.show()
```



Multivariate analysis

- More than two variables analysis
- Helps to understand complex data
- Conclusion drawn are more realistic
- Visualization methods:
  - Pairplots
  - Plotly plots
- Various techniques – MANOVA (Multivariate analysis of variance), Multiple linear regression, Factor analysis, Cluster analysis, etc.

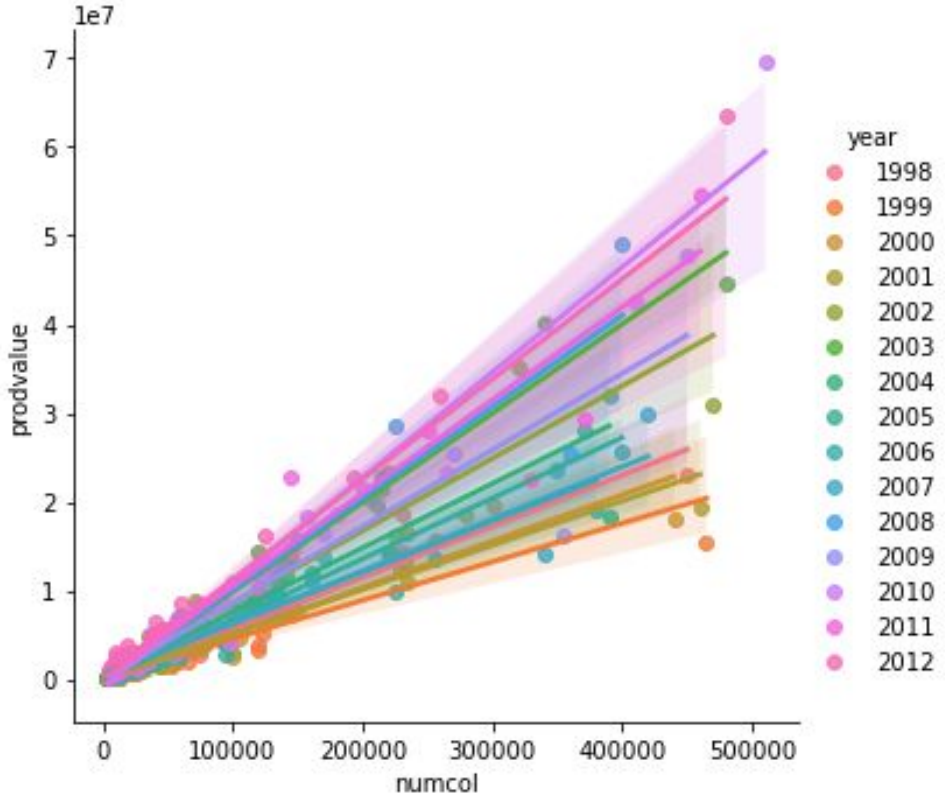


Multivariate example

We can visualize relationship between the Number of Colonies & Value in Production at an overall level, at state and year levels as well :

```
sns.lmplot(x="numcol", y="prodvalue", data=df, hue='year')
```

```
<seaborn.axisgrid.FacetGrid at 0x19acad51fa0>
```



Data normalization and scaling

- Data normalization – Transforms different scaled data to common scale
- Min max scaling (fixed range 0 to 1)
- Standard scaling also called - z–score (ranges from -3 to 3)
- Max absolute scaling (ranges between -1 to 1)

```
# Max absolute Scaling using sklearn's MaxAbsScaler
from sklearn.preprocessing import MaxAbsScaler
max_abs = MaxAbsScaler()
max_abs.fit(df_scale)
```

```
MaxAbsScaler()
```

```
# max absolute values generated by fit method
max_abs.max_abs_
```

```
array([1.86e+01, 3.55e+05])
```

```
# transform the data and store it in dataframe
scaled_reading = max_abs.transform(df_scale)
scaled_df = pd.DataFrame(scaled_reading, columns = df_scale.columns)
scaled_df
```

	fuel	meter_reading
0	0.537634	0.563380
1	0.623656	0.507042
2	0.962366	1.000000
3	1.000000	0.845070

Data preprocessing - One hot encoding

- Almost all machine learning algorithms needs categorical conversion
- Categorical to integer number mapping
- Each categorical value is mapped to new columns with binary values 0 or 1
- Useful for data which has no relations
- Pandas get_dummies() method
- Other methods:-
  - Manual encoding
  - Label encoding

```
df = pd.DataFrame({"Id": [1,2,3,4,5], "Name": ["Alex", "Raman", "Samina", "Rathi", "Sam"],
                   "Gender": ["Male", "Male", "Female", "Female", "Male"]})
df
```

	Id	Name	Gender
0	1	Alex	Male
1	2	Raman	Male
2	3	Samina	Female
3	4	Rathi	Female
4	5	Sam	Male

```
# creates dummy columns for each categorical value in Gender feature
df = pd.get_dummies(df["Gender"])
df
```

	Female	Male
0	0	1
1	0	1
2	1	0
3	1	0
4	0	1

Missing values imputation

- For unknown reasons data contain missing values
- Blank space, nan values
- With price of losing data, one can discard missing value rows and/or columns
- Better approach is to impute missing values with mean, median, mode or a constant

```
import numpy as np
from sklearn.impute import SimpleImputer

# create object for simpleimputer with mean as replacement for missing values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

# fit on data to get mean values
imputer.fit([[4, 2], [np.nan, 6], [5, 3], [np.nan, np.nan]])
```

```
SimpleImputer()
```

```
# transform the values
imputer.transform([[4, 2], [np.nan, 6], [5, 3], [np.nan, np.nan]])
```

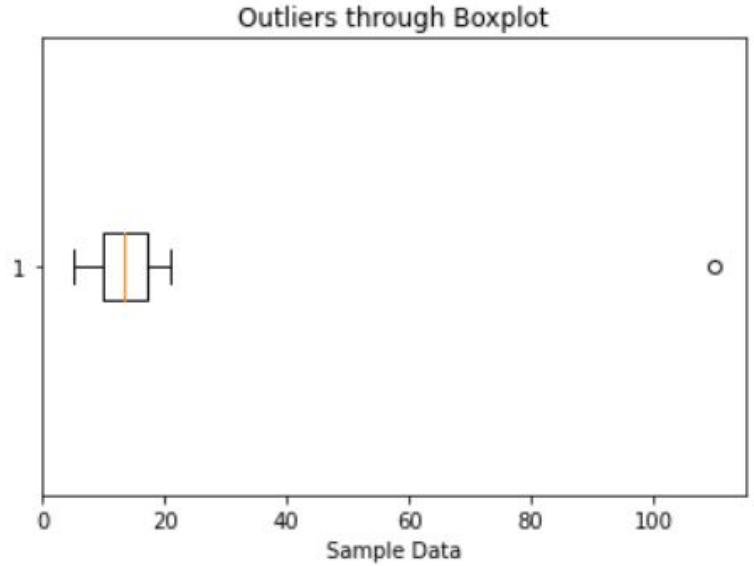
```
array([[4.        , 2.        ],
       [4.5       , 6.        ],
       [5.        , 3.        ],
       [4.5       , 3.66666667]])
```

Dealing with outliers

- “Odd one Out”
- Mean – only measure which is affected which in turns affect Standard deviation
- Negatively affects training data of a machine learning model
- Can be identified through – Box plot, z-score, Interquartile range (IQR)
- Can be removed or imputed

```
import matplotlib.pyplot as plt

plt.boxplot(list1, vert=False)
plt.title("Outliers through Boxplot")
plt.xlabel('Sample Data')
plt.show()
```



Summary

- Discussed univariate, bivariate and multivariate analysis.
- Normalization and scaling in Python.
- Data preprocessing such as one hot encoding, missing value imputation, dealing with outliers.

Hands On

THANK YOU

Happy Learning ☺️