

Chapter 2: Exploratory Data Analysis (Analysis)

Introduction	2
Univariate Analysis	3
Histograms	4
Bar charts/Count Plots	4
Exercise 2.01: EDA – Univariate Analysis	5
Bivariate Analysis	9
Types:	9
Bivariate Analysis of two numerical variables	9
Scatter Plot:	9
Linear Correlation	9
Bivariate Analysis of two categorical variables	9
Stacked Column Chart	9
Combination Chart	9
Bivariate Analysis of a numerical variable and a categorical variable	10
Line chart with Error Bars	10
Combination Chart	10
Z-test and T-test	10
Bivariate Correlations	11
Exercise 2.02: EDA – Bivariate Analysis	11
Multivariate Analysis	13
Types	13
Cluster Analysis	13
Correspondence Analysis	13
Principal Component Analysis	13
Single Index:	14
Multiple Index	14
Aggregates	15
Exercise 2.03: Multivariate analysis	15
Automated EDA Tooling	19
Exercise 2.04: Automated EDA Tooling	20

Activity 2.01: Loading the “City Land Mass - Miami” Dataset and performing Univariate and Bivariate Analysis on it. 22

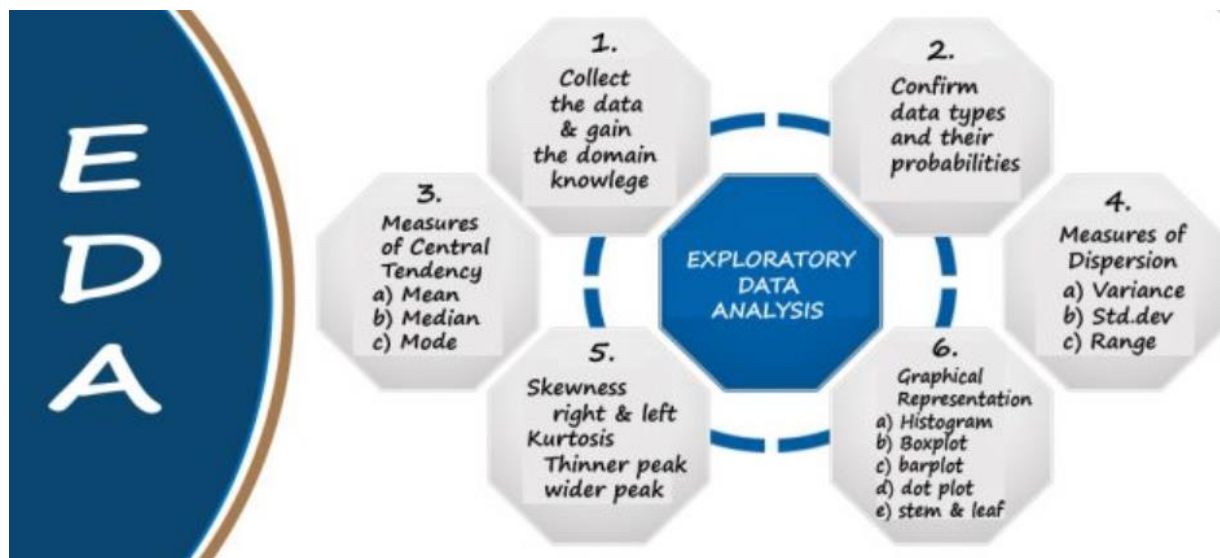
Conclusion 23

Introduction

In the previous chapter, we learned about the aspects related to Exploratory Data Analysis. We were analyzing the datasets by making some investigations that were focusing on the structure, quality, content, and quantity of that data. Now in this chapter, we will be focusing on the more advanced structure of EDA. We will be practicing the concepts through exercises.

Exploratory data analysis is a data exploration approach that encourages statisticians to explore their data and possibly formulate hypotheses that might cause new data collection and experiments. EDA is a robust and efficient approach for checking that your data contains all the required information for model fitting and hypothesis testing. It also provides you with an understanding of how to handle missing values and make transformations of variables as needed.

It provides a solid base to build a robust understanding of the data, and issues associated with either the info or process. EDA is an innovative way to solve business problems by researching, planning, and creating solutions.



EDA is classified in two different ways where each method is either graphical or non-graphical. Moreover, each method is either **univariate**, **bivariate** or **multivariate**. These three methods will be covered in this chapter.

The data set that will be used in this chapter contains the following features:

- The name, state, district #, party, room #, phone #, and committee assignments of the US House of Representatives.

Also referred to as a congressman or congresswoman, each representative is elected to a two-year term serving the people of a specific congressional district. The number of voting representatives in the House is fixed by law at no more than 435, proportionally representing the population of the 50 states. Currently, there are five delegates representing the District of Columbia, the Virgin Islands, Guam, American Samoa, and the Commonwealth of the Northern Mariana Islands. A resident commissioner represents Puerto Rico. Learn more about representatives at The House Explained. You can learn more about this dataset from [here](#).

Univariate Analysis

Univariate analysis is a type of descriptive statistics that deal with one variable from a large amount of data. The objective is to derive the data; define and summarize it; and analyze its mean, variance, covariance, or correlation.

In a dataset, each variable is explored separately. It is possible for two kinds of variables:

- Categorical
- Numerical

The most common patterns to be identified when performing a univariate analysis are mean, mode and median. For example, if we were studying the air quality around a city that is experiencing heavy smog levels, we could identify the central tendency of a sample ($n = 3$) of people's perception of the air quality by looking at their answers to the question: "How clean or dirty did you feel while you were driving?"

Similarly, Dispersion (range, variance), Quartiles (interquartile range), and Standard deviation can also be used for this analysis.

Before going into the details of the practice exercise, let us take a look at the data. This data has the following features:

```
int64 columns:
  Index(['District'], dtype='object')

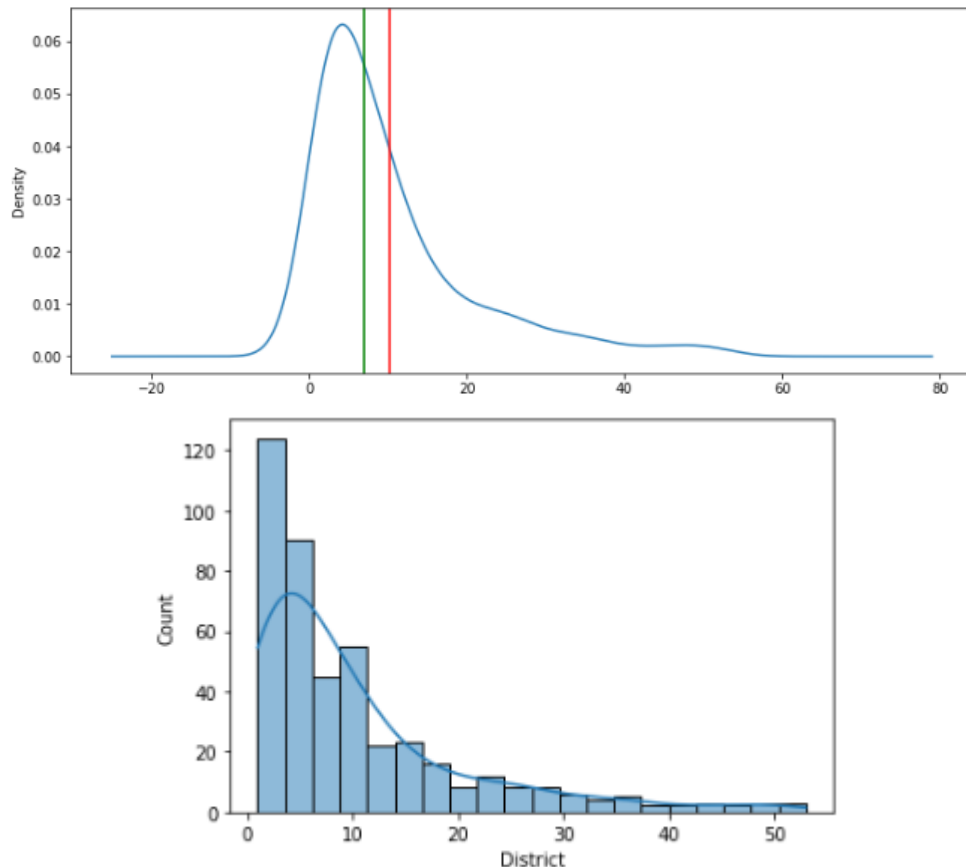
float64 columns:
  Index([], dtype='object')

object columns:
  Index(['Name', 'State', 'Party', 'Room', 'Phone', 'House Positions',
        'Committee Assignment'],
        dtype='object')
```

Histograms

Histograms are similar to bar charts, displaying similar categorical variables against the category of data. They display the categories in the form of bins which indicate the number of data points within a range. They are used to represent a range of values and the frequency of those values within an interval.

In the upcoming exercises, we will be working on the histogram plots of continuous variables. Some outputs have been shown here:

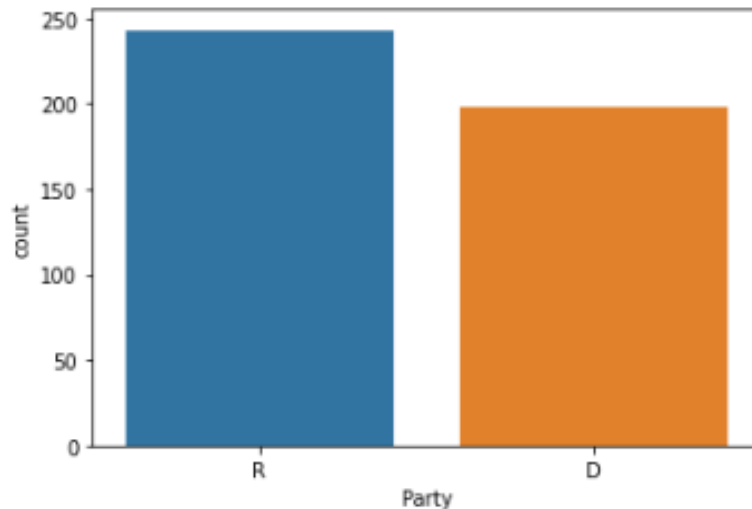


Bar charts/Count Plots

The bar graph is very convenient when comparing categories of data or different groups of data. It lets users track the changes over time. It is best for visualizing discrete data. For the bar charts following syntax will be used:

```
sns.countplot(x="Party", data=df)
df['Party'].value_counts()
```

It has provided the result as shown in the figure. It is for the party variable:



You can learn more about Univariate analysis from [here](#).

Exercise 2.01: EDA – Univariate Analysis

In this exercise, a univariate analysis will be done by plotting histograms and bar charts. Load the dataset, import visualization libraries, and perform the analysis upon one variable out of all the variables of the given dataset.

First, perform the steps mentioned below to cover the major aspects of exploratory data analysis techniques:

1. Firstly, open the collab notebook and load the dataset from GitHub as shown below:

```
df = pd.read_csv('https://raw.githubusercontent.com/fenago/datawrangling/main/Chapter%202/Directory-of-Representatives-Jan-17.csv')
df.head(5)
df.info()
```

Output will be as follows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 441 entries, 0 to 440
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Name                  441 non-null   object
1   State                 441 non-null   object
2   District              441 non-null   int64
3   Party                 441 non-null   object
4   Room                  441 non-null   object
5   Phone                 441 non-null   object
6   House Positions       10 non-null    object
7   Committee Assignment  55 non-null    object
dtypes: int64(1), object(7)
memory usage: 27.7+ KB
```

2. Import the visualization libraries along with the other required ones.
3. Clean the dataset if required and handle with the data.

4. Get the total number of null values as shown below:

```
df.isnull().sum()
print(df.District)
```

Output will be as follows:

```
0      1
1      2
2      3
3      4
4      5
..
436    5
437    6
438    7
439    8
440   10
Name: District, Length: 441, dtype: int64
```

5. Rename the columns as shown below:

```
df = df.rename(columns={'Committee Assignment': 'Assignment',
                        'House Positions': 'Positions'})
df.head()
```

Output will be as follows:

	Name	State	District	Party	Room	Phone	Positions	Assignment
0	Byrne, Bradley	Alabama	1	R	119 CHOB	202-225-4931	NaN	Rules
1	Roby, Martha	Alabama	2	R	442 CHOB	202-225-2901	NaN	NaN
2	Rogers, Mike	Alabama	3	R	2184 RHOB	202-225-3261	NaN	NaN
3	Aderholt, Robert	Alabama	4	R	235 CHOB	202-225-4876	NaN	NaN
4	Brooks, Mo	Alabama	5	R	2400 RHOB	202-225-4801	NaN	NaN

6. Perform univariate analysis of continuous variables as shown below:

```
integer_columns = df.select_dtypes(include=['int64']).columns
float_columns = df.select_dtypes(include=['float64']).columns
object_columns = df.select_dtypes(include=['object']).columns
```

```
# display columns
print('\nint64 columns:\n',integer_columns)
print('\nfloat64 columns:\n',float_columns)
print('\nobject columns:\n',object_columns)

# Remove the .columns and it saves it as a dataframe
num_features = df.select_dtypes(exclude=['object'])
cat_features = df.select_dtypes(include=['object'])
type(num_features)
type(cat_features)
cat_features
```

Output will be as follows:

```
int64 columns:
Index(['District'], dtype='object')

float64 columns:
Index([], dtype='object')

object columns:
Index(['Name', 'State', 'Party', 'Room', 'Phone', 'Positions', 'Assignment'], dtype='object')
```

	Name	State	Party	Room	Phone	Positions	Assignment
0	Byrne, Bradley	Alabama	R	119 CHOB	202-225-4931	NaN	Rules
1	Roby, Martha	Alabama	R	442 CHOB	202-225-2901	NaN	NaN
2	Rogers, Mike	Alabama	R	2184 RHOB	202-225-3261	NaN	NaN
3	Aderholt, Robert	Alabama	R	235 CHOB	202-225-4876	NaN	NaN
4	Brooks, Mo	Alabama	R	2400 RHOB	202-225-4801	NaN	NaN
...
436	Sensenbrenner, F. James	Wisconsin	R	2449 RHOB	202-225-5101	NaN	NaN
437	Grothman, Glenn	Wisconsin	R	1217 LHOB	202-225-2476	NaN	NaN
438	Duffy, Sean P.	Wisconsin	R	2330 RHOB	202-225-3365	NaN	NaN
439	Gallagher, Mike	Wisconsin	R	1007 LHOB	202-225-5665	NaN	NaN
440	Cheney, Liz	Wyoming	R	416 CHOB	202-225-2311	NaN	Rules

441 rows x 7 columns

7. Make a list of numeric features from data as shown below:

```
[ ] num_list = ['District']
num_list
```

8. Make a list of categorical features from data as shown below:

```
[ ] cat_list = ['Name', 'State', 'Party', 'Room', 'Phone', 'House Positions',
               'Committee Assignment']
```

9. Remove the .columns and save it as a data frame as shown below:

```
num_features = df.select_dtypes(exclude=['object'])
cat_features = df.select_dtypes(include=['object'])
type(num_features)
type(cat_features)
num_features.describe(include='all')
```

Output will be as follows:

	District
count	441.000000
mean	10.247166
std	10.395810
min	1.000000
25%	3.000000
50%	7.000000
75%	13.000000
max	53.000000

10. Plot histograms as shown below:

```
[16] plt.figure(figsize=(12,5))
df['District'].plot(kind='density')
plt.axvline(x=df.District.mean(),label='mean',color='Red')
plt.axvline(x=df.District.median(),label='median',color='Green')
plt.boxplot
```

11. Show the counts of observations in each categorical bin using bars as shown below:

```
sns.countplot(x="Name", data=df)
#df['Name'].value_counts()
```


Bivariate Analysis

Bi means two and variate means variable, so here there are two variables. The analysis is related to the cause and the relationship between the two variables.

Types:

- **Bivariate Analysis of two numerical variables**

This type of bivariate analysis includes both dependent and independent variables with their numerical values. Some of its visual representation tools are mentioned below. These tools are visualized in the upcoming exercise.

Scatter Plot:

It is drawn before working out a linear correlation or fitting a regression line. It is a useful representation of the relationship between two numerical variables.

Linear Correlation

It quantifies the strength of a linear relationship between the two numerical variables. Without any correlation between the two variables, there is no tendency for the increase or decrease in the values of one variable with the increase or decrease in the values of another variable.

- **Bivariate Analysis of two categorical variables**

This type of bivariate analysis includes both dependent and independent variables with their categorical values.

Stacked Column Chart

It is a visualization graph to see the relationship between two categorical variables. It makes a comparison between the percentage that each category from one variable contributes to overall such categories of the second variable.

Combination Chart

It is also a visualization tool that utilizes two or more chart types to emphasize that charts contain different types of information. This type of chart is the best visualization method to demonstrate the predictability power of a predictor on the X-axis against a target on the Y-axis.

- **Bivariate Analysis of a numerical variable and a categorical variable**

This type of bivariate analysis includes one variable as categorical and the other variable as numerical.

Line chart with Error Bars

It displays the information as a series of data points connected by straight-line segments. Each data point is the average of the numerical data for the relevant category of the categorical variable with an error bar showing standard error.

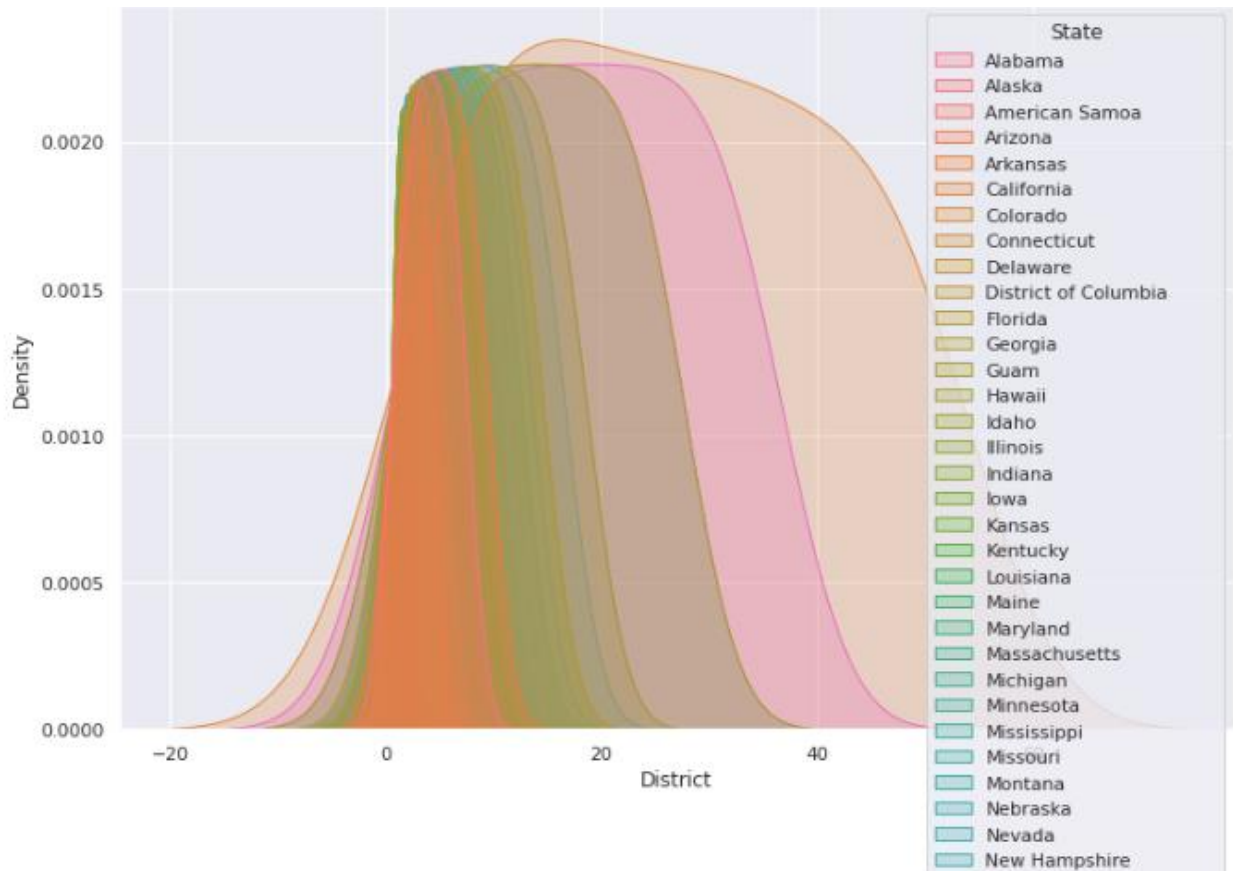
Combination Chart

A combination chart can be used here too. It is used to demonstrate the same relationship as mentioned earlier.

Z-test and T-test

These tests assess whether the averages of the two groups are statistically different from each other. These tests are preferred when the averages of a numerical variable for two categories of a categorical variable are compared.

```
# x = <NUMERIC VARIABLE>, hue = <CATEGORICAL VARIABLE>
plt.figure(figsize=(12,8))
sns.kdeplot(data=df,x='District',hue='State',fill=True)
```



Bivariate Correlations

Correlations measure how variables or rank orders are related. Before calculating a correlation coefficient, screen your data for outliers (which can cause misleading results) and evidence of a linear relationship. **Pearson's** correlation coefficient tells us about the linear association. Two variables can be perfectly related, but if the relationship is not linear, Pearson's correlation coefficient is not an appropriate statistic for measuring their association. In the upcoming exercise, we have used “spearman’s method”.

You can learn more about bivariate analysis [Here](#).

We will practice these concepts in the upcoming exercise.

Exercise 2.02: EDA – Bivariate Analysis

In this exercise, the bivariate analysis will be done and the analysis will be done through the plots. Load the dataset, and import the required visualization libraries and tools to see the relationship between major categories of the two variables.

First, perform the steps mentioned in Exercise 2.01 to cover the major aspects of exploratory data analysis techniques:

Then perform bivariate analysis between two variables as shown below:

1. First, we will perform the GroupBy operation on the continuous variables. Groupby allows us to split our data into separate groups to perform computations for better analysis.
2. Get a global view of all continuous variables with respect to one categorical variable as shown below:

```
|: # by=
df.groupby(by='State').agg('mean')
```

```
|: df.groupby(by='State').agg('mean')[['District']]
```

```
|: # x = , hue =
plt.figure(figsize=(12,8))
sns.kdeplot(data=df,x='District',hue='State',fill=True)
```

```
|: plt.figure(figsize=(12,8))
sns.kdeplot(data=df,x='District',hue='Name',fill=True)
```

3. Then perform correlation.
4. Deal with the analysis of two variables and interpret the relationship between them as shown below:

```
: num_features.corr()
```

```
: sns.set(rc={'figure.figsize':(30,10)})
sns.set_context("talk", font_scale=0.7)
```

```
: sns.heatmap(df.iloc[:,1:].corr(method='spearman'), cmap='rainbow_r', annot=True)
```

```
: df.drop("State", axis=1).apply(lambda x: x.corr(df.State,method='spearman'))
```

Final output will be as follows:

```

Name      -0.039382
District  -0.114813
Party      0.151540
Room       0.028012
Phone      0.054027
Positions  0.091186
Assignment -0.004251
dtype: float64
```

Multivariate Analysis

In multivariate analysis, more than 2 variables are analyzed. It is a tremendously hard task for the human brain to visualize a relationship among 4 variables in a graph. This type of analysis is mostly used for studying more complex sets of data.

Types

- Cluster analysis
- Factor Analysis
- Multiple Regression Analysis
- Principal Component Analysis

Many different ways exist to perform multivariate analysis. But it depends upon the type of data being used.

Cluster Analysis

Cluster analysis is useful for finding groups of related objects in a data set. It tries to classify the objects in such a way that the similarity between two objects from the same group is maximum and minimum otherwise. A well-known example of clustering is in marketing, where customers are grouped into segments based on their buying patterns, demographics, and preferences.

Correspondence Analysis

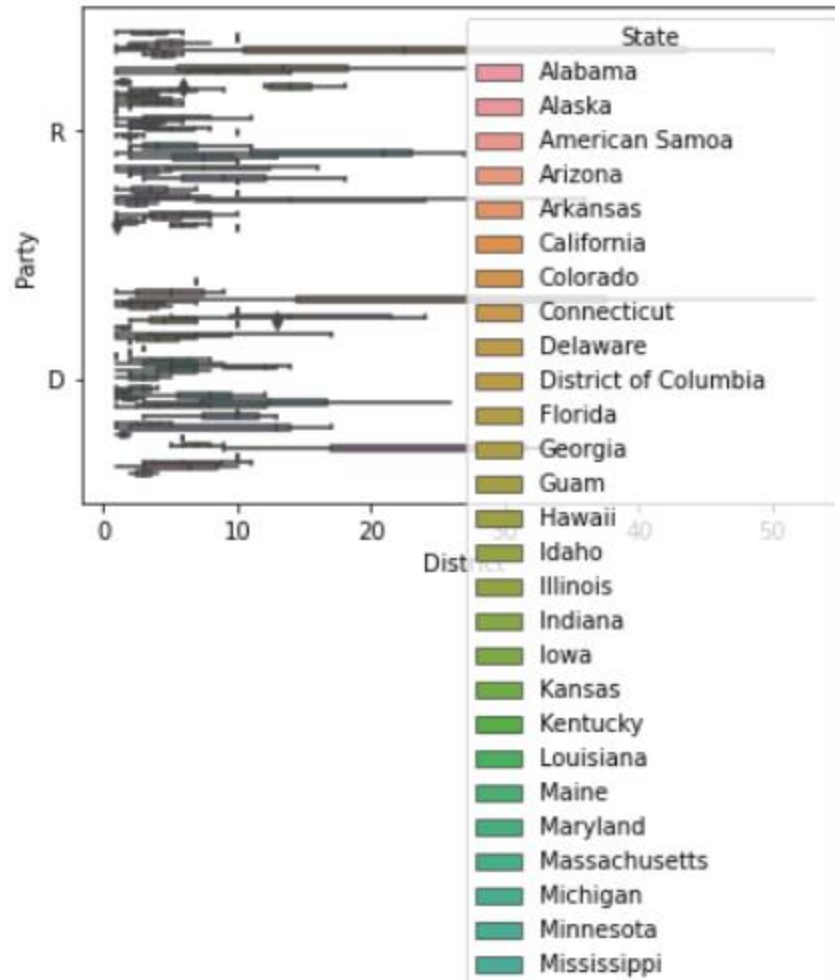
Correspondence Analysis using the data from a contingency table shows relative relationships between and among two different groups of variables. A contingency table is a 2D table with rows and columns as groups of variables.

Principal Component Analysis

Principal Components Analysis (**PCA**) is a linear method for transforming a large dimensional data set into a smaller, standardized dimensionality that is more easily studied and understood. Here we are converting our original variables into a new set of variables called principal components that represent the most important dimensions in a dataset.

Some outputs are the following:

```
sns.boxplot(data=df, x='District', y='Party', hue='State')
```



Single Index:

Categorical value is compared with all continuous values with pivot tables. The following syntax will be followed:

```
# Single Index -
table = pd.pivot_table(data=df, index=['State'])
table
```

Multiple Index

Multiple values are compared with all continuous values with pivot tables. The following syntax will be followed:

```
# Multiple values concerning all continuous values in the dataset
table = pd.pivot_table(df, index=['Room', 'District'])
table
```

Aggregates

Aggregates on specific features will be explored in the upcoming exercise. It involves the following code snippet:

```
# Aggregate on specific features with values parameter
table = pd.pivot_table(df, index=['Room', 'State'], dropna=False)
table
```

```
table = df.pivot_table(index=['Room', 'State'],
                        columns='Assignment',
                        aggfunc='size',
                        fill_value=0,)
table
```

Exercise 2.03: Multivariate analysis

In this exercise, you will practice the concepts related to the relationships between more than two variables. Load the dataset, import the visualization libraries and perform the analysis between more than two variables:

First, perform all the steps for loading data and for exploring the data covered in Exercises 2.01 and 2.02. Then perform the following steps to work on the multivariate analysis of the data.

1. Display the list of numerical features as shown below:

```
] : num_list
    #numeric/numeric/categorical or numeric/categorical/categorical
```

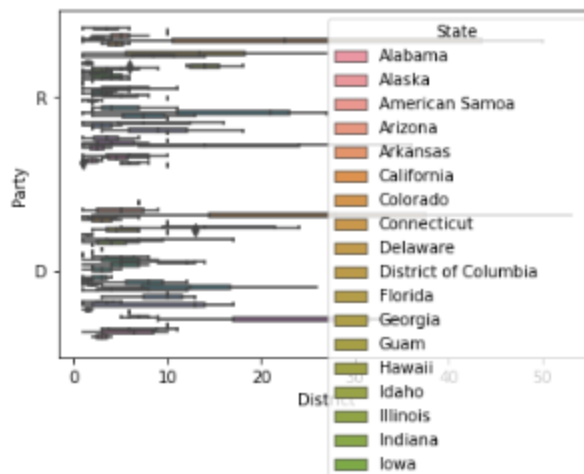
2. Plot box plots as follows:

```
] : sns.boxplot(data=df, x='District', y='Party', hue='State')
```

```
] : plt.style.use('ggplot')
```

Output will look like this figure:


<matplotlib.axes._subplots.AxesSubplot at 0x7f6fe8cabf50>



3. Compare the categorical variable with all continuous variables with pivo tables as shown below:

```
] : table = pd.pivot_table(data=df, index=['State'])  
table
```

Output will be similar to this output:

	District
state	
Alabama	4.000000
Alaska	10.000000
American Samoa	10.000000
Arizona	5.000000
Arkansas	2.500000
California	26.433962
Colorado	4.000000
Connecticut	3.000000
Delaware	10.000000
District of Columbia	10.000000
Florida	14.000000
Georgia	7.500000
Guam	10.000000
Hawaii	1.500000
Idaho	1.500000
Illinois	9.500000
Indiana	5.000000

- Then explore the multiple values with respect to all continuous values in dataset as shown below:

```
In [ ]: table = pd.pivot_table(df, index=['Room', 'District'])
table
```

```
In [ ]: table = pd.pivot_table(df, index=['Room', 'District', 'State'])
table
```

Your output will look like this figure:

```

/usr/local/lib/python3.7/dist-packages/pandas/core/ap
return self._try_aggregate_string_function(obj, f, '

```

Room	District	State
1004 LHOB	8	Ohio
1005 LHOB	8	Indiana
1007 LHOB	8	Wisconsin
1009 LHOB	17	Illinois
1011 LHOB	8	Texas
...
512 CHOB	22	New York
513 CHOB	17	California
514 CHOB	8	Pennsylvania
515 CHOB	8	Illinois
516 CHOB	16	Pennsylvania

441 rows x 0 columns

5. Then work on the aggregates on specific features with values parameters as shown below:

```

]: table = pd.pivot_table(df, index=['District', 'State'], dropna=False)
table

```

```

]: table = df.pivot_table(index=['District', 'State'],
                           columns='Assignment',
                           aggfunc='size',
                           fill_value=0,)
table

```

```

]: table = df.pivot_table(index=['District', 'State'],
                           columns='Assignment',
                           aggfunc='var',
                           fill_value=0,)
table

```

Final pivot table with aggfunc = 'var' will have the following output:

```
/usr/local/lib/python3.7/dist-packages/pandas/core/apply.py:111:
return self._try_aggregate_string_function(obj, f,
```

District	State
1	Alabama
	Massachusetts
	Ohio
	Pennsylvania
	Tennessee
	Utah
2	Massachusetts
	Colorado
	Mississippi
	Oregon
3	Utah
	Mississippi
	Arizona

Automated EDA Tooling

EDA describes the processes of detecting outliers, and missing values, converting categorical variables, and determining the skewness of the dataset. It can be tedious at times to comprehend your data and use it to build models, but with some helpful tools, you can quickly gain an understanding of your database and build better models.

In the upcoming exercise, we will be using “**sweetviz**” for the same purpose.

```
#Installing the library
!pip install dataprep

#Importing
from dataprep.eda import create_report
#Creating report
create_report(df)

!pip install skimpy
from skimpy import skim
skim(df)
```

```
#Installing the library
!pip install sweetviz
#Importing the library
```

```
import sweetviz as sv
report = sv.analyze(df)
report.show_html()

# Splitting data set into training and testing set
training_data = df.sample(frac=0.8, random_state=25)
testing_data = df.drop(training_data.index)

#Applying compare function
report2 = sv.compare([training_data, "TRAINING SET"], [testing_data, "TESTING SET"])
report2.show_html()
```

Finally, you will get the following output:

Done! Use 'show' commands to display/save. [100%] 00:01 -> (00:00 left)
Report SWEETVIZ_REPORT.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report IS saved in your

Exercise 2.04: Automated EDA Tooling

In this exercise, you will practice the concepts related to some tools for EDA. Follow the same steps you learned before. Data Preprocessing is necessary. Load the dataset, import the visualization libraries, and perform the task.

Perform the following steps to practice the Automated EDA tooling techniques.

1. Open the collab notebook and import necessary libraries.
2. Load the dataset and perform other steps mentioned above.

Following output will be produced:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 441 entries, 0 to 440
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Name                   441 non-null   object
1   State                  441 non-null   object
2   District               441 non-null   int64
3   Party                  441 non-null   object
4   Room                   441 non-null   object
5   Phone                  441 non-null   object
6   House Positions        10 non-null    object
7   Committee Assignment   55 non-null    object
dtypes: int64(1), object(7)
memory usage: 27.7+ KB
```

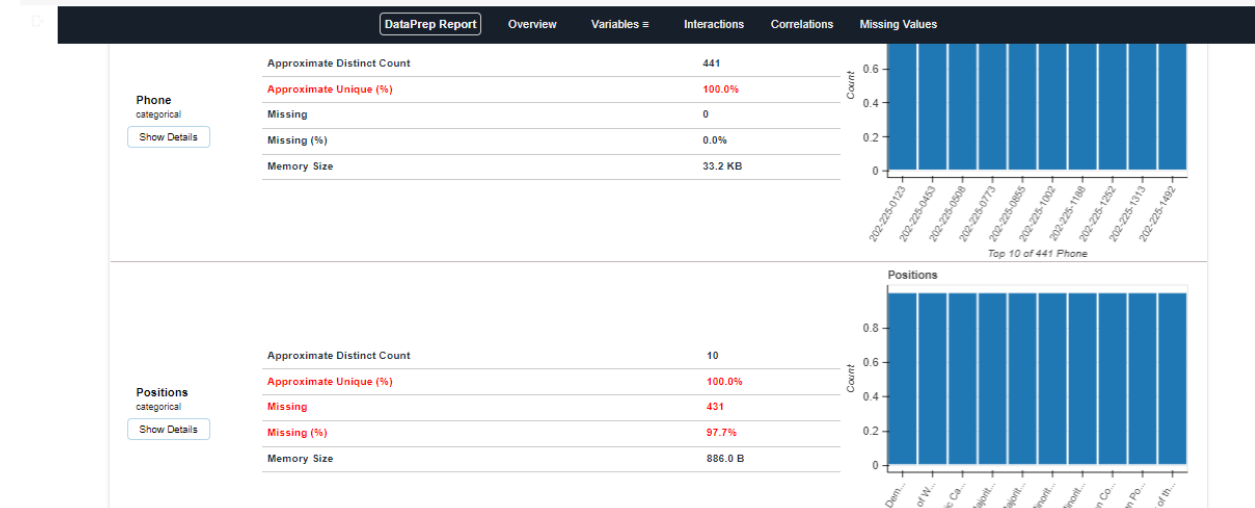
3. To start with it, install "dataprep" library as shown below:

```
[12]
!pip install dataprep
```

4. Create report as shown below:

```
#Importing
from dataprep.eda import create_report
#Creating report
create_report(df)
```

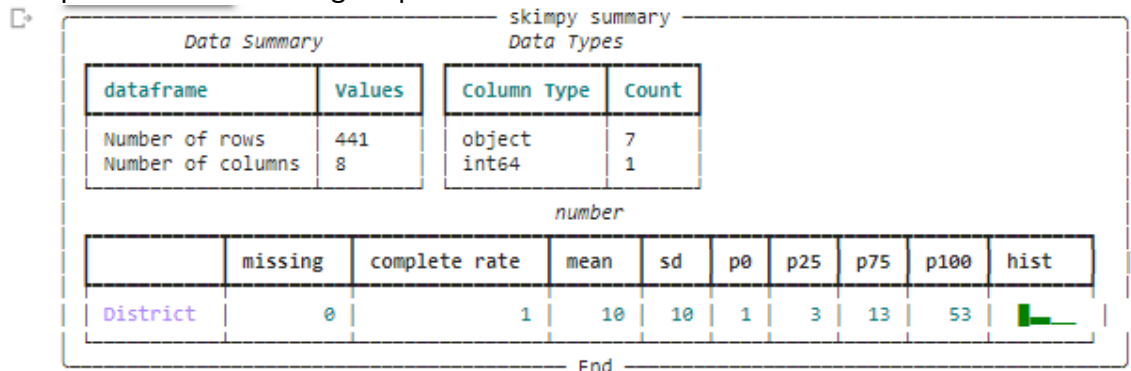
Report will look like this figure:



5. Install skimpy as shown below:

```
[ ]:
!pip install skimpy
from skimpy import skim
skim(df)
```

It will produce the following output:



6. Install sweetviz and show html as shown below:

```
[15] #Installing the library
      !pip install sweetviz
      #Importing the library
      import sweetviz as sv
      report = sv.analyze(df)
      report.show_html()
```

Activity 2.01: Loading the “City Land Mass - Miami” Dataset and performing Univariate and Bivariate Analysis on it.

In this activity, you will be checking the working on another dataset. This dataset contains the records of Landmass from Miami. You have to perform Univariate and bivariate analyses on the data to test the skill. Kindly apply the above-mentioned points covered in the exercises to implement them.

The steps are as follows:

1. Open the collab notebook and load the dataset mentioned above. It is shown here:

```
] : df = pd.read_csv('https://raw.githubusercontent.com/fenago/datawrangling/main/Chapter%202/City_Land_Mass.csv')
    df.head(5)
    df.info()
```

2. Import the required libraries. Visualization libraries will be imported for visuals.
3. Work on the Data Preprocessing to clean the dataset if required.
4. You need to perform the univariate analysis of Continuous variables.
5. Store columns with specific data types as shown below:

```
# store columns with specific data type
integer_columns = df.select_dtypes(include=['int64']).columns
float_columns = df.select_dtypes(include=['float64']).columns
object_columns = df.select_dtypes(include=['object']).columns
```

```
# display columns
print('\nint64 columns:\n',integer_columns)
print('\nfloat64 columns:\n',float_columns)
print('\nobject columns:\n',object_columns)

# Remove the .columns and it saves it as a dataframe
num_features = df.select_dtypes(exclude=['object'])
cat_features = df.select_dtypes(include=['object'])
type(num_features)
type(cat_features)
cat_features
```

6. Plot the histograms to perform a univariate analysis.
7. Perform Bivariate Analysis of Categorical variables vs Continuous Variables.

8. Get a global view of all continuous variables with respect to one categorical variable.
9. Perform the “**groupby()**” operation to split the data into separate columns to perform computations for better analysis

```
df.groupby(by='CLNDNAME').agg('mean')[['FID']]
```

Note: Firstly try to perform it yourself. Then verify it from the solution provided.

In order to check the solution, you can access it from here:

https://github.com/fenago/datawrangling/blob/main/Chapter%202/Activity_2.01_Land_Mass_Miami.ipynb

Conclusion

In this chapter, we learned how to perform rigorous analysis on the dataset. Practicing EDA, we have seen Univariate, bivariate, and multivariate analyses of the data. Moreover, we have also learned about automated EDA tooling. Extensive methods were covered in this chapter to analyze the data.

It was observed that different types of relationships exist between the different variables. Moreover, different operations need to be performed between them depending upon their types. Similarly, the graphs were also dependent upon the type of data. It can be a numerical or a categorical variable, but it must be analyzed before starting the process. Data cleaning and Data preprocessing should also be done before starting the procedure.

In the next chapter, we will cover the “**Feature Engineering**” techniques, including feature selection, Feature Importance, Feature Reengineering, and binning.