# 6_python_II

December 21, 2024

## 1 Data Manipulation and Visualization with Python

In this notebook, we will cover data manipulation and visualization using Python. We will use the pandas library for data manipulation and the matplotlib and seaborn libraries for data visualization.

[1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set seaborn style for plots
sns.set(style="whitegrid")
```

## 2 Reading in a Dataset and Gathering Basic Information

Let's start by reading in a CSV file and gathering basic information about the dataset.

[2]:
```
# Read in the CSV file
df = pd.read_csv('mpg.csv')

# Display the first few rows of the dataframe
df.head()
```

### 2.1 Basic Information about the DataFrame

Here are some good ways to get basic information about a dataframe in Python:

- `head()`: Displays the first few rows of the dataframe.
- `tail()`: Displays the last few rows of the dataframe.
- `shape`: Returns the dimensions of the dataframe (number of rows and columns).
- `columns`: Returns the column names of the dataframe.
- `info()`: Displays the structure of the dataframe, including data types and a preview of the data.
- `describe()`: Provides summary statistics for each column in the dataframe.

[3]:
```
# Display the first few rows of the dataframe
df.head()
```

```
# Get the dimensions of the dataframe
df.shape

# Get the column names of the dataframe
df.columns

# Display the structure of the dataframe
df.info()

# Provide summary statistics for each column in the dataframe
df.describe()
```

## 3 Data Manipulation with Pandas

We will now cover some basic data manipulation techniques using the pandas library.

[4]:
```
# Create a sample dataframe
data = {
    'name': ['Andie', 'Bridger', 'Scott'],
    'gender': ['Female', 'non-binary', 'Male'],
    'male': [False, False, True],
    'income_cat': ['middle', 'poor', 'rich'],
    'park_dist': [1.0, 0.5, 0.1]
}
df_sample = pd.DataFrame(data)
df_sample
```

### 3.1 `assign()`

The `assign()` method can be used to add new columns or modify existing ones.

[5]:
```
# Add 1 mile to park_dist
df_sample = df_sample.assign(park_dist=lambda x: x.park_dist + 1)
df_sample
```

### 3.2 `np.where()`

The `np.where()` function can be used to conditionally modify values in a dataframe.

[6]:
```
# Correct park_dist for non-male individuals
df_sample['park_dist_correct'] = np.where(df_sample['male'] == False,␣
 ↪df_sample['park_dist'] - 0.25, df_sample['park_dist'])
df_sample
```

### 3.3 `filter()`

Filtering rows in a dataframe can be done using boolean indexing.

```
[7]: # Filter rows where pollution_level is 'Low'
df_env_data = pd.DataFrame({
    'ecosystem': ['Forest', 'Desert', 'Wetland', 'Grassland', 'Urban'],
    'species_richness': [120, 45, 80, 60, 30],
    'pollution_level': ['Low', 'High', 'Medium', 'Low', 'High']
})
low_pollution_data = df_env_data[df_env_data['pollution_level'] == 'Low']
low_pollution_data
```

### 3.4 dropna()

Dropping rows with missing values can be done using the `dropna()` method.

```
[8]: # Drop rows with missing values in the 'ecosystem' column
df_env_data_na = pd.DataFrame({
    'ecosystem': ['Forest', 'Desert', 'Wetland', np.nan, 'Urban'],
    'species_richness': [120, 45, 80, 60, 30],
    'pollution_level': ['Low', 'High', 'Medium', 'Low', 'High']
})
df_env_data_clean = df_env_data_na.dropna(subset=['ecosystem'])
df_env_data_clean
```

### 3.5 select()

Selecting specific columns can be done using the `loc` or `iloc` methods.

```
[9]: # Select only the 'ecosystem' and 'pollution_level' columns
pollution_data = df_env_data[['ecosystem', 'pollution_level']]
pollution_data
```

### 3.6 groupby()

Grouping data and calculating aggregate statistics can be done using the `groupby()` method.

```
[10]: # Group by 'ecosystem' and calculate the mean species richness
df_env_long = pd.DataFrame({
    'ecosystem': ['Forest', 'Desert', 'Wetland', 'Grassland', 'Urban',␣
  ↪'Forest', 'Desert', 'Wetland', 'Grassland', 'Urban'],
    'species_richness': [120, 45, 80, 60, 30, 110, 50, 85, 65, 35],
    'pollution_level': ['Low', 'High', 'Medium', 'Low', 'High', 'Low', 'High',␣
  ↪'Medium', 'Low', 'High']
})
df_env_grouped = df_env_long.groupby('ecosystem').species_richness.mean().
  ↪reset_index()
df_env_grouped
```

### 3.7 agg()

The `agg()` method can be used to apply multiple aggregation functions to grouped data.

[11]:
```
# Group by 'ecosystem' and calculate the mean and total species richness
summary_table = df_env_long.groupby('ecosystem').species_richness.agg(['mean',␣
 ↪'sum']).reset_index()
summary_table.columns = ['ecosystem', 'mean_species_richness',␣
 ↪'total_species_richness']
summary_table
```

## 4 Basic Data Visualization

Let's create some basic plots to visualize the data.

[12]:
```
# Histogram
plt.figure(figsize=(10, 6))
sns.histplot(df['hwy'], bins=10, kde=False, color='black')
plt.title('Distribution of Highway Miles per Gallon')
plt.xlabel('Highway Miles per Gallon')
plt.ylabel('Count')
plt.show()
```

[13]:
```
# Box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='cyl', y='hwy', data=df)
plt.title('Highway MPG Distribution by Cylinder Count')
plt.xlabel('Number of Cylinders')
plt.ylabel('Highway Miles per Gallon')
plt.show()
```

[14]:
```
# Bar chart
plt.figure(figsize=(10, 6))
sns.countplot(x='manufacturer', data=df, color='black')
plt.title('Number of Observations by Manufacturer')
plt.xlabel('Manufacturer')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```

[15]:
```
# Scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='displ', y='hwy', data=df)
plt.title('Engine Displacement vs Highway MPG')
plt.xlabel('Engine Displacement (litres)')
plt.ylabel('Highway Miles per Gallon')
plt.show()
```

[16]:
```
# Scatter plot with color grouping
plt.figure(figsize=(10, 6))
sns.scatterplot(x='displ', y='hwy', hue='manufacturer', data=df)
```

```
plt.title('Engine Displacement vs Highway MPG by Company')
plt.xlabel('Engine Displacement (litres)')
plt.ylabel('Highway Miles per Gallon')
plt.legend(title='Company')
plt.show()
```

[17]:
```
# Facet plot
g = sns.FacetGrid(df, col='manufacturer', col_wrap=4, height=4)
g.map(sns.scatterplot, 'displ', 'hwy')
g.set_axis_labels('Engine Displacement (litres)', 'Highway Miles per Gallon')
g.fig.suptitle('Engine Displacement vs Highway MPG by Manufacturer', y=1.03)
plt.show()
```