DATA, MSML, BIOI 602 Principles of Data Science

Data Exploration

Lecture - 7

[Slides from Fardina Fathmiul Alam]

Why have a lecture on data exploration?

When we start our process, **all we have is a CSV, with no idea what we have**.

- There's generally **no label and no guide**.
- It's **useful to have a process** for exploring it.

Exploratory data analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to

- **discover** patterns
- **spot** anomalies
- **test** hypothesis and
- **to check assumptions** with the help of summary statistics and graphical representations.

Exploratory data analysis (EDA)

It is a good practice to understand the data first and try to gather as many insights from it.

EDA is all about making sense of data in hand, before getting them dirty with it.

Before making inferences from data it is essential to examine all your variables**.**

Why?

To listen to the data:

- to catch mistakes

- to see patterns in the data

- to find violations of statistical assumptions

- to generate hypotheses

…and because if you don’t, you will have trouble later

Exploratory data analysis (EDA)

EDA provides a structured process for approaching this scenario. Involves a series of steps and techniques to

- systematically explore the data,
- uncover hidden patterns, and
- gain insights.

This process **ensures that you don't miss critical information and can make informed decisions** about subsequent analyses or modeling.

EDA also emphasizes the importance of data visualization

INTERNATIONAL MERCANTILE MARINE LINES

ALL OUR STEAMERS HAVE WIRELESS TELEGRAPH AND SUBMARINE SIGNALS

The Largest Steamers in the World

New 45,000 Tons Each

OLYMPIC Sail from NEW YORK **April 13** 3 P. M.
MAY 4, 25, JUNE 15, JULY 6.

TITANIC **APRIL 20** 12 Noon
May 11, June 1
June 22, July 13

FRENCH A LA CARTE RESTAURANT, TURKISH & ELECTRIC BATHS,
SWIMMING POOL, FOUR ELEVATORS, GYMNASIUM,
VERANDAH CAFE, PALM COURT, SQUASH RACQUET COURT.

NEARBY SAILINGS

Baltic..... April 11, 12:00 Noon
Vaderland.... April 13, 10:00 A. M.
Minnewaska.. April 14.
Cedric..... April 18, 12:00 Noon

AMERICAN Pier 62, N. R. 9:30 A. M.

Plymouth—Cherbourg—Southampton
St. Paul..... May 4—St. Louis.....May 11

ATLANTIC TRANSPORT

New York—London Direct, Pier 58, N.R.
Minnewaska.....Apr. 18 | Minnehaha..May 11
Minnetonka May 4, 10:30 AM | Minnewaska..May 18

RED STAR Pier 61, N. R. 10 A. M.

London, Paris via Dover—Antwerp
Vaderland.....Apr. 12 | Kroonland.....Apr. 27
Lapland..... ...Apr 19 | Finland.....May 4

WHITE STAR Piers 59 & 60, N. R.

Plymouth—Cherbourg—Southampton
Olympic (new) Ap. 13, 3 P.M. | Olympic (new)...May 4
Titanic (new) Ap. 20, 12 N | Titanic (new)...May 11

New York—Queenstown—Liverpool
Baltic....Apr. 11, noon | Celtic .....Apr. 25
Cedric....Apr. 18, noon | Adriatic .....May 2

FROM BOSTON To The MEDITERRANEAN
CANOPIC, Ap. 27, 3 P.M. CRETIC, May 18, 11 AM

CANADIAN SERVICE.

LARGEST STEAMERS FROM CANADA

MONTREAL VIA QUEBEC TO LIVERPOOL.

*CANADA	May 4, June 1,	June 29
MEGANTIC	May 11, June 8,	July 6
*TEUTONIC	May 18, June 15,	July 13
LAURENTIC	May 25, June 22,	July 20

*Only One Class of Cabin (II) Passengers.

PASSENGER DEPARTMENT, 9 BROADWAY, NEW YORK.

Example

Today we will use “Titanic Dataset” for EDA

- **real-world relevance and diverse features**

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

https://colab.research.google.com/drive/11JjW055RQ-poNCXTlzPiUxgs9hMi5Xns?usp=sharing

Asking the Right Questions

Identifying the Key Variable of Interest: Generally, there will be one column you care about the most: Revenue, engagement, survival, etc. Focus on how the other variables relate to that.

Understanding Relationships, Considering Dependencies:

You'll also need your common sense! Think about **what columns might be related**, what might contribute to what, what might dependent or independent.

Start your analysis with a list of questions, but don't be afraid to change them!

REMEMBER:

- Asking questions and conducting EDA reveal insights, patterns, and trends, aiding analysis and decision-making.
- Use suitable visualizations and statistical tests to support findings.

- What is the demographic breakdown of passengers?
- What was the overall survival rate of passengers on the Titanic?
- Did age have an impact on survival rates?
- Were passengers in higher classes more likely to survive?
- What was the distribution of fares paid by passengers?
- Were families more likely to survive together?
- Did the point of embarkation impact survival rates?
- How did gender influence survival rates?
- Were there any patterns in passenger names or titles that might be interesting?
- What is the correlation between various features (e.g., age, fare, class)?
- What was the distribution of fares paid by passengers?
- What was the survival rate of different passenger classes broken down by gender?

REMEMBER:

- Asking questions and conducting EDA : Start your analysis with a list of questions based on your initial hypotheses and domain knowledge.
- Refine your questions to gain a deeper understanding of the factors, if needed.

- Do first-class passengers have a higher survival rate compared to third-class passengers?"
- Is there a significant difference in survival rates between males and females?"
- How does age impact the likelihood of survival?
- Are passengers who embarked from different ports more likely to survive?"

Focus on: you're not only describing the data but also seeking meaningful explanations for observed patterns

EDA Checklist

An EDA checklist If a checklist is good enough for pilots to use every flight, it's good enough for data scientists to use with every dataset.

- **Define Objective:** What question(s) are you trying to solve (or prove wrong)? Clearly state the problem or question you aim to address with the data. Having a clear objective guides the analysis and ensures focus.
- **Understand Data Types:** What kind of data do you have and how do you treat different types? Identify the types of data present in the dataset (e.g., numerical, categorical) and understand how to handle each type appropriately.
- **Handle Missing Data:** What's missing from the data and how do you deal with it (e.g., imputation, deletion)? [NEXT CLASS TOPIC]
- **Detect Outliers:** Where are the outliers and why should you care about them? Outliers can skew results and need to be assessed for their relevance and potential handling.
- **Feature Engineering:** How can you add, change or remove features to get more out of your data?

More questions we can explore

- Were there any outliers in the dataset, especially regarding fares or ages?
- Did passengers with missing values for certain features have different survival rates?

A Starting Point

- Once your dataframe is loaded, you'll be able to look at each column and it's datatype
  - `df.columns` will list all the columns
  - `df.dtypes` will tell you the datatypes of each column
- Oftentimes, data will come with a data dictionary explaining each column; more often it won't. You'll generally need to ask around to find out what's what. If you can't figure it out, work from the units and the name of the column.

Basic Data Manipulation

Some data might be in formats that are hard to work with. Feel free to add your own columns:

- “date” might become “date_in_days_since_zero”
- Survey results, such as “strong agree”, “agree”, “neutral”, “disagree”, “strong disagree”, might become 2, 1, 0, -1, -2

Simply use the apply function. Similarly, feel free to change units:

- Days can become seconds
- Pounds can become grams

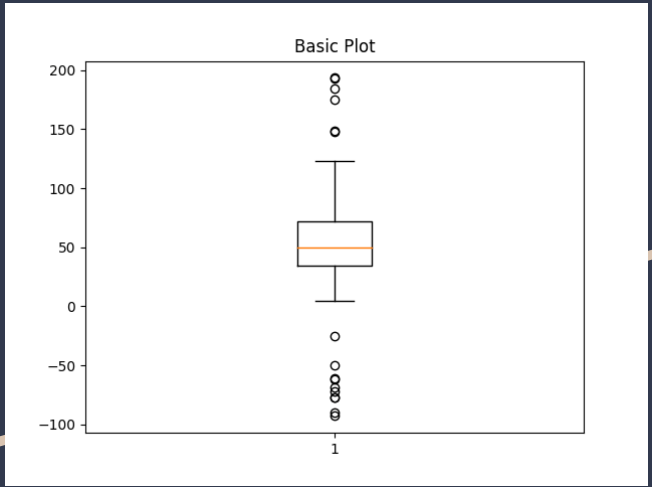
etc.

Learning About Individual Columns

- We can use the **describe** method to generate summary statistics for individual columns in the dataset as part of our exploratory data analysis (EDA).
- Example: df[“column”].describe() will give you:

```
count 69.000000
mean 112.696254
std 75.825020
min 23.590017
25% 69.131239
50% 95.814021
75% 143.869780
max 532.230000
Name: damage, dtype: float64
```

Learning About Individual Columns

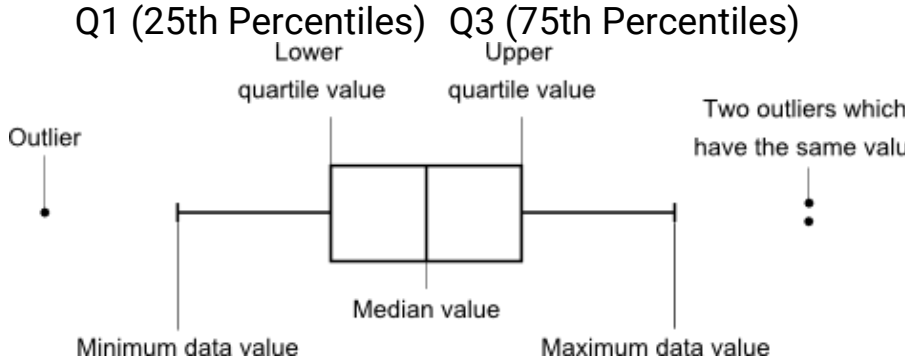


- We can also create box and whisker plots to visualize the distribution of numeric columns.
- Example: You can also create a box and whisker chart:

```
fig, ax = plt.subplots()

ax.boxplot(x.values())
```

Which might give you something like this:



Distributions

Next, you might want to know about how variables are distributed.

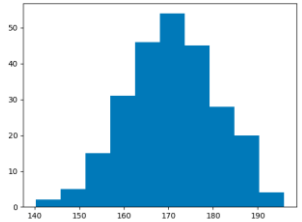
Example: For categorical variables, often df[“column”].value_counts() is enough:

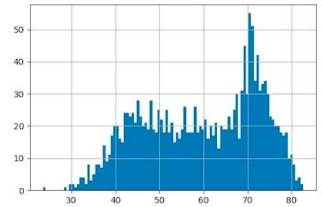
```
New Orleans 9
Seattle 5
Boston 4
San Francisco 3
Long Beach 3
Memphis 3
Virginia Beach 3
Detroit 3
Dallas 2
Orlando 2
```

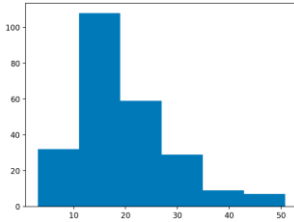
Distributions

If you have a more continuous variable, you can visualize the distribution of continuous or numeric variables such as "Age" or "Fare.

Ex: df[“column”].hist() will usually work:







Each histogram can tell a different story!

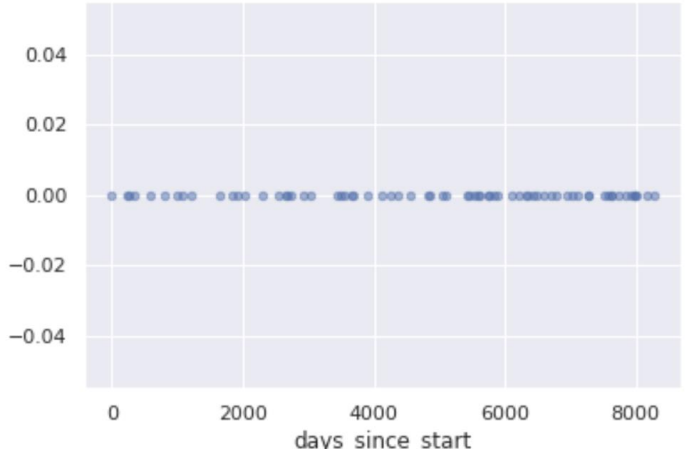
Event Frequency

"Event frequency" is about how often something happens; typically used for visualizing events over a continuous variable like time

- To visualize it, we can create a scatter plot with dots representing events.
- If events overlap, we can make the dots slightly see-through using alpha parameter for clarity.
- helps to understand how often different events occur.

For even frequency, you can create a one dimensional scatter plot. Turn on alpha if the events overlap!

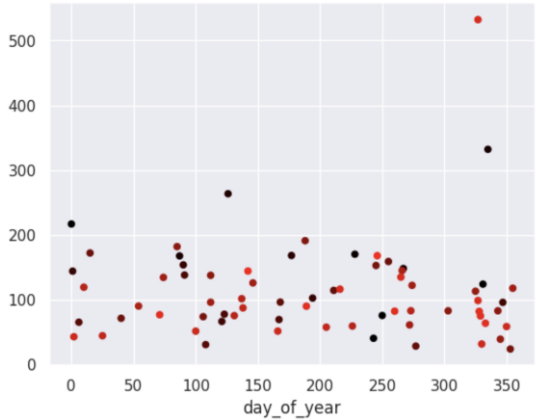
```
df["zeroes"] = 0
df.plot.scatter("days", "zeroes", alpha=.5)
```



Checking Seasonality

With all temporal data, you should check for seasonal patterns (yearly, weekly, monthly). You can do this with code like this:

```
df["color"] = (int(df["day_of_year"] / 365), 0, 0)
df["day_of_year"] = df["days_since_start"] % 365
df.plot.scatter("day_of_year", "x", c=color)
```



In this graph, each shade of red corresponds to one specific year.

Variable Relationships

Visualize relationships between variables.

- Use graphs, charts, and plots to explore and understand how two or more variables are related or correlated.
- helps to see patterns, trends, and associations between variables in a graphical and intuitive way.

Variable Relationships

You might want to know if two variables have a relationship!

For starters, you can use df.corr() to look at all the correlations between the variables.

Variable Relationships

The df.corr() method in pandas calculates the pairwise correlation coefficients between all numerical columns in a DataFrame.

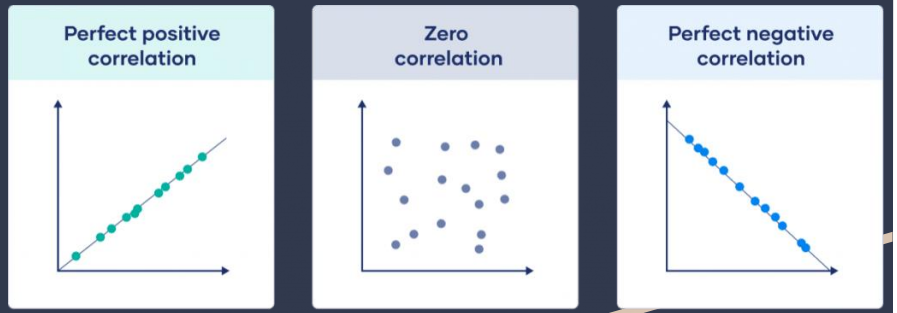
By default, it calculates Pearson correlation coefficients, which measure the linear relationship between two variables.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Pearson correlation coefficient ranges from -1 to 1, where:

- 1 indicates a perfect positive linear relationship (perfectly correlated),
- -1 indicates a perfect negative linear relationship (perfectly anti-correlated)
- 0 indicates no linear relationship.

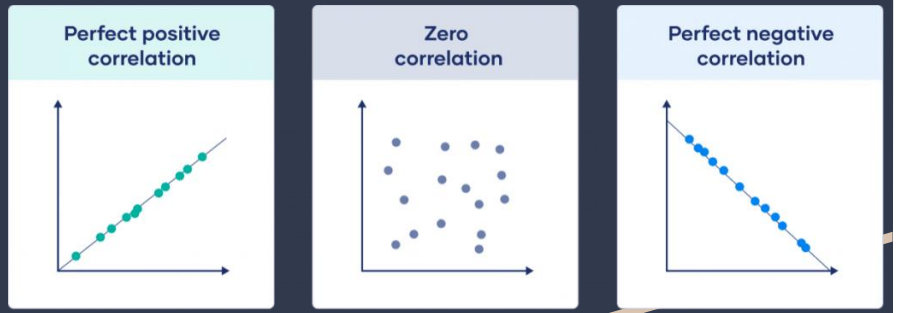
The Pearson correlation coefficient (r)



- **The Pearson correlation coefficient (r)** is the most common way of measuring a linear correlation.
- It is a number between –1 and 1 that measures the strength and direction of the relationship between two variables.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Variable Relationships



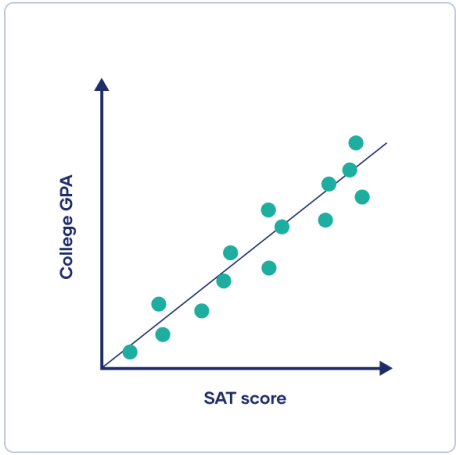
Remember, Correlation coefficient represents a number (summarize data) between -1 and 1 that tells you the strength and direction of a relationship between variables.

- it reflects how similar the measurements of two or more variables are across a dataset.

Correlation coefficient value	Correlation type	Meaning
1	Perfect positive correlation	When one variable changes, the other variables change in the same direction.
0	Zero correlation	There is no relationship between the variables.
-1	Perfect negative correlation	When one variable changes, the other variables change in the opposite direction.

Variable Relationships

Example: You gather a sample of 5,000 college graduates and survey them on their high school SAT scores and college GPAs. You visualize the data to check for a linear pattern:



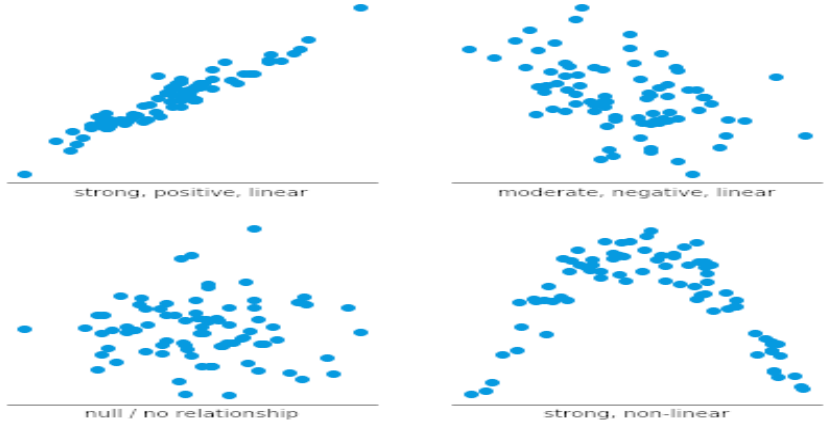
Variable Relationships: Scatter Plot Usage

If you'd like to learn more about a pair of variable's relationships, **use a scatter plot!**

Scatter plots are visual tools for exploring relationships visually

```
df.plot.scatter("col1", "col2")
```

It is the most useful graph for displaying the relationship between two quantitative variables.



Statistical Significance

If you have two datasets with different averages, you aren't allowed to assume they come from different distributions! Oftentimes you'll need to run a ttest:

```
stats.ttest_ind(df[x], df[y])
```

Identifying Errors

Look for:

- **Duplicated values.** Compare the relative length of len(df) and len(df.drop_duplicates())
  - len(df) =the total number of rows in the original DataFrame
  - len(df.drop_duplicates()) = the number of rows after removing duplicates.
  - compare
- **Check for missing values:** sum(df[“column”].isna()) will tell you how many missing values are in a series
- **Check for outliers:** These should show up in your box and whisker plot.
  - Make sure you understand why your outliers are outliers.

Finishing

Once we are done, we should be an expert on our data.

- **Understand all data ranges**
  - helps to identify outliers, anomalies, and extreme values that may require further investigation or treatment.
- **Know what variables relate to each other**
  - identify correlations and dependencies between variables
- **Know where your errors are**
  - handling missing values, correcting data entry mistakes, and addressing outliers.
  - Errors in the data can lead to incorrect analysis and conclusions,
- **Have an idea of what other data you may need**

Additional

Check more example from previous semester:

https://colab.research.google.com/drive/15VaaSgutTEOQWc7HA-LPAOfFX5LzS5hC?usp=sharing

Courtesy: Professor Maksym

Commonly used syntax patterns for df

Applying Functions Directly: You can apply functions directly to a DataFrame column:

df[column_name].function()

df['frequency'].sum()

Commonly used syntax patterns for df

Applying Functions to a Column

You can apply custom functions to a DataFrame column like this:

df['column_name'].apply(custom_function)

```
# Define a custom function to double a value
Def double_value(value):
    return value * 2

# Apply the custom function to the 'values' column and create a new column 'doubled_values'
df['doubled_values'] = df['values'].apply(double_value)

Or use lambda:
df['doubled_values'] = df['values'].apply(lambda x: x**2))
```

Some commonly used syntax patterns for df

Filtering Data

Filtering Rows Based on a Condition

You can filter rows in a DataFrame based on a condition and then apply statistical functions to a specific column:

```
df[df['column_name'] > value]
```

boolean_condition

Q: What about multiple condition?

```
df[df['condition']][column_name].statistics_function()
```

Q: What about multiple condition?

Commonly used syntax patterns for df

Filtering Data

Filtering Rows Based on a Condition

You can filter rows in a DataFrame based on a condition and then apply statistical functions to a specific column:

	Fruit_Name	Quantity_Kg
0	Apple	5
1	Banana	8
2	Orange	10
3	Grapes	3
4	Kiwi	6

df[df['column_name'] > value]

boolean_condition

df['Quantity_Kg'] > 5

	Fruit_Name	Quantity_Kg
1	Banana	8
2	Orange	10
4	Kiwi	6

Q: What about multiple condition?

Commonly used syntax patterns for df

Filtering Data

Filtering Rows Based on a Condition

You can filter rows in a DataFrame based on a condition and then apply statistical functions to a specific column:

	Student_ID	Test_Score
0	1	85
1	2	92
2	3	78
3	4	95
4	5	88

df[df['condition']][column_name].statistics_function()

```
df[df['Test_Score'] > 90]['Test_Score'].mean()
```

Commonly used syntax patterns for df

Grouping and Aggregating Data

You can group data by a condition and calculate statistics within each group:

df.groupby('condition')[column_name].statistics_function()

	Product_Name	Category	Price_USD
0	Apple	Fruit	0.50
1	Banana	Fruit	0.30
2	Orange	Fruit	0.60
3	Laptop	Electronics	800.00
4	Tablet	Electronics	400.00
5	Book	Books	20.00

df.groupby('Category')['Price_USD'].mean()

```
Category
Books          20.0
Electronics   600.0
Fruit           0.47
Name: Price_USD, dtype: float64
```

Commonly used syntax patterns for df

Applying Functions to Create New Columns

You can apply a function to create a new column in the DataFrame:

```
df['new_col_name']= df['other_column_name'].apply(custom_function)
```

Commonly used syntax patterns for df

Applying Functions to Create New Columns

You can apply a function to create a new column in the DataFrame:

```
df['new_col_name']= df['other_column_name'].apply(custom_function)
```

Commonly used syntax patterns for df

In-Place Operations

Perform in-place operations on a column.

In-place operation

```
df['column_name'].function(value, inplace=True)
```

Non-in-place operation (default behavior)

```
df_copy = df['column_name'].function (value, inplace=False)
```

Some commonly used syntax patterns for df

Grouping and Aggregating Data

You can group data by a condition and calculate statistics within each group:

```
df.groupby('condition')[column_name].statistics_function()
```