CoGrammar

Datasets and DataFrames

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: Open Class Questions

Data Science Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: Feedback on Lectures

Lecture Objectives

- Learn how to **read and manipulate data** with **Pandas**

CoGrammar

Datasets

- ★ A dataset is a **structured collection of information relevant to a specific investigation or project**
- ★ In data science, they provide the raw material for analysis and modeling. Understanding different dataset formats ensures you can work with data from various sources (databases, online repositories, etc.).
- ★ With the help of pandas DataFrames, we can effortlessly manipulate data to suit our needs.

DataFrames

- ★ A DataFrame is the way the Pandas library in Python **represents tabular data**. **It's like a powerful spreadsheet within your code.**

- ★ **Rows:** Each row represents a **single observation** or data point (e.g., a person, a product, a transaction).

- ★ **Columns:** Each column represents a **variable or feature** (e.g., height, price, date). Data within a column usually shares the **same data type** (numbers, text, etc.).

Jupyter Notebook

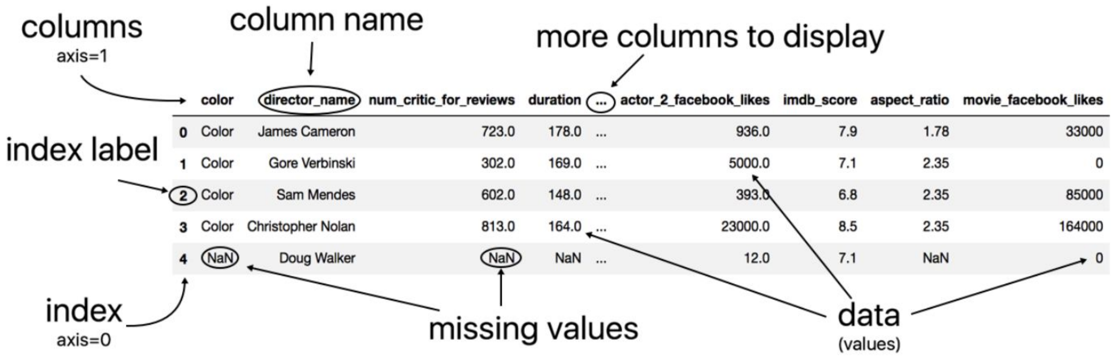
- ★ An **interactive environment** perfect for **data science work**. They let you **combine code, the results of the code (output), and explanatory text (like in a scientific report).**

- ★ This fosters **clear data exploration and storytelling**, all in one place

CoGrammar

Pandas DataFrame

★ The pandas' library documentation defines a DataFrame as a "**two-dimensional, size-mutable, with labelled rows and columns.**"



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

Anatomy of a DataFrame

CoGrammar

Pandas DataFrame

★ Pandas provides functions like **pd.read_csv(), pd.read_excel(), etc.,** to bring your data directly into your coding environment as DataFrames.

★ This is where you start turning your raw data into something easily workable.

```
# url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv'
# df = pd.read_csv(url)

iris = datasets.load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
```

CoGrammar

Exploring Datasets

★ **df.head(), df.tail()**: Peek at the **top** and **bottom** rows for initial understanding.

```
df.head()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Exploring Datasets

★ **df.head(), df.tail()**: Peek at the **top** and **bottom** rows for initial understanding.

```
df.tail()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

CoGrammar

Exploring Datasets

★ **df.shape**: Tells you the **dimensions (rows, columns)** of your data.

```
df.shape
✓ 0.0s

(150, 5)
```

Exploring Datasets

★ **df.info()**: Gives the **data types** of each column, and if columns have missing values.

```
df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

CoGrammar

Exploring Datasets

★ **df.describe()**: Quick **summary statistics for numerical columns**.

```
df.describe()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

CoGrammar

Exploring Datasets

★ All the previous methods **let you quickly grasp the size and structure of your data, helping you plan analysis steps**.

Manipulating Data

★ **Selecting Columns**: You often work with a **subset of features.** Using df[['column1', 'column2']] gets you only specified columns.

```
df.columns
✓ 0.0s

Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'species'],
      dtype='object')
```

```
# Select specific columns
df_selected = df[['species', 'petal length (cm)', 'petal width (cm)']]
✓ 0.0s
```

Manipulating Data

★ **Filtering Rows:** Focus on specific subsets meeting certain conditions, e.g., df[df['age'] > 30].

```
# Filter by flower species
df_setosa = df[df['species'] == 'setosa']
```

✓ 0.0s

CoGrammar

Manipulating Data

★ **Creating New Columns:** Derived features, e.g., calculating BMI from height and weight.

```
# Create a new calculated column
df['petal area (cm^2)'] = df['petal length (cm)'] * df['petal width (cm)']
✓ 0.0s
```

CoGrammar

Manipulating Data

★ **Renaming/Dropping**: Improve clarity or get rid of unneeded data.

```
# Rename a column
df = df.rename(columns={'sepal length (cm)': 'sepal_len'})
```

✓ 0.0s

★ Data manipulation gives you a **highly customized DataFrame focused on your exact analysis needs**.

Built-in Methods

- Pandas offers a toolbox of functions for calculations:
  - **mean()** - Computes the mean for each column.
  - **min()** - Computes the minimum for each column.
  - **max()** - Computes the maximum for each column.
  - **std()** - Computes the standard deviation for each column.
  - **var()** - Computes the variance for each column.
  - **nunique()** - Computes the number of unique values in each column.

- This is the start of understanding the characteristics of your data.

Grouping and Aggregation

★ **df.groupby()**: Divide your data **based on categories** in a column (e.g., group by country).

```
print(df['petal area (cm^2)'].mean())
print(df['species'].nunique())
print(df.groupby('species')['petal length (cm)'].std())
✓ 0.0s

5.794066666666667
3
species
0    0.173664
1    0.469911
2    0.551895
Name: petal length (cm), dtype: float64
```

Grouping and Aggregation

★ **.agg()**: Apply calculations within each group (e.g., average salary per country).

```
df.groupby('species').agg(
    mean_petal_length=('petal length (cm)', 'mean'),
    max_sepal_width=('sepal width (cm)', 'max')
)
```

✓ 0.0s

species	mean_petal_length	max_sepal_width
0	1.462	4.4
1	4.260	3.4
2	5.552	3.8

CoGrammar

Data Visualization

- ★ There is a whole lecture dedicated to this, but libraries like **Matplotlib and Seaborn** make visually exploring data easy. **Plots (scatter plots, histograms, etc.) rapidly uncover relationships and distributions that are less clear from tables of numbers.**

- ★ Some good examples are in the Seaborn Gallery.

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Thank you for joining!