



Foundational Python Track: Introduction to Data Manipulation and Visualization

Yassin Khalifa



University of
Pittsburgh

PittCRC
Center for Research Computing

PittIT

March 2, 2023



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Data structures in Python

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

List	Dictionary
List is a collection of index-values pairs as that of array in c++.	Dictionary is a hashed structure of key-value pairs.
List is created by placing elements in [] separated by commas “,”	Dictionary is created by placing elements in { } as “key”: “value”, each key value pair is separated by commas “,”
The indices of list are integers starting from 0.	The keys of dictionary can be of any data type.
The elements are accessed via indices.	The elements are accessed via keys.
The order of the elements entered are maintained.	There is no guarantee for maintaining order.

What is missing:

- Data structures with more flexible indexing for accessibility using both keys and numeric indices.
- Built-in methods that allow for data manipulation within the data structure.



Series

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

Series is a "dict-like" one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index.

```
# import libraries
>>> import pandas as pd
>>> import numpy as np

# the basic method to create a series is to call
>>> s = pd.Series(data, index=index)
```

data can be many different things:

- a Python dict
- an ndarray
- a scalar value

index is a list of axis labels



Series: Create

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
>>> s
a    0.469112
b   -0.282863
c   -1.509059
d   -1.135632
e    1.212112
dtype: float64
>>> s.index
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
>>> d = {"x": 0.0, "y": 1.0, "z": 2.0}
>>> pd.Series(d)
x    0.0
y    1.0
z    2.0
dtype: float64
```



Series: Indexing and data operations

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> s['a']
0.469112
>>> s[['a', 'c']]
a    0.469112
c   -1.509059
dtype: float64
>>> s[0]
0.469112
>>> s[2:]
c   -1.509059
d   -1.135632
e    1.212112
dtype: float64
>>> s[s < 0]
b   -0.282863
c   -1.509059
d   -1.135632
dtype: float64
```

```
>>> np.power(s, 4)
a    0.000009
b    0.060818
c    0.015160
d    1.335574
e    6.791131
dtype: float64
>>> s.to_numpy()
array([ 0.4691, -0.2829, -1.5091,
       -1.1356,  1.2121])
>>> s = pd.Series(np.random.randn(5),
                  name="something")
0   -0.390516
1    0.644527
2    1.315244
3    0.142816
4   -0.445948
Name: something, dtype: float64
```



DataFrame

Data Handling and Manipulation

Data Structures in pandas

IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object.

```
# the basic method to create a dataframe is to call  
>>> s = pd.DataFrame(data)
```

Like Series, DataFrame accepts many different kinds of input:

- Dict of 1D ndarrays, lists, dicts, or Series
- 2-D numpy.ndarray
- Structured or record ndarray
- A Series
- Another DataFrame



DataFrame: Create

Data Handling and Manipulation
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> d = {"one": pd.Series([1.0, 2.0,
   > 3.0], index=["a", "b", "c"]),
   > "two": pd.Series([1.0, 2.0, 3.0, 4.0],
   > index=["a", "b", "c", "d"])}
>>> df = pd.DataFrame(d)
>>> df
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
>>> pd.DataFrame(d, index=["d", "a",
   > "c"])
   one  two
d  NaN  4.0
a  1.0  1.0
c  3.0  3.0
>>> pd.DataFrame(d, index=["d", "a"],
   > columns=["two", "three"])
```

```
   two  three
d  4.0  NaN
a  1.0  NaN
>>> d = {"one": [1.0, 2.0, 3.0], "two":
   > [3.0, 2.0, 1.0]}
>>> pd.DataFrame(d)
   one  two
0  1.0  3.0
1  2.0  2.0
2  3.0  1.0
>>> pd.DataFrame(d, index=["a", "b",
   > "c"])
   one  two
a  1.0  3.0
b  2.0  2.0
c  3.0  1.0
>>> d = {"one": pd.Series([1.0, 2.0,
   > 3.0], index=["a", "b", "c"]),
   > "two": pd.Series([1.0, 2.0, 3.0, 4.0],
   > index=["a", "b", "c", "d"]})
```



DataFrame: Indexing and selection

Data Handling and Manipulation
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> df = pd.DataFrame(d)
>>> df
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
>>> df["one"]
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
>>> df["three"] = df["one"] * df["two"]
>>> df["flag"] = df["one"] > 2
   one  two  three  flag
a  1.0  1.0   1.0  False
b  2.0  2.0   4.0  False
c  3.0  3.0   9.0  True
d  NaN  4.0   NaN  False
```

```
>>> del df["two"]
   one  three  flag
a  1.0    1.0  False
b  2.0    4.0  False
c  3.0    9.0  True
d  NaN    NaN  False
>>> df["newcol"] = "Python"
   one  three  flag  newcol
a  1.0    1.0  False  Python
b  2.0    4.0  False  Python
c  3.0    9.0  True   Python
d  NaN    NaN  False  Python
>>> df.insert(1, "bar", df["one"])
   one  bar  three  flag  newcol
a  1.0  1.0   1.0  False  Python
b  2.0  2.0   4.0  False  Python
c  3.0  3.0   9.0  True   Python
d  NaN  NaN   NaN  False  Python
>>> df.assign(newcol2 =
→ np.power(df['one'], 2))
```



DataFrame: Indexing and selection

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

Operation	Syntax	Result
Select column	<code>df[col]</code>	Series
Select row by label	<code>df.loc[label]</code>	Series
Select row by integer location	<code>df.iloc[loc]</code>	Series
Slice rows	<code>df[5 : 10]</code>	DataFrame
Select rows by boolean vector	<code>df[bool_vec]</code>	DataFrame



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- **IO Tools and Essential Functionality**
 - Time Series and Date Functionality
 - Indexing and Data Selection
 - Grouping and Aggregation
 - Merge, join and concatenate
 - Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



The pandas I/O API functions

Data Handling and Manipulation

Data Structures in pandas

IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

Data Description	Reader	Writer
CSV	<i>read_csv</i>	<i>to_csv</i>
JSON	<i>read_json</i>	<i>to_json</i>
XML	<i>read_xml</i>	<i>to_xml</i>
MS Excel	<i>read_excel</i>	<i>to_excel</i>
HDF5 Format	<i>read_hdf</i>	<i>to_hdf</i>
Python Pickle Format	<i>read_pickle</i>	<i>to_pickle</i>
SQL	<i>read_sql</i>	<i>to_sql</i>
Google BigQuery	<i>read_gbq</i>	<i>to_gbq</i>



Load data into dataframes

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity = pd.read_csv('./data/dailyActivity_merged.csv')
>>> dailyCalories = pd.read_csv('./data/dailyCalories_merged.csv')
>>> dailyIntensities = pd.read_csv('./data/dailyIntensities_merged.csv')
>>> dailySteps = pd.read_csv('./data/dailySteps_merged.csv')
>>> sleepDay = pd.read_csv('./data/sleepDay_merged.csv')
>>> weightLogInfo = pd.read_csv('./data/weightLogInfo_merged.csv')
>>> sleepDay.head()
      Id          SleepDay  TotalSleepRecords  TotalMinutesAsleep  TotalTimeInBed
0  1503960366  4/12/2016 12:00:00 AM           1                  327                 346
1  1503960366  4/13/2016 12:00:00 AM           2                  384                 407
2  1503960366  4/15/2016 12:00:00 AM           1                  412                 442
3  1503960366  4/16/2016 12:00:00 AM           2                  340                 367
4  1503960366  4/17/2016 12:00:00 AM           1                  700                 712
```



Explore the data

Data Handling and Manipulation

Data Structures in pandas

IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.head(3)
```

	Id	ActivityDate	TotalSteps	...	Calories
0	1503960366	4/12/2016	13162	...	1985
1	1503960366	4/13/2016	10735	...	1797
2	1503960366	4/14/2016	10460	...	1776

[3 rows x 15 columns]

```
>>> dailyActivity.tail(3)
```

	Id	ActivityDate	TotalSteps	...	Calories
937	8877689391	5/10/2016	10733	...	2832
938	8877689391	5/11/2016	21420	...	3832
939	8877689391	5/12/2016	8064	...	1849

[3 rows x 15 columns]



Explore the data

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Id                940 non-null    int64  
 1   ActivityDate      940 non-null    object  
 2   TotalSteps        940 non-null    int64  
 3   TotalDistance     940 non-null    float64 
 4   TrackerDistance   940 non-null    float64 
 5   LoggedActivitiesDistance 940 non-null  float64 
 6   VeryActiveDistance 940 non-null    float64 
 7   ModeratelyActiveDistance 940 non-null  float64 
 8   LightActiveDistance 940 non-null    float64 
 9   SedentaryActiveDistance 940 non-null  float64 
 10  VeryActiveMinutes 940 non-null    int64  
 11  FairlyActiveMinutes 940 non-null    int64  
 12  LightlyActiveMinutes 940 non-null    int64  
 13  SedentaryMinutes   940 non-null    int64  
 14  Calories          940 non-null    int64  
dtypes: float64(7), int64(7), object(1)
memory usage: 110.3+ KB
```



Explore the data

Data Handling and Manipulation

Data Structures in pandas

IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.shape
```

```
(940, 16)
```

```
>>> dailyActivity.describe()
```

	Id	TotalSteps	TotalDistance	...	Calories
count	9.400000e+02	940.000000	940.000000	...	940.000000
mean	4.855407e+09	7637.910638	5.489702	...	2303.609574
std	2.424805e+09	5087.150742	3.924606	...	718.166862
min	1.503960e+09	0.000000	0.000000	...	0.000000
25%	2.320127e+09	3789.750000	2.620000	...	1828.500000
50%	4.445115e+09	7405.500000	5.245000	...	2134.000000
75%	6.962181e+09	10727.000000	7.712500	...	2793.250000
max	8.877689e+09	36019.000000	28.030001	...	4900.000000

```
[8 rows x 14 columns]
```



Explore the data

Data Handling and Manipulation

Data Structures in pandas

IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.describe(percentiles=[0.2, 0.4, 0.6, 0.8])
```

	Id	TotalSteps	TotalDistance	...	Calories
count	9.400000e+02	940.000000	940.000000	...	940.000000
mean	4.855407e+09	7637.910638	5.489702	...	2303.609574
std	2.424805e+09	5087.150742	3.924606	...	718.166862
min	1.503960e+09	0.000000	0.000000	...	0.000000
20%	2.026352e+09	2991.800000	2.056000	...	1725.000000
40%	4.020333e+09	6116.600000	4.266000	...	2010.000000
50%	4.445115e+09	7405.500000	5.245000	...	2134.000000
60%	5.553957e+09	8865.400000	6.260000	...	2364.600000
80%	7.086362e+09	11599.000000	8.256000	...	2899.600000
max	8.877689e+09	36019.000000	28.030001	...	4900.000000

[10 rows x 14 columns]



Explore the data

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> penguins = sns.load_dataset("penguins")
>>> penguins.head(4)
   species      island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g     sex
0    Adelie  Torgersen         39.1          18.7            181.0       3750.0   Male
1    Adelie  Torgersen         39.5          17.4            186.0       3800.0 Female
2    Adelie  Torgersen         40.3          18.0            195.0       3250.0 Female
3    Adelie  Torgersen        NaN             NaN             NaN           NaN      NaN
>>> penguins.sum()
species              AdelieAdelieAdelieAdelieAdelieAdelieAdel...
island                TorgersenTorgersenTorgersenTorgersenTorgersenT...
bill_length_mm          15021.3
bill_depth_mm            5865.7
flipper_length_mm         68713.0
body_mass_g              1437000.0
>>> penguins.count()
penguins.count()
species          344
island           344
bill_length_mm    342
bill_depth_mm     342
flipper_length_mm 342
body_mass_g        342
sex               333
```



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- **Time Series and Date Functionality**
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Converting from object to date

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity['ActivityDate'] = pd.to_datetime(dailyActivity['ActivityDate'])
>>> dailyActivity.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  --  
 0   Id                940 non-null    int64  
 1   ActivityDate      940 non-null    datetime64[ns]
 2   TotalSteps        940 non-null    int64  
 3   TotalDistance     940 non-null    float64 
 4   TrackerDistance   940 non-null    float64 
 5   LoggedActivitiesDistance 940 non-null  float64 
 6   VeryActiveDistance 940 non-null    float64 
 7   ModeratelyActiveDistance 940 non-null  float64 
 8   LightActiveDistance 940 non-null    float64 
 9   SedentaryActiveDistance 940 non-null  float64 
 10  VeryActiveMinutes 940 non-null    int64  
 11  FairlyActiveMinutes 940 non-null    int64  
 12  LightlyActiveMinutes 940 non-null    int64  
 13  SedentaryMinutes   940 non-null    int64  
 14  Calories           940 non-null    int64  
dtypes: datetime64[ns](1), float64(7), int64(7)
memory usage: 110.3 KB
```



Create date/time fields in dataframes

Data Handling and Manipulation

Data Structures in pandas IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dti = pd.date_range("2020-01-01", periods=4, freq="2H")
>>> dti
DatetimeIndex(['2020-01-01 00:00:00', '2020-01-01 02:00:00', '2020-01-01 04:00:00',
               '2020-01-01 06:00:00'], dtype='datetime64[ns]', freq='2H')
>>> d = {"one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]),
         "two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),
         "date":dti}
>>> df=pd.DataFrame(d)
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, a to d
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
 ---  --     --     --    
 0   one     3 non-null      float64 
 1   two     4 non-null      float64 
 2   date    4 non-null      datetime64[ns]
dtypes: datetime64[ns](1), float64(2)
memory usage: 128.0+ bytes
```



Create date/time fields in dataframes

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality

Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> df['date'] = df['date'] + pd.Timedelta("2 hours")
   one    two           date
a  1.0  1.0 2020-01-01 02:00:00
b  2.0  2.0 2020-01-01 04:00:00
c  3.0  3.0 2020-01-01 06:00:00
d  NaN  4.0 2020-01-01 08:00:00
>>> df['date'] = df['date'] + pd.Timedelta("2 days")
   one    two           date
a  1.0  1.0 2020-01-03 02:00:00
b  2.0  2.0 2020-01-03 04:00:00
c  3.0  3.0 2020-01-03 06:00:00
d  NaN  4.0 2020-01-03 08:00:00
>>> df['dateDoW'] = df['date'].dt.day_name()
   one    two           date  dateDoW
a  1.0  1.0 2020-01-03 02:00:00  Friday
b  2.0  2.0 2020-01-03 04:00:00  Friday
c  3.0  3.0 2020-01-03 06:00:00  Friday
d  NaN  4.0 2020-01-03 08:00:00  Friday
```



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection**
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Indexing by boolean vectors

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity['ActivityDoW'] = dailyActivity['ActivityDate'].dt.day_name()
>>> dailyActivity.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Id               940 non-null    int64  
 1   ActivityDate     940 non-null    datetime64[ns]
 2   TotalSteps       940 non-null    int64  
 3   TotalDistance    940 non-null    float64 
 4   TrackerDistance  940 non-null    float64 
 5   LoggedActivitiesDistance 940 non-null    float64 
 6   VeryActiveDistance 940 non-null    float64 
 7   ModeratelyActiveDistance 940 non-null    float64 
 8   LightActiveDistance 940 non-null    float64 
 9   SedentaryActiveDistance 940 non-null    float64 
 10  VeryActiveMinutes 940 non-null    int64  
 11  FairlyActiveMinutes 940 non-null    int64  
 12  LightlyActiveMinutes 940 non-null    int64  
 13  SedentaryMinutes 940 non-null    int64  
 14  Calories          940 non-null    int64  
 15  ActivityDoW       940 non-null    object 
dtypes: datetime64[ns](1), float64(7), int64(7), object(1)
memory usage: 117.6+ KB
```



Indexing by boolean vectors

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity['ActivityDoW']=='Saturday'
0      False
1      False
...
939    False
Name: ActivityDoW, Length: 940, dtype: bool
>>> sat_activity = dailyActivity[dailyActivity['ActivityDoW']=='Saturday']
>>> sat_activity.head(3)
   Id ActivityDate  TotalSteps  TotalDistance ... Calories  ActivityDoW
4  1503960366  2016-04-16       12669        8.16    ...    1863  Saturday
11 1503960366  2016-04-23       14371        9.04    ...    1949  Saturday
18 1503960366  2016-04-30       14673        9.25    ...    1947  Saturday
[3 rows x 16 columns]
>>> we_activity = dailyActivity[(dailyActivity['ActivityDoW']=='Saturday') | (dailyActivity['ActivityDoW']=='Sunday')]
>>> we_activity.head(3)
   Id ActivityDate  TotalSteps  TotalDistance ... Calories  ActivityDoW
4  1503960366  2016-04-16       12669        8.16    ...    1863  Saturday
5  1503960366  2016-04-17       9705        6.48    ...    1728  Sunday
11 1503960366  2016-04-23       14371        9.04    ...    1949  Saturday
[3 rows x 16 columns]
>>> we_activity = dailyActivity[dailyActivity['ActivityDoW'].isin(['Saturday', 'Sunday'])]
>>> we_activity.head(3)
   Id ActivityDate  TotalSteps  TotalDistance ... Calories  ActivityDoW
4  1503960366  2016-04-16       12669        8.16    ...    1863  Saturday
5  1503960366  2016-04-17       9705        6.48    ...    1728  Sunday
11 1503960366  2016-04-23       14371        9.04    ...    1949  Saturday
[3 rows x 16 columns]
```



Indexing by integer location

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity.iloc[100,:]
```

Id	1844505072
ActivityDate	2016-04-20 00:00:00
TotalSteps	8
TotalDistance	0.01
TrackerDistance	0.01
LoggedActivitiesDistance	0.0
VeryActiveDistance	0.0
ModeratelyActiveDistance	0.0
LightActiveDistance	0.01
SedentaryActiveDistance	0.0
VeryActiveMinutes	0
FairlyActiveMinutes	0
LightlyActiveMinutes	1
SedentaryMinutes	1439
Calories	1349
ActivityDoW	Wednesday
Name:	100, dtype: object

```
>>> dailyActivity.iloc[100, [2, 3, 4]]
```

TotalSteps	8	
TotalDistance	0.01	
TrackerDistance	0.01	
Name:	100, dtype: object	
>>> dailyActivity.iloc[100:103, [2, 4]]	TotalSteps	TrackerDistance
100	8	0.01
101	8054	5.32
102	5372	3.55



Indexing by label

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity.loc[100,:]
```

Id	1844505072
ActivityDate	2016-04-20 00:00:00
TotalSteps	8
TotalDistance	0.01
TrackerDistance	0.01
LoggedActivitiesDistance	0.0
VeryActiveDistance	0.0
ModeratelyActiveDistance	0.0
LightActiveDistance	0.01
SedentaryActiveDistance	0.0
VeryActiveMinutes	0
FairlyActiveMinutes	0
LightlyActiveMinutes	1
SedentaryMinutes	1439
Calories	1349
ActivityDoW	Wednesday
Name:	100, dtype: object

```
>>> dailyActivity.set_index('1' +  
    → dailyActivity.index.astype(str),  
    → inplace=True)  
>>> dailyActivity.loc['1100',  
    → ['TotalSteps', 'Calories',  
    → 'ActivityDate']]  
TotalSteps 8  
Calories 1349  
ActivityDate 2016-04-20 00:00:00  
Name: 100, dtype: object  
>>> dailyActivity.loc[['1100', '1101',  
    → '1102'], ['TotalSteps', 'Calories',  
    → 'ActivityDate']]  
TotalSteps TrackerDistance  
          TotalSteps Calories ActivityDate  
1100        8      1349  2016-04-20  
1101      8054      2062  2016-04-21  
1102      5372      1827  2016-04-22
```



Indexing by column label and row slicing

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity[['ActivityDate', 'TotalSteps']]  
    ActivityDate  TotalSteps  
110  2016-04-12      13162  
...  
1939  2016-05-12      8064  
[940 rows x 2 columns]  
>>> dailyActivity[dailyActivity.columns[[1,2]]]  
    ActivityDate  TotalSteps  
110  2016-04-12      13162  
...  
1939  2016-05-12      8064  
[940 rows x 2 columns]  
>>> dailyActivity[10:15]  
    Id ActivityDate  TotalSteps  TotalDistance  Calories  ActivityDoW  
110  1503960366  2016-04-22      12764        8.13     1827    Friday  
111  1503960366  2016-04-23      14371        9.04     1949  Saturday  
112  1503960366  2016-04-24      10039        6.41     1788   Sunday  
113  1503960366  2016-04-25      15355        9.80     2013   Monday  
114  1503960366  2016-04-26      13755        8.79     1970 Tuesday  
[5 rows x 16 columns]  
>>> dailyActivity[10:15][['TotalSteps', 'TotalDistance']]  
    TotalSteps  TotalDistance  
110        12764        8.13  
111        14371        9.04  
112        10039        6.41  
113        15355        9.80  
114        13755        8.79
```



Row slicing by label

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> dailyActivity.loc['l100':'l103'][['Id', 'TotalSteps']]  
      Id  TotalSteps  
l100  1844505072          8  
l101  1844505072        8054  
l102  1844505072        5372  
l103  1844505072        3570  
>>> dailyActivity.loc['l100':'l103', 'Id':'TotalSteps']  
      Id  ActivityDate  TotalSteps  
l100  1844505072  4/20/2016          8  
l101  1844505072  4/21/2016        8054  
l102  1844505072  4/22/2016        5372  
l103  1844505072  4/23/2016        3570  
>>> dailyActivity.loc['l100':'l103', 'Id'] = 0  
      Id  ActivityDate  TotalSteps  
l100    0   4/20/2016          8  
l101    0   4/21/2016        8054  
l102    0   4/22/2016        5372  
l103    0   4/23/2016        3570  
>>> dailyActivity.iloc[100:103, dailyActivity.columns.get_loc('Id'):dailyActivity.columns.get_loc('TotalSteps')]  
      Id  ActivityDate  
l100    0   4/20/2016  
l101    0   4/21/2016  
l102    0   4/22/2016
```



Random sample selection

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots
Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.sample(n=4)[['Id', 'TotalSteps', 'TotalDistance']]  
          Id  TotalSteps  TotalDistance  
1414    4388161847      8863        6.82  
1298    3372868164      8844        6.03  
1550    5553957443     11886        7.76  
1359    4020332650         0        0.00  
>>> dailyActivity.sample(n=4, weights=np.random.rand(len(dailyActivity)))[['Id', 'TotalSteps', 'TotalDistance']]  
          Id  TotalSteps  TotalDistance  
1407    4319703577      3672        2.46  
1168    2022484408     10119        7.19  
1929    8877689391      4790        3.64  
1465    4445114986     2923        1.96  
>>> dailyActivity.sample(n=4, weights=np.random.rand(len(dailyActivity)), replace=True)[['Id', 'TotalSteps',  
↔  'TotalDistance']]  
          Id  TotalSteps  TotalDistance  
1819    8378563200     12386        9.82  
1717    7007744171     14816       10.98  
1723    7007744171     11085        7.42  
1574    5577150313     12574       9.42  
>>> dailyActivity.sample(frac=0.003, replace=True)[['Id', 'TotalSteps', 'TotalDistance']]  
          Id  TotalSteps  TotalDistance  
1258    2347167796     16901       11.37  
1502    4558609924      6435        4.25  
1696    6962181067     12627       8.35
```



Misc data selection functionality

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation

Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.where(dailyActivity['TotalSteps']>11100)
      Id ActivityDate TotalSteps TotalDistance ... Calories
10  1.503960e+09  4/12/2016    13162.0     8.500000 ... 1985.0
11        NaN        NaN        NaN        NaN ...     NaN
...
1939       ...       ...       ...       ...
[940 rows x 15 columns]
>>> dailyActivity.mask(dailyActivity['TotalSteps']>11100)
      Id ActivityDate TotalSteps TotalDistance ... Calories
10        NaN        NaN        NaN        NaN ...     NaN
11  1.503960e+09  4/13/2016    10735.0      6.97 ... 1797.0
...
1939  8.877689e+09  5/12/2016    8064.0      6.12 ... 1849.0
[940 rows x 15 columns]
>>> dailyActivity2 = dailyActivity
>>> dailyActivity2.loc['l120':'l122', 'Id'] = 0
>>> dailyActivity.loc['l120':'l122', 'Id']
l120    0
l121    0
l122    0
Name: Id, dtype: int64
>>> dailyActivity3 = dailyActivity.copy(deep=False)
Shallow Copy: Changes reflect into the original dataframe iff you change the existing data but reflect into the copy
↳ only if you add more data that is not available in the original dataframe.
>>> dailyActivity4 = dailyActivity.copy(deep=True)
Deep Copy: This is a completely independent dataframe with its own data and index and no matter what modifications were
↳ done, they will reflect into the original dataframe.
```



Misc data selection functionality

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> penguins = sns.load_dataset("penguins")
>>> penguins.head(4)
species      island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g    sex
0   Adelie  Torgersen          39.1           18.7            181.0       3750.0  Male
1   Adelie  Torgersen          39.5           17.4            186.0       3800.0 Female
2   Adelie  Torgersen          40.3           18.0            195.0       3250.0 Female
3   Adelie  Torgersen           NaN             NaN              NaN             NaN             NaN             NaN
>>> penguins == penguins
species      island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g    sex
0      True      True        True        True        True        True        True        True
1      True      True        True        True        True        True        True        True
2      True      True        True        True        True        True        True        True
3      True      True       False       False       False       False       False       False
>>> np.all(penguins == penguins)
False
>>> penguins.equals(penguins)
True
```



Dataframe index change

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation

Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> sat_activity = dailyActivity[dailyActivity['ActivityDoW']=='Saturday'][['Id', 'TotalSteps', 'TotalDistance',
   ↪ 'Calories']]
>>> sat_activity.head(3)
      Id  TotalSteps  TotalDistance  Calories
14  1503960366       12669        8.16     1863
111 1503960366       14371        9.04     1949
118 1503960366       14673        9.25     1947
>>> sat_activity.loc['10',:]
"KeyError"
>>> sat_activity = dailyActivity[dailyActivity['ActivityDoW']=='Saturday'][['Id', 'TotalSteps', 'TotalDistance',
   ↪ 'Calories']].reset_index()
    index      Id  TotalSteps  TotalDistance  Calories
0       14  1503960366       12669        8.160000     1863
1       111 1503960366       14371        9.040000     1949
2       118 1503960366       14673        9.250000     1947
>>> sat_activity = dailyActivity[dailyActivity['ActivityDoW']=='Saturday'][['Id', 'TotalSteps', 'TotalDistance',
   ↪ 'Calories']].reset_index(drop=True)
      Id  TotalSteps  TotalDistance  Calories
0   1503960366       12669        8.160000     1863
1   1503960366       14371        9.040000     1949
2   1503960366       14673        9.250000     1947
>>> sat_activity.set_index('Id', inplace=True)
      TotalSteps  TotalDistance  Calories
Id
1503960366       12669        8.16     1863
1503960366       14371        9.04     1949
1503960366       14673        9.25     1947
```



Editing/dropping dataframe columns

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity['TotalSteps'] = pd.cut(dailyActivity['TotalSteps'], bins=3, labels=['low', 'medium', 'high'])
   Id ActivityDate TotalSteps ... Calories ActivityDoW
10 1503960366 2016-04-12    medium ... 1985      Tuesday
11 1503960366 2016-04-13      low ... 1797     Wednesday
12 1503960366 2016-04-14      low ... 1776     Thursday
[3 rows x 16 columns]
>>> categories = {'low':0, 'medium':1, 'high':2}
>>> dailyActivity['TotalSteps'] = dailyActivity['TotalSteps'].map(categories)
   Id ActivityDate TotalSteps ... Calories ActivityDoW
10 1503960366 2016-04-12          1 ... 1985      Tuesday
11 1503960366 2016-04-13          0 ... 1797     Wednesday
12 1503960366 2016-04-14          0 ... 1776     Thursday
[3 rows x 16 columns]
>>> sat_activity.drop(['TotalSteps', 'Calories'], axis=1, inplace=True)
>>> sat_activity.head(3)
           TotalDistance
Id
1503960366        8.16
1503960366        9.04
1503960366        9.25
>>> sat_activity.rename(columns={'TotalDistance':'dist'}, inplace=True)
>>> sat_activity.head(3)
           dist
Id
1503960366        8.16
1503960366        9.04
1503960366        9.25
```



Hierarchical indexing (Multiindexing)

Data Handling and Manipulation
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

Hierarchical indexing in pandas, is a multi-level indexing that extends indices to lists of tuples (each of which is unique) and allows for working with high dimensional data.

```
>>> arrays = [["bar", "bar", "baz", "baz", "foo",
   ↪  "foo", "qux", "qux"], ["one", "two", "one", "two",
   ↪  "one", "two", "one", "two"]]
>>> tuples = list(zip(*arrays))
[('bar', 'one'),
 ('bar', 'two'),
 ('baz', 'one'),
 ('baz', 'two'),
 ('foo', 'one'),
 ('foo', 'two'),
 ('qux', 'one'),
 ('qux', 'two')]
>>> index = pd.MultiIndex.from_tuples(tuples,
   ↪  names=["first", "second"])
MultiIndex([('bar', 'one'),
            ('bar', 'two'),
            ('baz', 'one'),
            ('baz', 'two'),
            ('foo', 'one'),
            ('foo', 'two'),
            ('qux', 'one'),
            ('qux', 'two')],  
           names=['first', 'second'])
```

```
>>> s = pd.Series(np.random.randn(8), index=index)
first    second
bar      one       0.469112
         two      -0.282863
baz      one      -1.509059
         two      -1.135632
foo      one       1.212112
         two      -0.173215
qux      one       0.119209
         two      -1.044236
dtype: float64
>>> iterables = [[["bar", "baz", "foo"], ["one",
   ↪  "two"]]]
>>> pd.MultiIndex.from_product(iterables,
   ↪  names=["first", "second"])
MultiIndex([('bar', 'one'),
            ('bar', 'two'),
            ('baz', 'one'),
            ('baz', 'two'),
            ('foo', 'one'),
            ('foo', 'two')],  
           names=['first', 'second'])
```



Slicing in hierarchical indexing

Data Handling and Manipulation
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

Hierarchical indexing in pandas, is a multi-level indexing that extends indices to lists of tuples (each of which is unique) and allows for working with high dimensional data.

```
>>> df = pd.DataFrame(np.random.randn(8,4),
→   index=index)
          0         1         2         3
first second
bar  one    0.672494 -1.640724  1.081894  0.692540
      two    0.548721  1.768512  0.455014 -0.852776
baz  one    0.709193 -0.465862  0.664963 -1.694255
      two    0.167234 -1.334412  1.192999  0.328606
foo  one    0.821292  0.492201  1.499802  1.564745
      two   -0.979484  1.051543  0.734119 -0.478086
qux one   -0.332113  0.380919  1.105194  0.852695
      two    0.572332 -1.063268  0.409088 -1.229017
>>> df.loc[('bar', 'two'), :]
0    0.548721
1    1.768512
2    0.455014
3   -0.852776
Name: (bar, two), dtype: float64
>>> df.loc[[('bar', 'two'), ('foo', 'one')], :]
          0         1         2         3
first second
bar  two    0.548721  1.768512  0.455014 -0.852776
foo  one    0.821292  0.492201  1.499802  1.564745
```

```
>>> df.loc[('bar', 'two'):('foo', 'two'), :]
first second
bar  two    0.548721  1.768512  0.455014 -0.852776
baz  one    0.709193 -0.465862  0.664963 -1.694255
      two    0.167234 -1.334412  1.192999  0.328606
foo  one    0.821292  0.492201  1.499802  1.564745
      two   -0.979484  1.051543  0.734119 -0.478086
>>> df.iloc[2, :]
0    0.709193
1   -0.465862
2    0.664963
3   -1.694255
Name: (baz, one), dtype: float64
>>> df.iloc[2:5, :]
          0         1         2         3
first second
baz  one    0.709193 -0.465862  0.664963 -1.694255
      two    0.167234 -1.334412  1.192999  0.328606
foo  one    0.821292  0.492201  1.499802  1.564745
```



Handling missing data

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation

Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots

Stacking Plots
Regression Plots

```
>>> import seaborn as sns
>>> penguins = sns.load_dataset("penguins")
>>> penguins.head(4)
   species      island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g     sex
0   Adelie  Torgersen          39.1           18.7             181.0       3750.0   Male
1   Adelie  Torgersen          39.5           17.4             186.0       3800.0 Female
2   Adelie  Torgersen          40.3           18.0             195.0       3250.0 Female
3   Adelie  Torgersen           NaN            NaN              NaN            NaN        NaN
>>> penguins.isna().head(4)
   species      island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g     sex
0    False     False        False        False        False        False    False
1    False     False        False        False        False        False    False
2    False     False        False        False        False        False    False
3    False     False       True         True        True         True    True
>>> np.any(penguins['sex'].isna())
True
>>> np.any(penguins.isna(), axis=0)
species          False
island           False
bill_length_mm   True
bill_depth_mm    True
flipper_length_mm  True
body_mass_g      True
sex              True
dtype: bool
```



Handling missing data

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

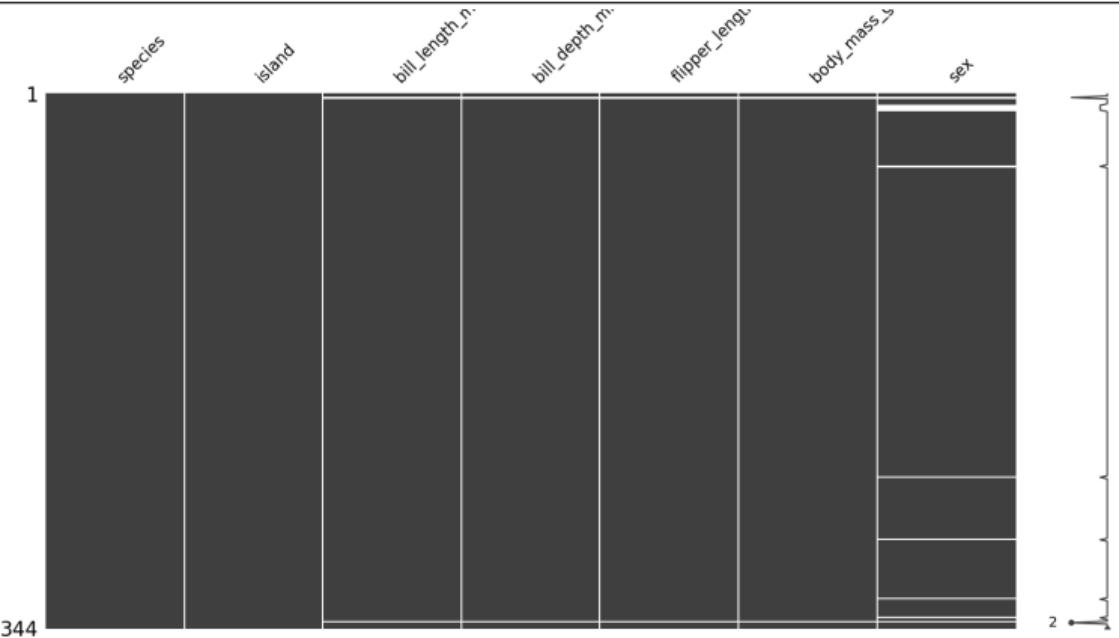
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> import missingno as msn
>>> msn.matrix(penguins)
>>> plt.show()
```





Handling missing data

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

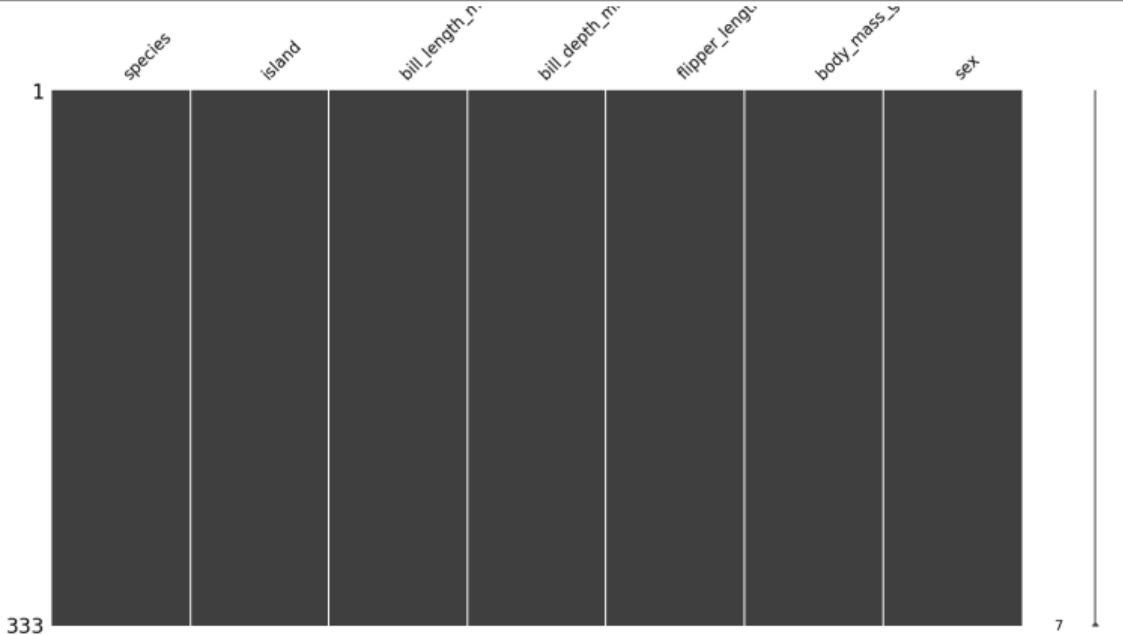
Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> penguins.dropna(how='any', inplace=True)
>>> msn.matrix(penguins)
>>> plt.show()
```





Handling missing data

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

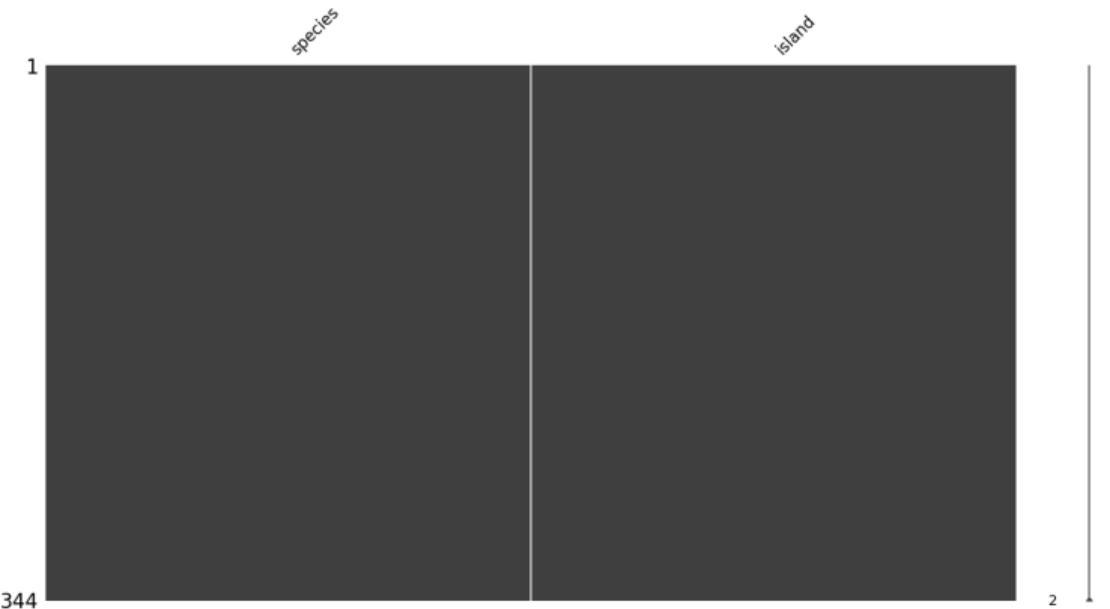
Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> penguins.dropna(how='any', axis=1, inplace=True)
>>> msn.matrix(penguins)
>>> plt.show()
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation**
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Group by field/column

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1

genre	total
adventure	7
fantasy	8
romance	3

```
SELECT genre, SUM(qty) AS total
FROM Books
GROUP BY genre
```



Group by field/column

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation

Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> unique_users = dailyActivity['Id'].unique()
>>> print('Number of unique users in the database is: {}'.format(len(unique_users)))
Number of unique users in the database is: 33
>>> dailyActivity.groupby('Id')[['Calories']].mean().head(5)
Id
1.503960e+09    1816.419355
1.624580e+09    1483.354839
1.644430e+09    2811.300000
1.844505e+09    1573.483871
1.927972e+09    2172.806452
Name: Calories, dtype: float64
>>> dailyActivity.groupby('Id', sort=True)[['Calories', 'TotalDistance']].mean().head()
          Calories  TotalDistance
Id
1503960366    1816.419355      7.809677
1624580081    1483.354839      3.914839
1644430081    2811.300000      5.295333
1844505072    1605.666667      1.735417
>>> dailyActivity.groupby('Id', sort=True).agg({'Calories':'mean', 'TotalDistance':'sum'}).head()
          Calories  TotalDistance
Id
0            1557.750000      11.710000
1503960366    1816.419355     242.099999
1624580081    1483.354839     121.360001
1644430081    2811.300000     158.860001
1844505072    1605.666667      41.650000
```



Common aggregation functions

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

Function	Description
<code>mean()</code>	Compute mean of groups
<code>sum()</code>	Compute sum of group values
<code>size()</code>	Compute group sizes
<code>count()</code>	Compute count of group
<code>std()</code>	Standard deviation of groups
<code>var()</code>	Compute variance of groups
<code>sem()</code>	Standard error of the mean of groups
<code>describe()</code>	Generates descriptive statistics
<code>first()</code>	Compute first of group values
<code>last()</code>	Compute last of group values
<code>nth()</code>	Take nth value, or a subset if n is a list
<code>min()</code>	Compute min of group values
<code>max()</code>	Compute max of group values



Group by field/column

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.groupby('Id', sort=True).agg({'Calories':'mean', 'TotalDistance': lambda x:x.max()-x.mean()}).head()  
          Calories  TotalDistance  
Id  
1503960366  1816.419355      4.400323  
1624580081  1483.354839      24.115162  
1644430081  2811.300000      7.944666  
1844505072  1605.666667      3.514583  
>>> dailyActivity.groupby('Id', sort=True)[['Calories', 'TotalDistance']].agg([np.sum, np.mean, np.std]).head(5)  
          Calories           TotalDistance  
                  sum        mean        std        sum        mean        std  
Id  
1624580081    45984  1483.354839  256.582871  121.360001  3.914839  4.796242  
1644430081    84339  2811.300000  507.488349  158.860001  5.295333  3.143581  
1844505072    38536  1605.666667  258.297379  41.650000  1.735417  1.719575  
>>> dailyActivity.groupby('Id', sort=True)[['Calories', 'TotalDistance']].agg([np.sum, np.mean,  
↔ np.std]).rename(columns={"sum": "x", "mean": "y", "std": "z"}).head(5)  
          Calories           TotalDistance  
                  x          y          z          x          y          z  
Id  
0            12462  1557.750000  493.147544  11.710000  1.463750  2.054785  
1503960366   56309  1816.419355  352.574793  242.099999  7.809677  1.984180  
1624580081   45984  1483.354839  256.582871  121.360001  3.914839  4.796242  
1644430081   84339  2811.300000  507.488349  158.860001  5.295333  3.143581  
1844505072   38536  1605.666667  258.297379  41.650000  1.735417  1.719575
```



Group by field/column

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> dailyActivity.groupby('Id', sort=True).agg(avg_cal=('Calories', 'mean'), max_dist=('TotalDistance', 'max')).head(5)
                                         avg_cal      max_dist
Id
1503960366    1816.419355   12.210000
1624580081    1483.354839   28.030001
1644430081    2811.300000   13.240000

>>> dailyActivity.groupby('Id', sort=True).agg(**{'avg>cal':('Calories', 'mean'), 'max$dist':('TotalDistance',
-> 'max')}).head(5)
                                         avg>cal      max$dist
Id
0            1557.750000   5.320000
1503960366    1816.419355   12.210000
1624580081    1483.354839   28.030001
1644430081    2811.300000   13.240000
1844505072    1605.666667   5.250000
```



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- **Merge, join and concatenate**
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Concatenation (row stacking)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> df1 = pd.DataFrame({ "A": ["A0",  
    "A1", "A2", "A3"], "B": ["B0", "B1",  
    "B2", "B3"], "C": ["C0", "C1", "C2",  
    "C3"], "D": ["D0", "D1", "D2",  
    "D3"]}, index=[0, 1, 2, 3])  
>>> df2 = pd.DataFrame({ "A": ["A4",  
    "A5", "A6", "A7"], "B": ["B4", "B5",  
    "B6", "B7"], "C": ["C4", "C5", "C6",  
    "C7"], "D": ["D4", "D5", "D6",  
    "D7"]}, index=[4, 5, 6, 7])  
>>> df3 = pd.DataFrame({ "A": ["A8",  
    "A9", "A10", "A11"], "B": ["B8",  
    "B9", "B10", "B11"], "C": ["C8",  
    "C9", "C10", "C11"], "D": ["D8",  
    "D9", "D10", "D11"]}, index=[8, 9,  
    10, 11])  
>>> result = pd.concat([df1, df2, df3],  
    axis=0)
```

	df1				Result			
	A	B	C	D	A	B	C	D
0	A0	B0	C0	D0	A0	B0	C0	D0
1	A1	B1	C1	D1	A1	B1	C1	D1
2	A2	B2	C2	D2	A2	B2	C2	D2
3	A3	B3	C3	D3	A3	B3	C3	D3
	df2				df2			
	A	B	C	D	A	B	C	D
4	A4	B4	C4	D4	A4	B4	C4	D4
5	A5	B5	C5	D5	A5	B5	C5	D5
6	A6	B6	C6	D6	A6	B6	C6	D6
7	A7	B7	C7	D7	A7	B7	C7	D7
	df3				df3			
	A	B	C	D	A	B	C	D
8	A8	B8	C8	D8	A8	B8	C8	D8
9	A9	B9	C9	D9	A9	B9	C9	D9
10	A10	B10	C10	D10	A10	B10	C10	D10
11	A11	B11	C11	D11	A11	B11	C11	D11



Concatenate dataframe with series

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> df1 = pd.DataFrame({"A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"],  
→   "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]}, index=[0, 1, 2, 3])  
>>> df4 = pd.DataFrame({"B": ["B2", "B3", "B6", "B7"], "D": ["D2", "D3", "D6", "D7"],  
→   "F": ["F2", "F3", "F6", "F7"]}, index=[2, 3, 6, 7])  
>>> s1 = pd.Series(["X0", "X1", "X2", "X3"], name="X")  
  
>>> result = pd.concat([df1, s1], axis=1)
```

df1				s1	Result						
	A	B	C	D		X	A	B	C	D	X
0	A0	B0	C0	D0	0	X0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	X1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	X2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	X3	3	A3	B3	C3	D3



Concatenation with group keys

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> df1 = pd.DataFrame({"A": ["A0",  
    "A1", "A2", "A3"], "B": ["B0", "B1",  
    "B2", "B3"], "C": ["C0", "C1", "C2",  
    "C3"], "D": ["D0", "D1", "D2",  
    "D3"]}, index=[0, 1, 2, 3])  
>>> df2 = pd.DataFrame({"A": ["A4",  
    "A5", "A6", "A7"], "B": ["B4", "B5",  
    "B6", "B7"], "C": ["C4", "C5", "C6",  
    "C7"], "D": ["D4", "D5", "D6",  
    "D7"]}, index=[4, 5, 6, 7])  
>>> df3 = pd.DataFrame({"A": ["A8",  
    "A9", "A10", "A11"], "B": ["B8",  
    "B9", "B10", "B11"], "C": ["C8",  
    "C9", "C10", "C11"], "D": ["D8",  
    "D9", "D10", "D11"]}, index=[8, 9,  
    10, 11])  
>>> result = pd.concat([df1, df2, df3],  
    keys=["x", "y", "z"])
```

df1					Result					
	A	B	C	D	x	0	A0	B0	C0	D0
0	A0	B0	C0	D0	x	1	A1	B1	C1	D1
1	A1	B1	C1	D1	x	2	A2	B2	C2	D2
2	A2	B2	C2	D2	x	3	A3	B3	C3	D3
3	A3	B3	C3	D3	df2					
df2					y	4	A4	B4	C4	D4
4	A4	B4	C4	D4	y	5	A5	B5	C5	D5
5	A5	B5	C5	D5	y	6	A6	B6	C6	D6
6	A6	B6	C6	D6	y	7	A7	B7	C7	D7
7	A7	B7	C7	D7	df3					
df3					z	8	A8	B8	C8	D8
8	A8	B8	C8	D8	z	9	A9	B9	C9	D9
9	A9	B9	C9	D9	z	10	A10	B10	C10	D10
10	A10	B10	C10	D10	z	11	A11	B11	C11	D11
11	A11	B11	C11	D11						



Concatenation with group keys

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> df1 = pd.DataFrame({"A": ["A0",
→ "A1", "A2", "A3"], "B": ["B0", "B1",
→ "B2", "B3"], "C": ["C0", "C1", "C2",
→ "C3"], "D": ["D0", "D1", "D2",
→ "D3"]}, index=[0, 1, 2, 3])
>>> df2 = pd.DataFrame({"A": ["A4",
→ "A5", "A6", "A7"], "B": ["B4", "B5",
→ "B6", "B7"], "C": ["C4", "C5", "C6",
→ "C7"], "D": ["D4", "D5", "D6",
→ "D7"]}, index=[4, 5, 6, 7])
>>> df3 = pd.DataFrame({"A": ["A8",
→ "A9", "A10", "A11"], "B": ["B8",
→ "B9", "B10", "B11"], "C": ["C8",
→ "C9", "C10", "C11"], "D": ["D8",
→ "D9", "D10", "D11"]}, index=[8, 9,
→ 10, 11])
>>> result = pd.concat({"x": df1, "y":
→ df2, "z": df3})
```

df1					Result					
	A	B	C	D	X	0	A0	B0	C0	D0
0	A0	B0	C0	D0	X	1	A1	B1	C1	D1
1	A1	B1	C1	D1	X	2	A2	B2	C2	D2
2	A2	B2	C2	D2	X	3	A3	B3	C3	D3
3	A3	B3	C3	D3	df2					
	A	B	C	D	Y	4	A4	B4	C4	D4
4	A4	B4	C4	D4	Y	5	A5	B5	C5	D5
5	A5	B5	C5	D5	Y	6	A6	B6	C6	D6
6	A6	B6	C6	D6	Y	7	A7	B7	C7	D7
7	A7	B7	C7	D7	df3					
	A	B	C	D	Z	8	A8	B8	C8	D8
8	A8	B8	C8	D8	Z	9	A9	B9	C9	D9
9	A9	B9	C9	D9	Z	10	A10	B10	C10	D10
10	A10	B10	C10	D10	Z	11	A11	B11	C11	D11
11	A11	B11	C11	D11						



"Column" join types

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

Table 1

1		
2		

Table 2

1		
3		
4		

Outer Join

1			
2			
3			
4			

Inner Join

1			

Left Join

1			
2			

Outer Join

1			
2			
3			
4			

Cross Join

1			1
1			3
1			4
2			1
2			3
2			4



Full outer join with concatenate

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> df1 = pd.DataFrame({"A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"],  
    ↪ "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]}, index=[0, 1, 2, 3])  
>>> df4 = pd.DataFrame({"B": ["B2", "B3", "B6", "B7"], "D": ["D2", "D3", "D6", "D7"],  
    ↪ "F": ["F2", "F3", "F6", "F7"]}, index=[2, 3, 6, 7])  
>>> result = pd.concat([df1, df4], axis=1, join='outer') #default
```

df1				df4			Result							
	A	B	C	D	B	D	F	A	B	C	D	B	D	F
0	A0	B0	C0	D0				AD	B0	CD	D0	NaN	NaN	NaN
1	A1	B1	C1	D1				A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3	A3	B3	C3	D3	B3	D3	F3
					B6	D6	F6					B6	D6	F6
					B7	D7	F7					B7	D7	F7



Inner join with concatenate

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> df1 = pd.DataFrame({"A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"],  
    ↪ "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]}, index=[0, 1, 2, 3])  
>>> df4 = pd.DataFrame({"B": ["B2", "B3", "B6", "B7"], "D": ["D2", "D3", "D6", "D7"],  
    ↪ "F": ["F2", "F3", "F6", "F7"]}, index=[2, 3, 6, 7])  
>>> result = pd.concat([df1, df4], axis=1, join='inner')
```

df1				df4			Result							
	A	B	C	D	B	D	F	A	B	C	D	B	D	F
0	A0	B0	C0	D0										
1	A1	B1	C1	D1										
2	A2	B2	C2	D2	B2	D2	F2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3	A3	B3	C3	D3	B3	D3	F3
					B6	D6	F6							
					B7	D7	F7							



Database-style left join

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> left = pd.DataFrame({"key1": ["K0", "K0", "K1", "K2"], "key2": ["K0", "K1", "K0",  
    ↪ "K1"], "A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"]})  
>>> right = pd.DataFrame({"key1": ["K0", "K1", "K1", "K2"], "key2": ["K0", "K0", "K0",  
    ↪ "K0"], "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]})  
>>> result = pd.merge(left, right, how="left", on=["key1", "key2"])
```

left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN



Database-style right join

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> left = pd.DataFrame({"key1": ["K0", "K0", "K1", "K2"], "key2": ["K0", "K1", "K0",  
    ↪ "K1"], "A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"]})  
>>> right = pd.DataFrame({"key1": ["K0", "K1", "K1", "K2"], "key2": ["K0", "K0", "K0",  
    ↪ "K0"], "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]})  
>>> result = pd.merge(left, right, how="right", on=["key1", "key2"])
```

left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A1	B1	C1	D1
2	K1	K0	A2	B2	C2	D2
3	K2	K0	NaN	NaN	C3	D3



Database-style outer join

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

```
>>> left = pd.DataFrame({"key1": ["K0", "K0", "K1", "K2"], "key2": ["K0", "K1", "K0",  
↪ "K1"], "A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"]})  
>>> right = pd.DataFrame({"key1": ["K0", "K1", "K1", "K2"], "key2": ["K0", "K0", "K0",  
↪ "K0"], "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]})  
>>> result = pd.merge(left, right, how="outer", on=["key1", "key2"])
```

left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3



Database-style inner join

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> left = pd.DataFrame({"key1": ["K0", "K0", "K1", "K2"], "key2": ["K0", "K1", "K0",  
    ↪ "K1"], "A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"]})  
>>> right = pd.DataFrame({"key1": ["K0", "K1", "K1", "K2"], "key2": ["K0", "K0", "K0",  
    ↪ "K0"], "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]})  
>>> result = pd.merge(left, right, how="inner", on=["key1", "key2"])
```

left					right				Result							
	key1	key2	A	B		key1	key2	C	D		key1	key2	A	B	C	D
0	K0	K0	A0	B0	0	K0	K0	CD	D0	0	K0	K0	A0	B0	CD	D0
1	K0	K1	A1	B1	1	K1	K0	C1	D1	1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	2	K1	K0	C2	D2	2	K1	K0	A2	B2	C2	D2
3	K2	K1	A3	B3	3	K2	K0	C3	D3							



Database-style cross join

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> left = pd.DataFrame({"key1": ["K0", "K0", "K1", "K2"], "key2": ["K0", "K1", "K0", "K1"], "A": ["A0", "A1", "A2", "A3"], "B": ["B0", "B1", "B2", "B3"]})  
>>> right = pd.DataFrame({"key1": ["K0", "K1", "K1", "K2"], "key2": ["K0", "K0", "K0", "K0"], "C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]})  
>>> result = pd.merge(left, right, how="cross")
```

	left				right				Result							
	key1	key2	A	B	key1	key2	C	D	key1_x	key2_x	A	B	key1_y	key2_y	C	D
0	K0	K0	A0	B0	0	K0	C0	D0	0	K0	A0	B0	0	K0	C0	D0
1	K0	K1	A1	B1	1	K1	C0	D1	1	K0	A0	B0	1	K0	C1	D1
2	K1	K0	A2	B2	2	K1	C0	D2	2	K0	A0	B0	2	K0	C2	D2
3	K2	K1	A3	B3	3	K2	C0	D3	3	K0	A0	B0	3	K0	C3	D3
4	K0	K0	A4	B4	4	K2	A1	B4	4	K0	A0	B0	4	K0	C0	D0
5	K0	K2	A5	B5	5	K1	A1	B5	5	K0	A0	B0	5	K0	C1	D5
6	K0	K3	A6	B6	6	K1	A1	B6	6	K0	A0	B0	6	K0	C1	D6
7	K0	K3	A7	B7	7	K1	A1	B7	7	K0	A0	B0	7	K0	C1	D7
8	K1	K0	A8	B8	8	K2	A0	B8	8	K0	A0	B0	8	K0	C0	D8
9	K0	K0	A9	B9	9	K2	A0	B9	9	K0	A0	B0	9	K0	C0	D9
10	K0	K0	A10	B10	10	K3	A0	B10	10	K0	A0	B0	10	K0	C0	D10
11	K0	K0	A11	B11	11	K3	A0	B11	11	K0	A0	B0	11	K0	C0	D11
12	K2	K2	A12	B12	12	K2	A1	B12	12	K0	A0	B0	12	K0	C0	D12
13	K2	K3	A13	B13	13	K2	A1	B13	13	K0	A0	B0	13	K0	C1	D13
14	K2	K3	A14	B14	14	K2	A1	B14	14	K0	A0	B0	14	K0	C1	D14
15	K2	K3	A15	B15	15	K2	A1	B15	15	K0	A0	B0	15	K0	C1	D15



Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Reshaping by melting

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots

Regression Plots

```
>>> df = pd.DataFrame({"first": ["John", "Mary"], "last": ["Doe", "Bo"], "height": [5.5, 6.0], "weight": [130, 150]})  
>>> df.melt(id_vars=["first", "last"])  
   first last variable  value  
0  John  Doe    height  5.5  
1  Mary  Bo    height  6.0  
2  John  Doe   weight  130.0  
3  Mary  Bo   weight  150.0  
  
>>> df.melt(id_vars=["first", "last"], var_name="quantity")  
   first last quantity  value  
0  John  Doe    height  5.5  
1  Mary  Bo    height  6.0  
2  John  Doe   weight  130.0  
3  Mary  Bo   weight  150.0
```

	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150



	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150



Reshaping by pivoting

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> df = pd.DataFrame({"foo": ["one", "one", "one", "two", "two", "two"], "bar": ["A", "B", "C", "A", "B", "C"], "baz":  
    [1, 2, 3, 4, 5, 6], "zoo": ["x", "y", "z", "q", "w", "t"]})  
>>> df.pivot(index = 'foo', columns='bar', values='baz')  
bar   A   B   C  
foo  
one   1   2   3  
two   4   5   6  
  
>>> df.pivot(index = 'foo', columns=['bar','zoo'], values='baz')  
bar   A   B   C   A   B   C  
zoo   x   y   z   q   w   t  
foo  
one  1.0  2.0  3.0  NaN  NaN  NaN  
two  NaN  NaN  NaN  4.0  5.0  6.0
```

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t



bar	A	B	C
foo			
one	1	2	3
two	4	5	6



Reshaping by stacking

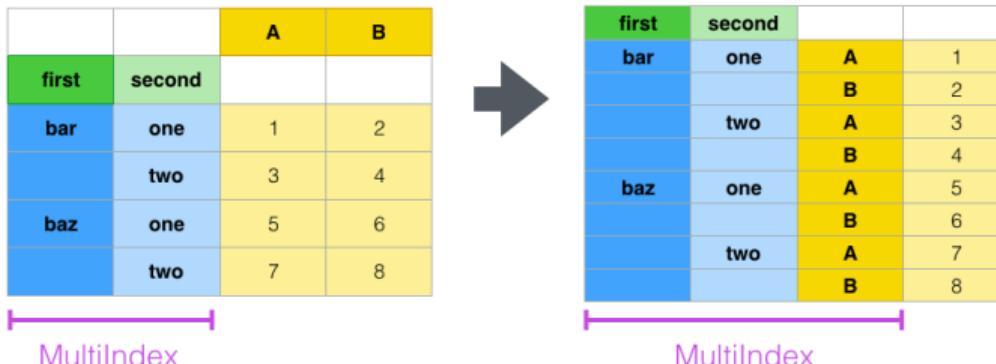
Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> tuples = list(zip(*[['bar", "bar", "baz", "baz"], ["one", "two", "one", "two"]]))  
>>> index = pd.MultiIndex.from_tuples(tuples, names=["first", "second"])  
>>> df = pd.DataFrame({'A': [1, 3, 5, 7], 'B': [2, 4, 6, 8]}, index=index)  
>>> df.stack()  
first  second  
bar    one      A    1  
          B    2  
        two      A    3  
          B    4  
baz    one      A    5  
          B    6  
        two      A    7  
          B    8
```





Reshaping by unstacking

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

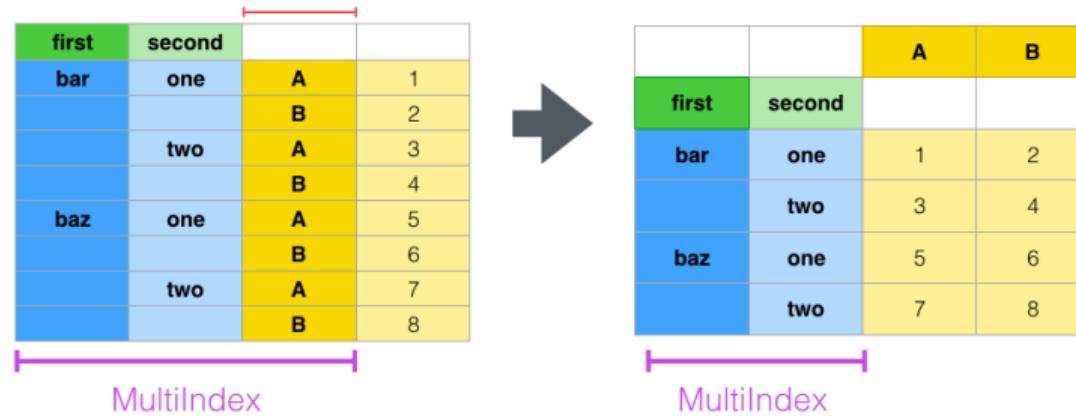
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots

Regression Plots

```
>>> df2=df.stack()  
>>> df2.unstack()  
          A   B  
first second  
bar    one    1  2  
      two    3  4  
baz    one    5  6  
      two    7  8
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Matplotlib vs Seaborn

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots



matplotlib



seaborn

Mainly deployed for basic plotting (line, bar, pie, scatter charts)

A graphics package for data visualization in Python, well integrated with NumPy and pandas with the pyplot module closely mirrors the MATLAB plotting commands

Doesn't include methods to handle categorical variables in certain plot types

Widely used in general, with less documentation

Offers variety of visualization patterns (e.g. advanced statistical plots) and uses easier (fewer) syntax

Extends Matplotlib for integration with dataframes and more straightforward set of commands.

Has methods to plot/handle categorical variables in boxplots and violin plots

Better documentation and widely used for plotting regression analysis



Anatomy of figure

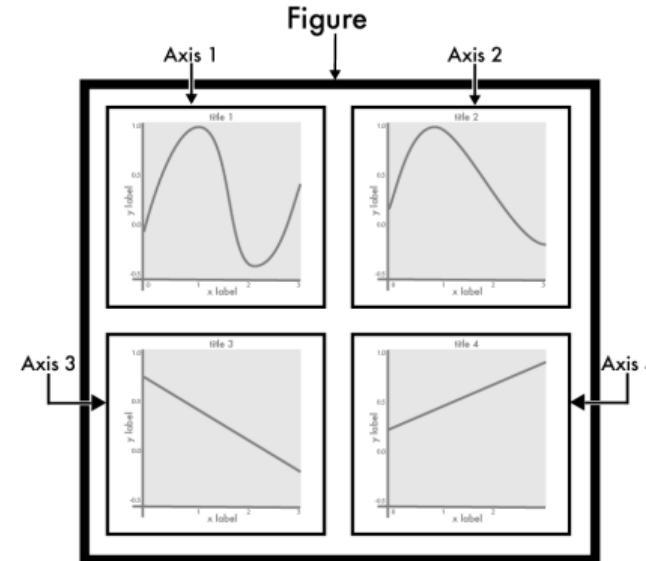
Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> fig, (ax1, ax2, ax3, ax4) = plt.subplots(2, 2, figsize=(10, 10))
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Simple line plots for the iris dataset

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation
Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

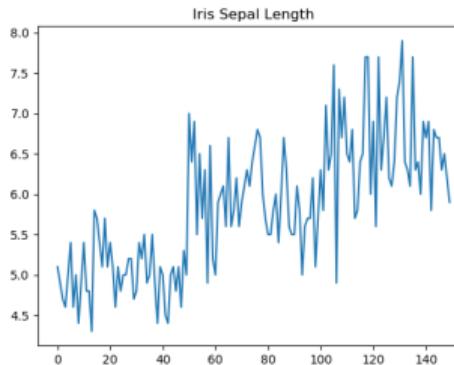
Box and Violin Plots

Stacking Plots

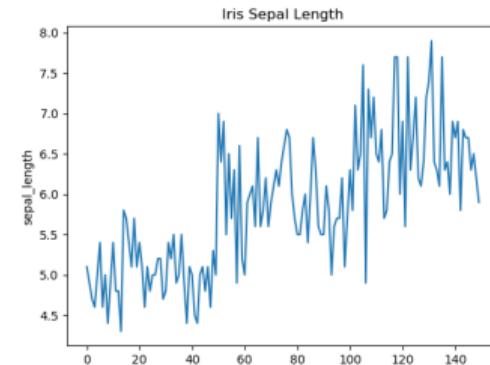
Regression Plots

```
>>> iris = sns.load_dataset("iris")
>>> iris.head(3)
   sepal_length  sepal_width  petal_length  petal_width species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
```

```
# create figure and axis
>>> fig, ax = plt.subplots()
>>> ax.plot(iris.index.to_list(),
→       iris["sepal_length"])
>>> ax.set_title("Iris Sepal Length")
>>> plt.show()
```



```
# create figure and axis
>>> fig, ax = plt.subplots()
>>> sns.lineplot(x=iris.index.to_list(),
→                 y=iris["sepal_length"])
>>> ax.set_title("Iris Sepal Length")
>>> plt.show()
```





Overlaid line plots for the iris dataset

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

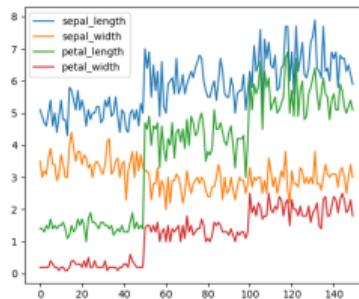
Data Visualization

Line Plots

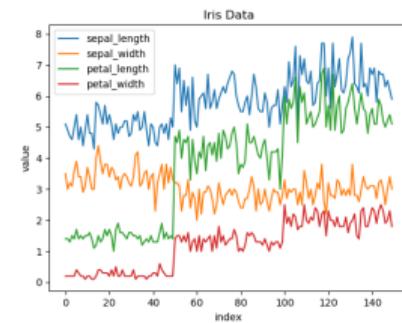
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> iris.drop("species", axis=1).head()
   sepal_length  sepal_width  petal_length
0          5.1         3.5         1.4
1          4.9         3.0         1.4
2          4.7         3.2         1.3
3          4.6         3.1         1.5
>>> fig, ax = plt.subplots()
>>> for column in iris.columns.drop("species"):
...     ax.plot(iris.index.to_list(), iris[column])
>>> ax.set_title("Iris Data")
>>> ax.legend(labels=iris.columns.drop("species"),
...             loc='upper left')
>>> plt.show()
```



```
>>> new_iris = iris.reset_index().drop("species",
...           axis=1).melt(id_vars='index',
...           var_name='property')
      index    property  value
0        0  sepal_length  5.1
1        1  sepal_length  4.9
2        2  sepal_length  4.7
>>> fig, ax = plt.subplots()
>>> sns.lineplot(x='index', y='value',
...                 hue='property', data=new_iris)
>>> ax.set_title("Iris Data")
>>> ax.legend(loc='upper left')
>>> plt.show()
```





Line plots with error bands

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

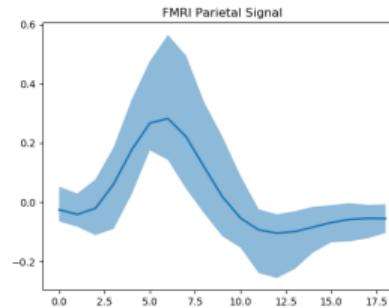
```
>>> fmri = sns.load_dataset("fmri")
>>> fmri.head(3)
   subject  timepoint  event    region    signal
0        s13         18  stim  parietal -0.017552
1        s5          14  stim  parietal -0.080883
2        s12         18  stim  parietal -0.081033
>>> new_fmri = fmri[(fmri['event']=='stim')&(fmri['region']=='parietal')]
   subject  timepoint  event    region    signal
0        s13         18  stim  parietal -0.017552
1        s5          14  stim  parietal -0.080883
2        s12         18  stim  parietal -0.081033
```



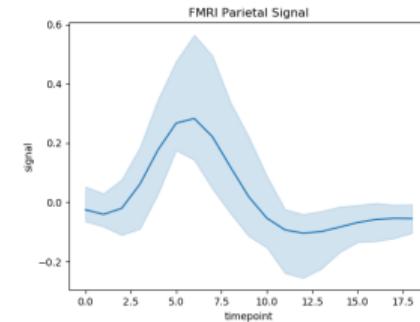
Line plots with error bands

Data Handling and Manipulation
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> fmri_gb_timep =  
    new_fmri.groupby('timepoint')['signal'].agg([np.max,  
    np.mean, np.min])  
    amin           amax      mean  
timepoint  
0       0.052619 -0.024995 -0.064454  
1       0.029822 -0.040460 -0.082174  
>>> fig, ax = plt.subplots()  
>>> ax.fill_between(fmri_gb_timep.index.to_list(),  
    fmri_gb_timep['amin'], fmri_gb_timep['amax'],  
    alpha=.5, linewidth=0)  
>>> ax.plot(fmri_gb_timep.index.to_list(),  
    fmri_gb_timep['mean'], linewidth=2)  
>>> ax.set_title("FMRI Parietal Signal")  
>>> plt.show()
```



```
>>> new_fmri = fmri[(fmri['event']=='stim') &  
    (fmri['region']=='parietal')]  
    subject   timepoint event   region   signal  
0         s13        18   stim   parietal -0.017552  
1         s5         14   stim   parietal -0.080883  
2         s12        18   stim   parietal -0.081033  
>>> fig, ax = plt.subplots()  
>>> sns.lineplot(x='timepoint', y='signal',  
    errorbar=lambda x:(x.min(), x.max()),  
    data=new_fmri)  
>>> ax.set_title("FMRI Parietal Signal")  
>>> plt.show()  
>>>  
>>>
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- **Scatter Plots**
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Simple scatter plots

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> diamonds = sns.load_dataset("diamonds")
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75



Simple scatter plots

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

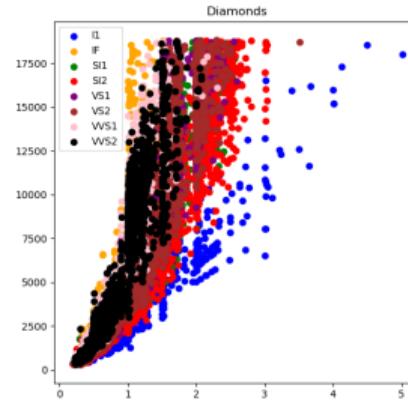
Bar and Distribution Plots

Box and Violin Plots

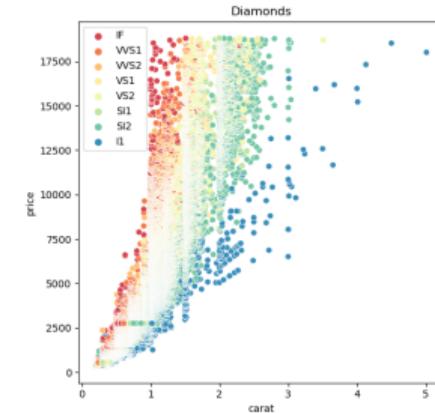
Stacking Plots

Regression Plots

```
>>> colors = {'I1':'blue', 'IF':'orange',
→   'SI1':'green', 'SI2':'red', 'VS1':'purple',
→   'VS2':'brown', 'VVS1':'pink', 'VVS2':'black'}
>>> fig, ax = plt.subplots(figsize=(6.5, 6.5))
>>> for key in colors.keys():
...     data = diamonds[diamonds['clarity'] == key]
...     ax.scatter(data["carat"], data["price"],
→     label=key, c=colors[key])
>>> ax.set_title("Diamonds")
>>> ax.legend()
>>> plt.show()
```



```
>>> fig, ax = plt.subplots(figsize=(6.5, 6.5))
>>> sns.scatterplot(x="carat", y="price",
→   hue="clarity",
→   palette=sns.color_palette("Spectral", 8),
→   data=diamonds, ax=ax)
>>> ax.set_title("Diamonds")
>>> ax.legend()
>>> plt.show()
>>>
>>>
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Histograms

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

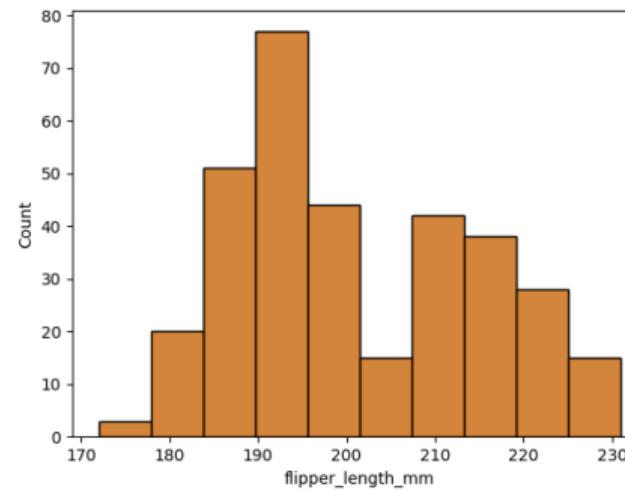
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.histplot(x="flipper_length_mm", data=penguins)  
>>> plt.show()
```





Histograms (horizontal)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

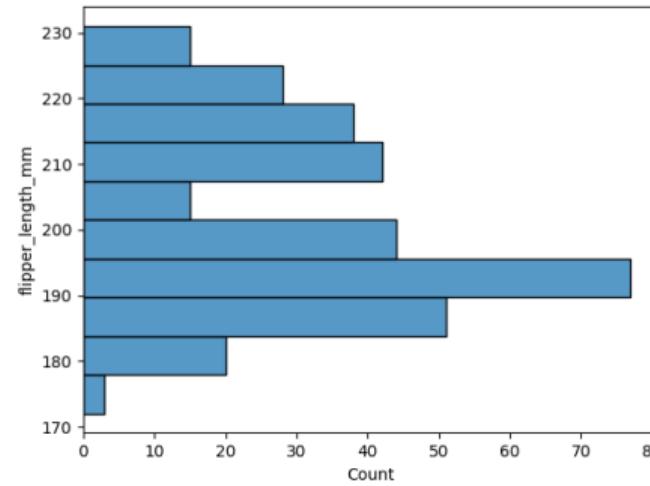
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>  
>>> sns.histplot(y="flipper_length_mm", data=penguins)  
>>> plt.show()
```





Histograms (change bin width)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

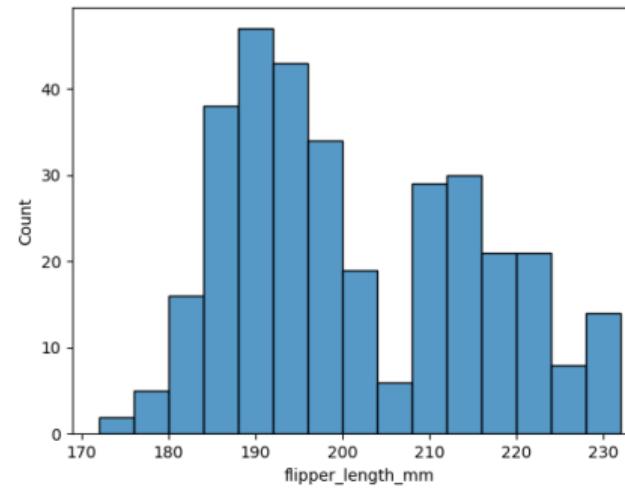
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>  
>>> sns.histplot(x="flipper_length_mm", data=penguins, binwidth=4)  
>>> plt.show()
```





Histograms (change bin number)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

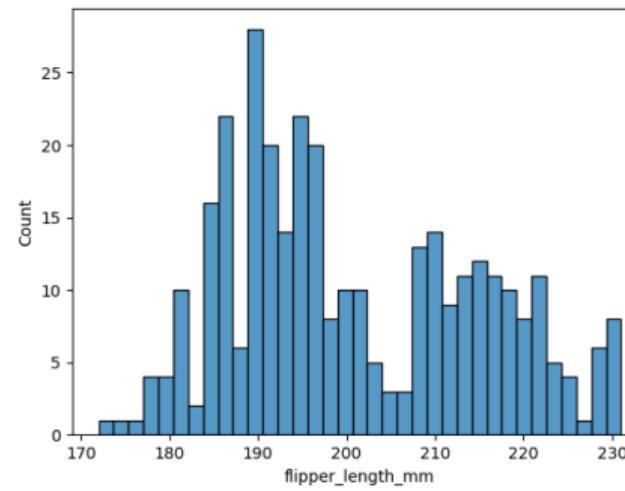
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.histplot(x="flipper_length_mm", data=penguins, bins=35)  
>>> plt.show()
```





Histograms (include pdf)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

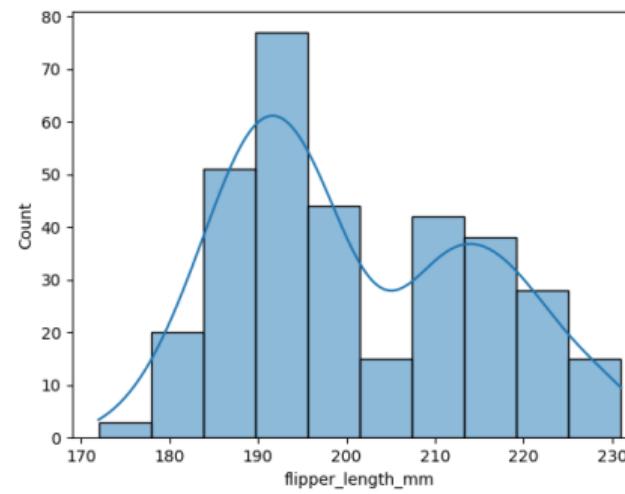
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>  
>>> sns.histplot(x="flipper_length_mm", data=penguins, kde=True)  
>>> plt.show()
```





Histograms (layering)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

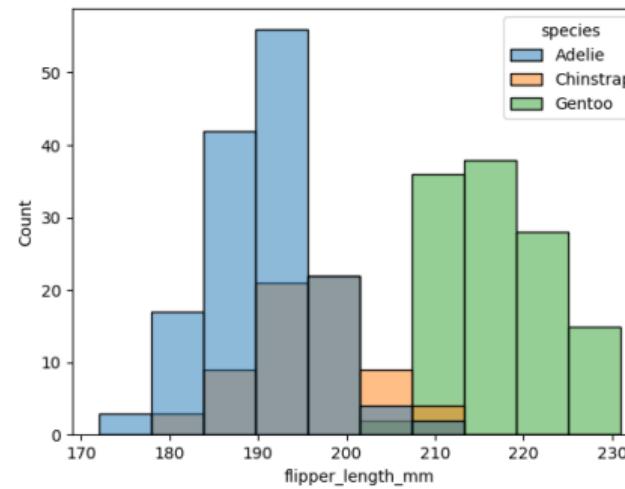
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.histplot(x="flipper_length_mm", hue="species", data=penguins)  
>>> plt.show()
```





Histograms (stacking)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

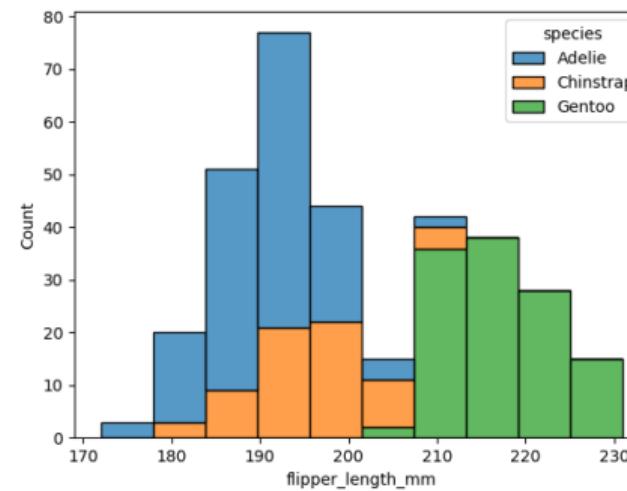
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.histplot(x="flipper_length_mm", hue="species", multiple="stack", data=penguins)
>>> plt.show()
```





Histograms (dodge)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

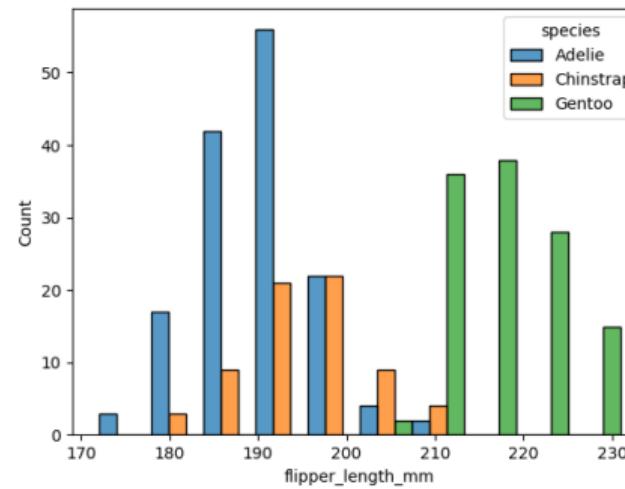
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.histplot(x="flipper_length_mm", hue="species", multiple="dodge", data=penguins)  
>>> plt.show()
```





Histograms (bi-variate)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

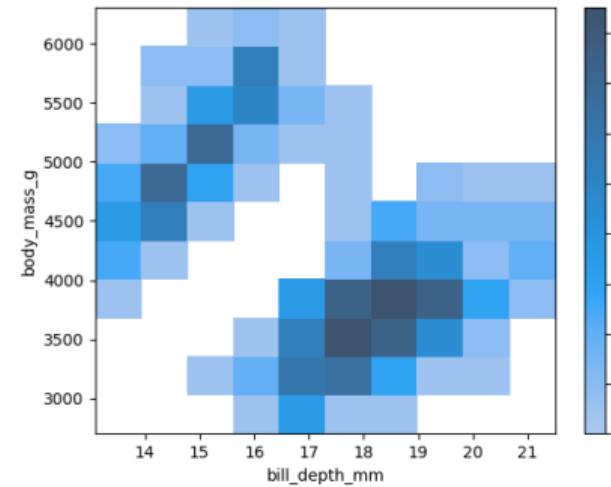
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.histplot(x="flipper_length_mm", y="body_mass_g", cbar=True, data=penguins)
>>> plt.show()
```





Histograms (facets)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

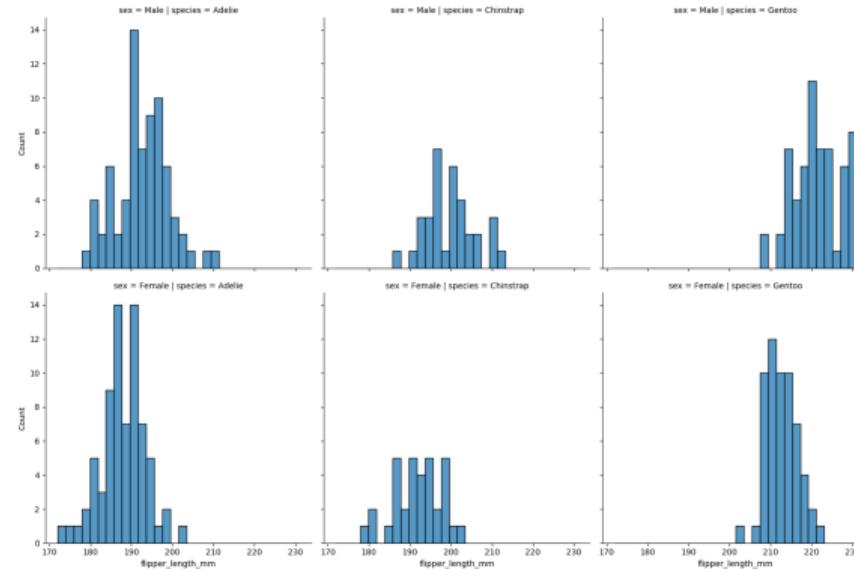
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.displot(x="flipper_length_mm", col="species", row="sex", bins=30, data=penguins)  
>>> plt.show()
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots**
- Stacking Plots
- Regression Plots



Box plots

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

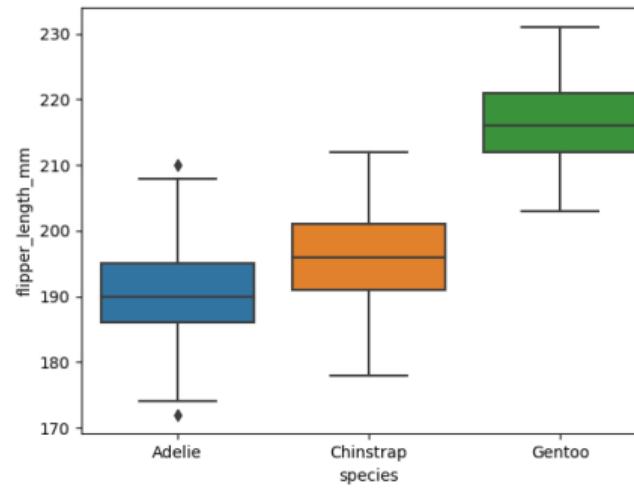
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>  
>>> sns.boxplot(x="species", y="flipper_length_mm", data=penguins)  
>>> plt.show()
```





Box plots (grouped)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

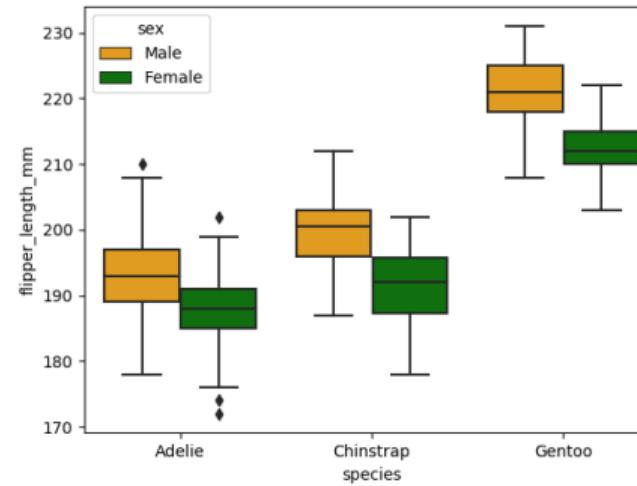
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.boxplot(x="species", y="flipper_length_mm", hue="sex", palette=["orange", "green"], data=penguins)  
>>> plt.show()
```





Violin plots

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

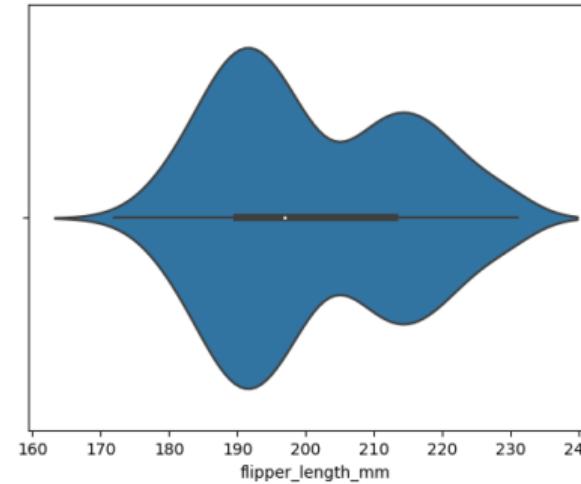
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>  
>>> sns.violinplot(x="flipper_length_mm", data=penguins)  
>>> plt.show()
```





Violin plots

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

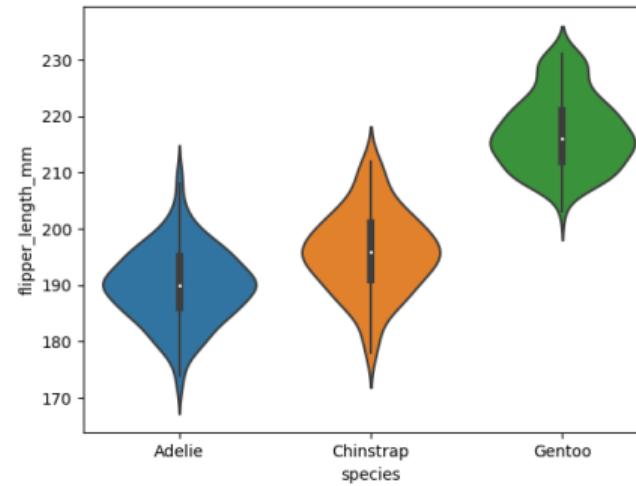
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.violinplot(x="species", y="flipper_length_mm", data=penguins)  
>>> plt.show()
```





Violin plot (grouped)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functions
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

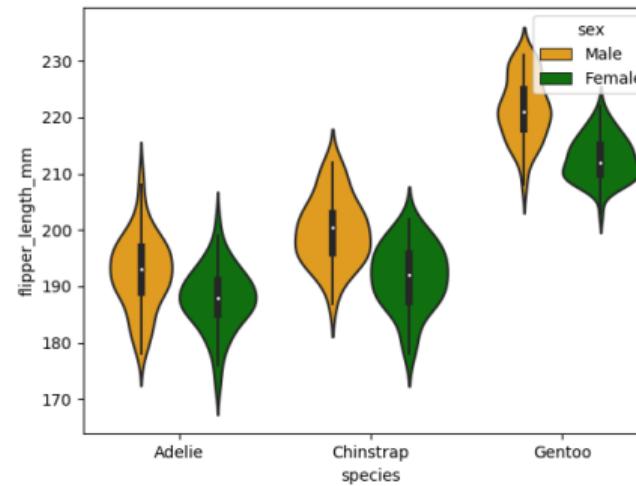
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.violinplot(x="species", y="flipper_length_mm", hue="sex", palette=["orange", "green"], data=penguins)  
>>> plt.show()
```





Categorical plots (scatter)

Data Handling and Manipulation

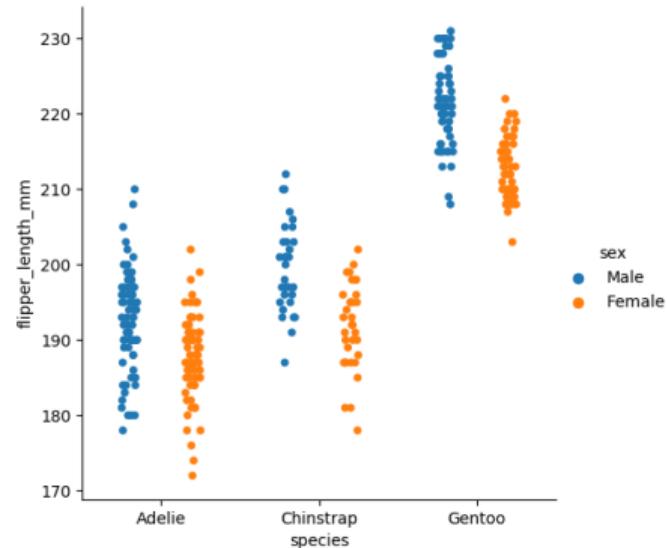
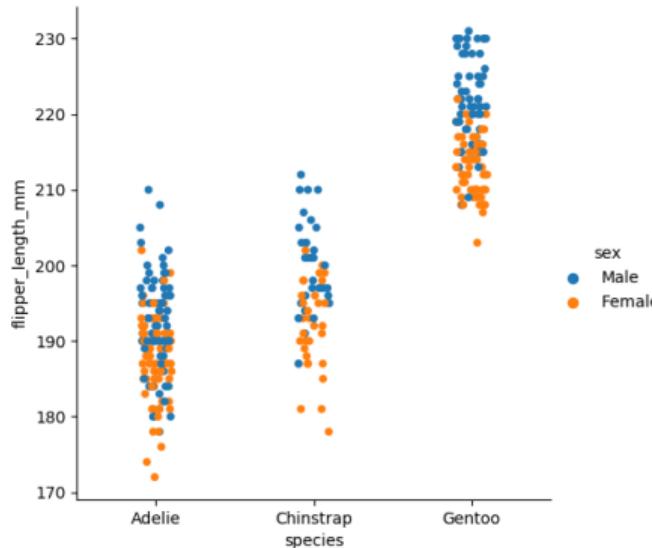
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>  
>>> sns.catplot(x="species", y="flipper_length_mm", hue="sex", data=penguins) #use dodge=True to separate  
>>> plt.show()
```





Categorical plots (box)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

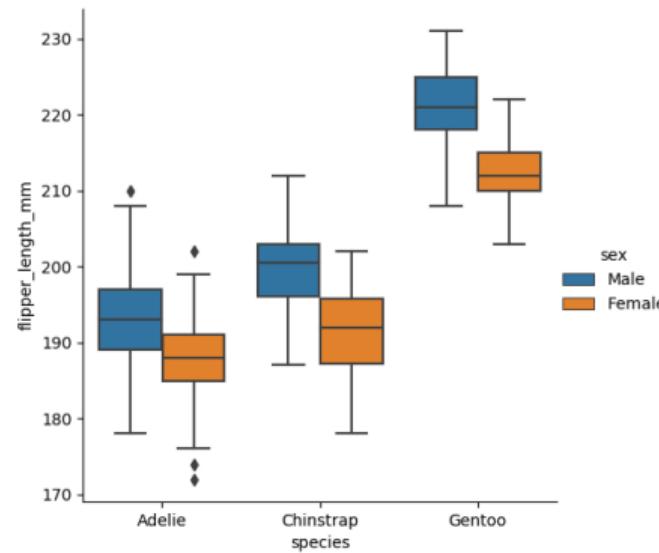
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.catplot(x="species", y="flipper_length_mm", hue="sex", kind="box", data=penguins)  
>>> plt.show()
```





Categorical plots (violin)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

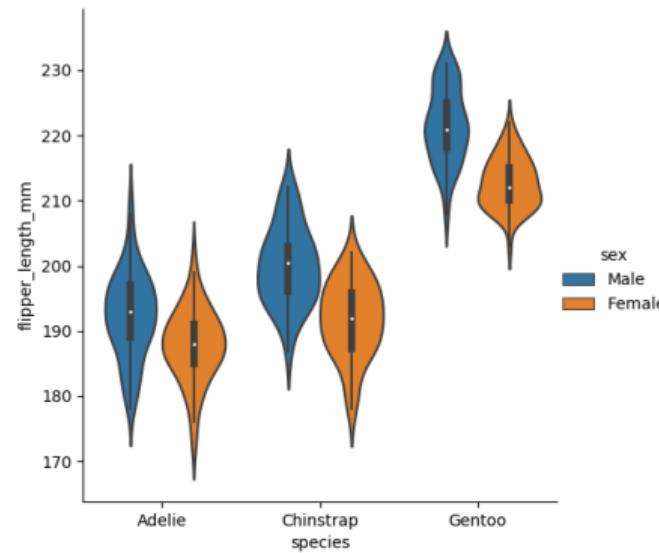
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.catplot(x="species", y="flipper_length_mm", hue="sex", kind="violin", data=penguins)  
>>> plt.show()
```





Categorical plots (bar)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

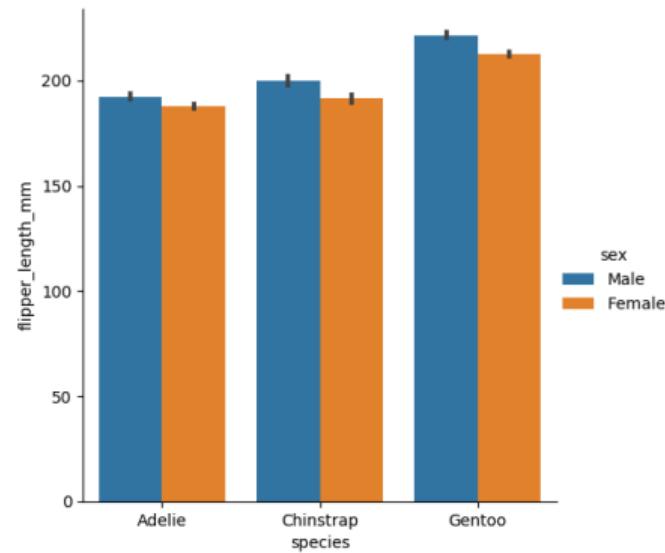
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>>
```

```
>>> sns.catplot(x="species", y="flipper_length_mm", hue="sex", kind="bar", data=penguins)  
>>> plt.show()
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- **Stacking Plots**
- Regression Plots



Stacking plots

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

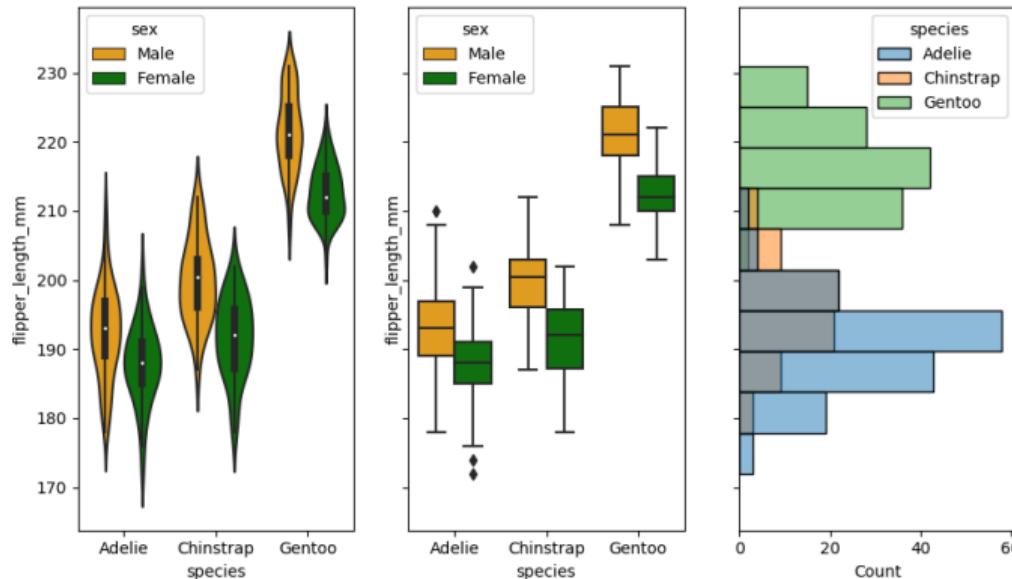
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> f, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(5, 7), sharey=True)
>>> sns.violinplot(x="species", y="flipper_length_mm", hue="sex", palette=["orange", "green"], data=penguins, ax=ax1)
>>> sns.boxplot(x="species", y="flipper_length_mm", hue="sex", palette=["orange", "green"], data=penguins, ax=ax2)
>>> sns.histplot(y="flipper_length_mm", hue="species", data=penguins, ax=ax3)
>>> plt.show()
```





Outline

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

1 Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality
- Indexing and Data Selection
- Grouping and Aggregation
- Merge, join and concatenate
- Reshaping and Pivoting

2 Data Visualization

- Line Plots
- Scatter Plots
- Bar and Distribution Plots
- Box and Violin Plots
- Stacking Plots
- Regression Plots



Linear regression plots (matplotlib)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> from sklearn.linear_model import LinearRegression
>>> penguins = sns.load_dataset("penguins")
>>> penguins.dropna(how='any', inplace=True)
   species      island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g     sex
0   Adelie    Torgersen        39.1          18.7            181.0       3750.0   Male
1   Adelie    Torgersen        39.5          17.4            186.0       3800.0 Female
2   Adelie    Torgersen        40.3          18.0            195.0       3250.0 Female
>>>
>>> new_penguins = penguins[penguins['species']=='Adelie']
>>>
>>> lm = LinearRegression()
>>>
>>> lm.fit(new_penguins['bill_length_mm'].values.reshape(-1,1), new_penguins['bill_depth_mm'].values.reshape(-1,1))
>>>
>>> pred_depth = lm.predict(new_penguins['bill_length_mm'].values.reshape(-1,1))
>>> fig, ax = plt.subplots()
>>> ax.scatter(new_penguins['bill_length_mm'].values.reshape(-1,1), new_penguins['bill_depth_mm'].values.reshape(-1,1),
   color="blue", label='original')
>>> ax.plot(new_penguins['bill_length_mm'].values.reshape(-1,1), pred_depth, color="red", linewidth=2,
   label='predicted')
>>> ax.set_title("Penguins regression")
>>> ax.set_xlabel("Length")
>>> ax.set_ylabel("Depth")
>>> ax.legend()
>>> plt.show()
```



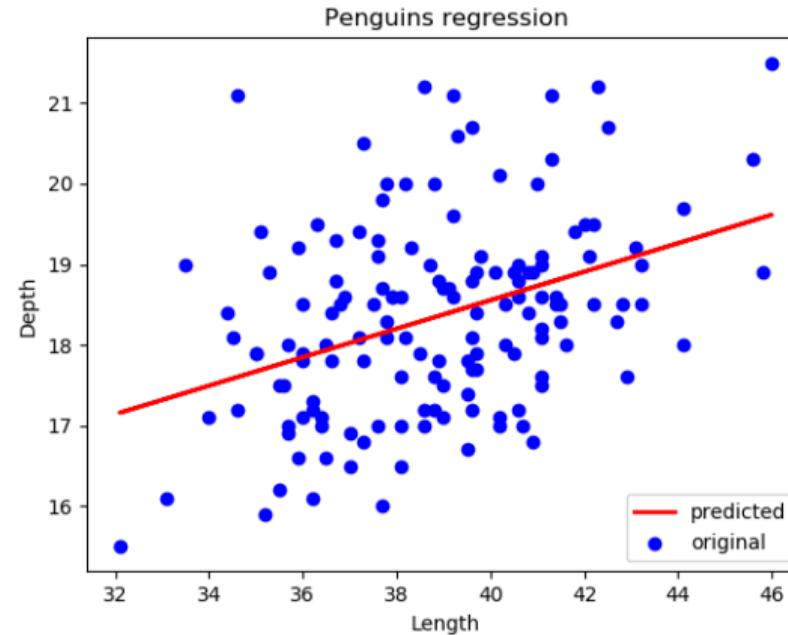
Linear regression plots (matplotlib)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots





Linear regression plots (seaborn)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

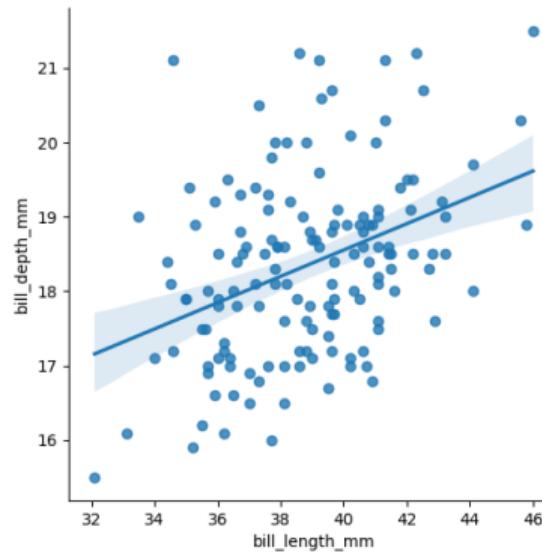
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots

Regression Plots

```
>>> new_penguins = penguins[penguins['species']=='Adelie']
>>> sns.lmplot(x="bill_length_mm", y="bill_depth_mm", data=new_penguins)
>>> plt.show()
```





Linear regression plots (seaborn)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

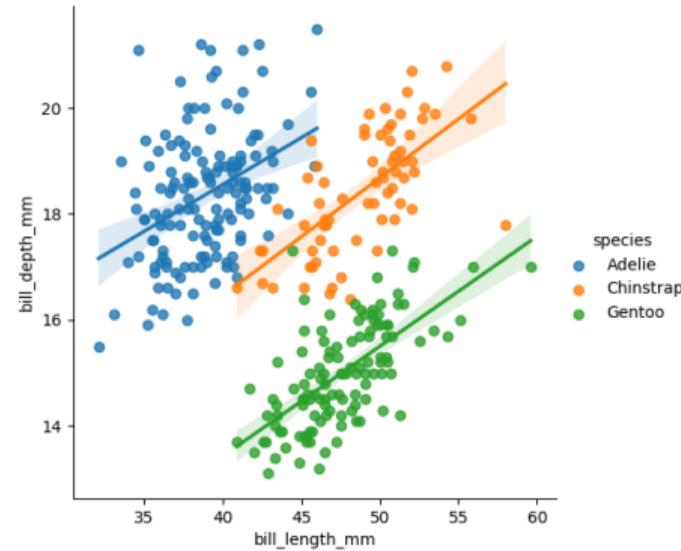
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots

Regression Plots

```
>>>  
>>> sns.lmplot(x="bill_length_mm", y="bill_depth_mm", hue="species", data=penguins)  
>>> plt.show()
```





Linear regression plots (seaborn)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

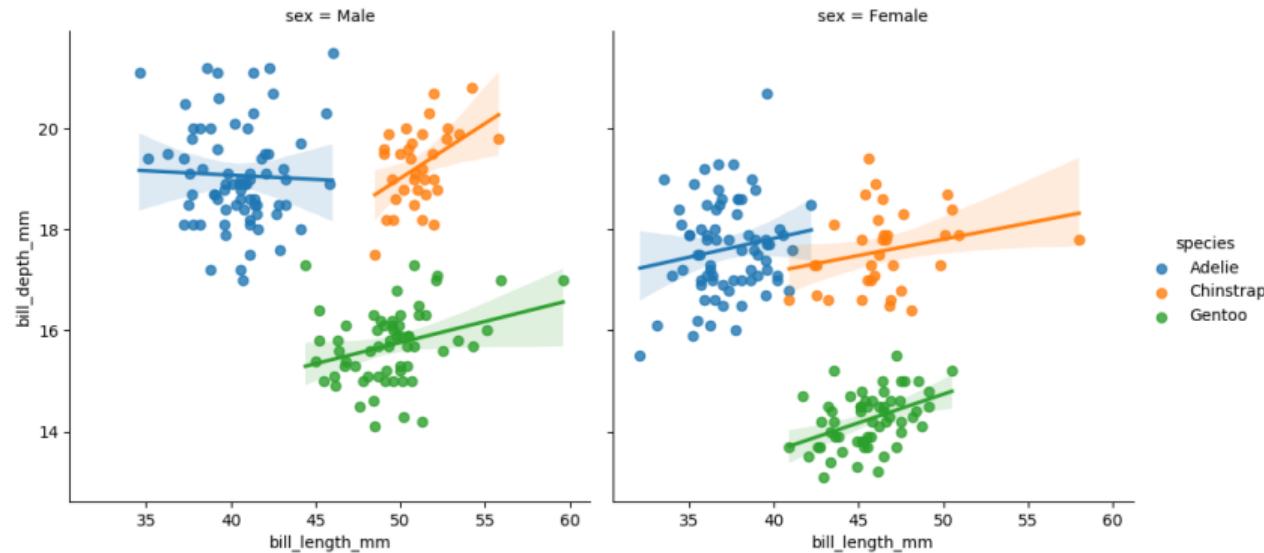
Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots

Regression Plots

```
>>>
```

```
>>> sns.lmplot(x="bill_length_mm", y="bill_depth_mm", hue="species", col="sex", data=penguins)  
>>> plt.show()
```





Linear regression plots (seaborn)

Data Handling and Manipulation

Data Structures in pandas
IO Tools and Essential Functions
Time Series and Date

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

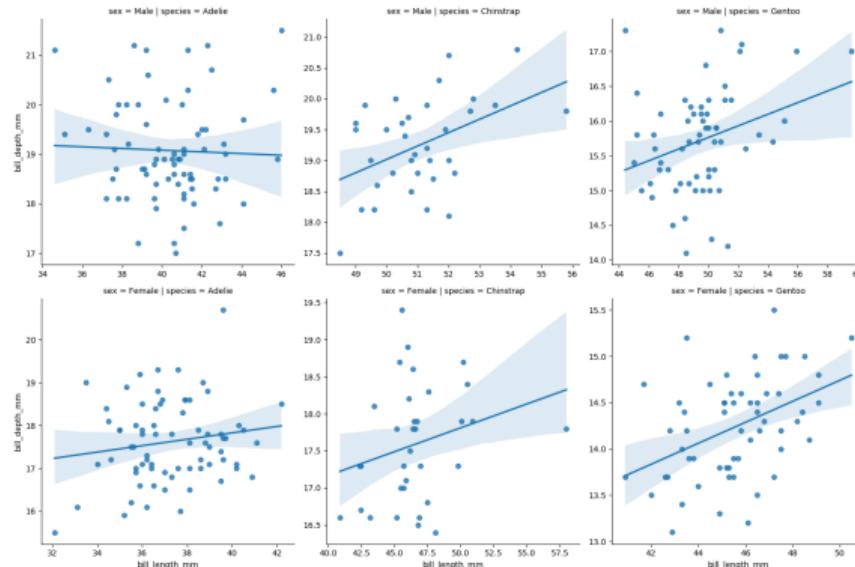
Data Visualization

Line Plots
Scatter Plots

Bar and Distribution Plots
Box and Violin Plots
Stacking Plots

Regression Plots

```
>>>  
>>> sns.lmplot(x="bill_length_mm", y="bill_depth_mm", col="species", row="sex", facet_kws=dict(sharex=False,  
→ sharey=False), data=penguins)  
>>> plt.show()
```

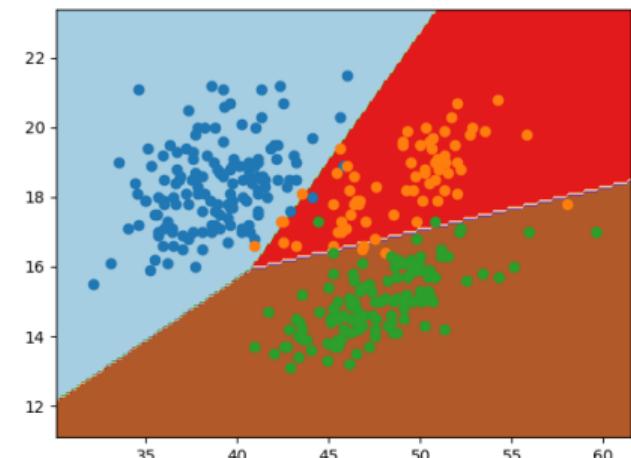




Decision boundaries in bi-variate logistic regression

Data Handling and Manipulation
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality
Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting
Data Visualization
Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> from sklearn.linear_model import
→ LogisticRegression
>>> penguins = sns.load_dataset("penguins")
>>> penguins.dropna(how='any', inplace=True)
>>> X = penguins.iloc[:, [2,3]]
>>> Y = penguins.iloc[:, 0]
>>> colors = {'Adelie':0, 'Chinstrap':1, 'Gentoo':2}
>>> Y = Y.map(colors)
>>> x1grid = np.arange(X.iloc[:, 0].min()-2,
→ X.iloc[:, 0].max()+2, 0.1)
>>> x2grid = np.arange(X.iloc[:, 1].min()-2,
→ X.iloc[:, 1].max()+2, 0.1)
>>> xx, yy = np.meshgrid(x1grid, x2grid)
>>> grid =
→ np.transpose(np.vstack((xx.flatten(),yy.flatten())))
>>> classifier = LogisticRegression()
>>> classifier.fit(X, Y)
>>> yhat = classifier.predict(grid)
>>> zz = yhat.reshape(xx.shape)
>>> plt.contourf(xx, yy, zz, cmap='Paired')
>>> for class_value in range(3):
...     plt.scatter(X[Y == class_value].iloc[:,0],
→ X[Y == class_value].iloc[:,1], cmap='Paired')
>>> plt.show()
```





Decision boundaries in bi-variate logistic regression

Data Handling and Manipulation

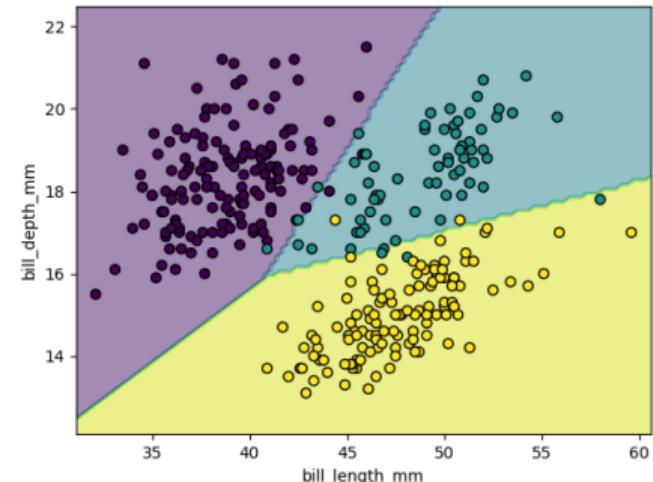
Data Structures in pandas
IO Tools and Essential Functionality
Time Series and Date Functionality

Indexing and Data Selection
Grouping and Aggregation
Merge, join and concatenate
Reshaping and Pivoting

Data Visualization

Line Plots
Scatter Plots
Bar and Distribution Plots
Box and Violin Plots
Stacking Plots
Regression Plots

```
>>> from sklearn.inspection import
→ DecisionBoundaryDisplay
>>> penguins = sns.load_dataset("penguins")
>>> penguins.dropna(how='any', inplace=True)
>>> X = penguins.iloc[:, [2,3]]
>>> Y = penguins.iloc[:, 0]
>>> classifier = LogisticRegression().fit(X, Y)
>>> colors = {'Adelie':0, 'Chinstrap':1, 'Gentoo':2}
>>> fig, ax = plt.subplots()
>>>
→ DecisionBoundaryDisplay.from_estimator(classifier,
→ X, response_method="predict",
→ xlabel=X.columns[0], ylabel=X.columns[1],
→ alpha=0.5, ax=ax)
>>> ax.scatter(X.iloc[:, 0].values.tolist(),
→ X.iloc[:, 1].values.tolist(), c=Y.map(colors),
→ edgecolor="k")
>>> plt.show()
```





Data Handling and Manipulation

- Data Structures in pandas
- IO Tools and Essential Functionality
- Time Series and Date Functionality

Indexing and Data Selection

Grouping and Aggregation

Merge, join and concatenate

Reshaping and Pivoting

Data Visualization

Line Plots

Scatter Plots

Bar and Distribution Plots

Box and Violin Plots

Stacking Plots

Regression Plots

THANK YOU!