Lab 3 - Recap Pandas

> Link: https://pandas.pydata.org/

> Documentation: https://pandas.pydata.org/docs/

Pandas is a library for data analysis with a focus on tabular data. Pandas is a powerful tool that not only provides many useful classes and functions but also does a great job of wrapping functionality from other packages. The result is a user interface that makes data analysis, and in particular financial analysis, a convenient and efficient task.

The DataFrame Class

> Documentation: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.max.html

At the core of pandas is the `DataFrame`, a class designed to efficiently handle data in tabular form — i.e., data characterized by a columnar organization. To this end, the `DataFrame` class provides, for instance, column labeling as well as flexible indexing capabilities for the rows (records) of the data set, similar to a table in a relational database or an Excel spreadsheet.

```
# !pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.
```

```
import pandas as pd
#pd.read_csv('sample_data/california_housing_test.csv')
df = pd.read_excel('Countries.xlsx')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-1124403841.py in <cell line: 0>()
      1 import pandas as pd
      2 #pd.read_csv('sample_data/california_housing_test.csv')
----> 3 df = pd.read_excel('Countries.xlsx')

                                 3 frames
/usr/local/lib/python3.12/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression,
memory_map, is_text, errors, storage_options)
    880         else:
    881             # Binary mode
--> 882             handle = open(handle, ioargs.mode)
    883         handles.append(handle)
    884

FileNotFoundError: [Errno 2] No such file or directory: 'Countries.xlsx'
```

Declare the DataFrame data in the code

```
# Imports pandas
import pandas as pd
# Defines the data as a list object.
df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'], index=['a', 'b', 'c', 'd'])
print(df)
# Add another column
df['countries'] = ['Germany', 'France', 'UK', 'Italy']
print(df)
```

```
   numbers
a       10
b       20
c       30
d       40
   numbers countries
a       10   Germany
b       20    France
c       30        UK
d       40     Italy
```

```
# Declaring a DatFrame without specifying the index
# Imports pandas
```

```
import pandas as pd
# Defines the data as a list object.
df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'])
print(df)
```

```
   numbers
0       10
1       20
2       30
3       40
```

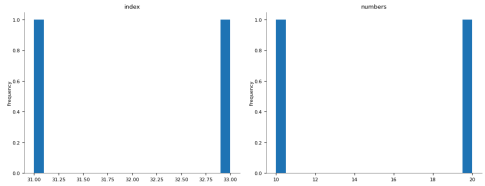
Working with a `DataFrame` object is in general pretty convenient and efficient with regard to the handling of the object, e.g., compared to regular ndarray objects, which are more specialized and more restricted when one wants to (say) enlarge an existing object. At the same time, `DataFrame` objects are often as computationally efficient as ndarray objects. The following are simple examples showing how typical operations on a DataFrame object work.

```
# Imports pandas
import pandas as pd
# Defines the data as a list object.
#df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'], index=['a', 'b', 'c', 'd'])
#df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'], index=['31', '33', '28', '39'])
df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'], index=[31, 33, 28, 39])
df['countries'] = ['Germany', 'France', 'UK', 'Italy']
print(df)
df.loc[33]
df.loc[[31, 33]]
```

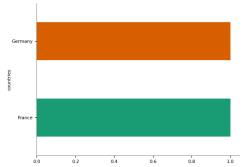
```
    numbers countries
31       10   Germany
33       20    France
28       30        UK
39       40     Italy
```

	numbers	countries
31	10	Germany
33	20	France

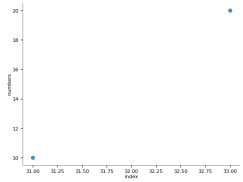
Distributions



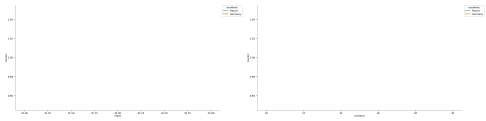
Categorical distributions



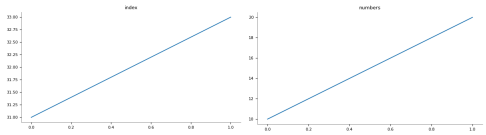
2-d distributions



Time series



Values



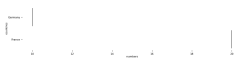
Faceted distributions

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and
```



```
df.columns
```

```
Index(['numbers', 'countries'], dtype='object')
```

```
df = pd.read_csv('sample_data/california_housing_test.csv')
print(df.columns)
print(df.index)
print(df)
#df.T.T
df.loc[1]
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
RangeIndex(start=0, stop=3000, step=1)
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
...          ...         ...            ...                 ...
2995      1258.0       607.0         1.1790            225000.0
2996      3496.0      1036.0         3.3906            237200.0
2997       693.0       220.0         2.2895             62000.0
2998        46.0        14.0         3.2708            162500.0
2999       753.0       260.0         8.5608            500001.0

[3000 rows x 9 columns]
```

	1
longitude	-118.300
latitude	34.260
housing_median_age	43.000
total_rooms	1510.000
total_bedrooms	310.000
population	809.000
households	277.000
median_income	3.599
median_house_value	176500.000

dtype: float64

sum, min, max

```
# Imports pandas
import pandas as pd
# Defines the data as a list object.
df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'])
df['countries'] = ['Germany', 'France', 'UK', 'Italy']
df['numbers2'] = [100, 200, 300, 400]
print(df)
```

```
   numbers countries  numbers2
0       10   Germany       100
1       20    France       200
2       30        UK       300
3       40     Italy       400
```

```
#df.sum()
df.sum(axis=0) # pe coloane
df['numbers'].sum()
#df[['numbers', 'numbers2']].sum(axis=1) # pe linii
```

```
np.int64(100)
```

```
df['numbers'].max()
```

```
40
```

```
import pandas as pd
df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'])
df['numbers2'] = [100, 200, 300, 400]
df['countries'] = ['Germany', 'France', 'UK', 'Italy']
```

```
#df['numbers'] = df['numbers'] + 2 * df['numbers2']# # inlocuim valoarea de pe coloana
df['numbers3'] = df['numbers'] + 2 * df['numbers2']
df
```

	numbers	numbers2	countries	numbers3
0	10	100	Germany	210
1	20	200	France	420
2	30	300	UK	630
3	40	400	Italy	840

```
import pandas as pd
df = pd.read_csv('sample_data/california_housing_test.csv')
# df['total_rooms_not_bedroom'] = df['total_rooms'] - df['total_bedrooms']
# df_total_rooms = df['total_rooms']
# df_total_rooms.max()
df['total_rooms'].max()
coloane_analizate = ['total_rooms', 'total_bedrooms']
df[coloane_analizate].max()
```

	0
total_rooms	30450.0
total_bedrooms	5419.0

dtype: float64

Adding a new row

```
import pandas as pd
df = pd.DataFrame([10, 20, 30, 40], columns=['numbers'])
df['countries'] = ['Germany', 'France', 'UK', 'Italy']
print(df)
df_new_row = pd.DataFrame([[50, 'Spain']], columns=['numbers', 'countries'])
print(df_new_row)
df = pd.concat([df, df_new_row]).reset_index(drop=True)
print(df)
```

```
   numbers countries
0       10   Germany
1       20    France
2       30        UK
3       40     Italy
   numbers countries
0       50     Spain
   numbers countries
0       10   Germany
1       20    France
2       30        UK
3       40     Italy
4       50     Spain
```

Basic Analytics

```
import numpy as np
import pandas as pd

np.random.seed(0) # fix the seed, so we get the same values everytime
array = np.random.random((6, 4))
df = pd.DataFrame(array, columns=['a', 'b', 'c', 'd'])
df
```

	a	b	c	d
0	0.548814	0.715189	0.602763	0.544883
1	0.423655	0.645894	0.437587	0.891773
2	0.963663	0.383442	0.791725	0.528895
3	0.568045	0.925597	0.071036	0.087129
4	0.020218	0.832620	0.778157	0.870012
5	0.978618	0.799159	0.461479	0.780529

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   a       6 non-null      float64
 1   b       6 non-null      float64
 2   c       6 non-null      float64
 3   d       6 non-null      float64
dtypes: float64(4)
memory usage: 324.0 bytes
```

```
df.describe()
```

	a	b	c	d
count	6.000000	6.000000	6.000000	6.000000
mean	0.583835	0.716983	0.523791	0.617204
std	0.359143	0.189711	0.267966	0.303309
min	0.020218	0.383442	0.071036	0.087129
25%	0.454944	0.663218	0.443560	0.532892
50%	0.558429	0.757174	0.532121	0.662706