

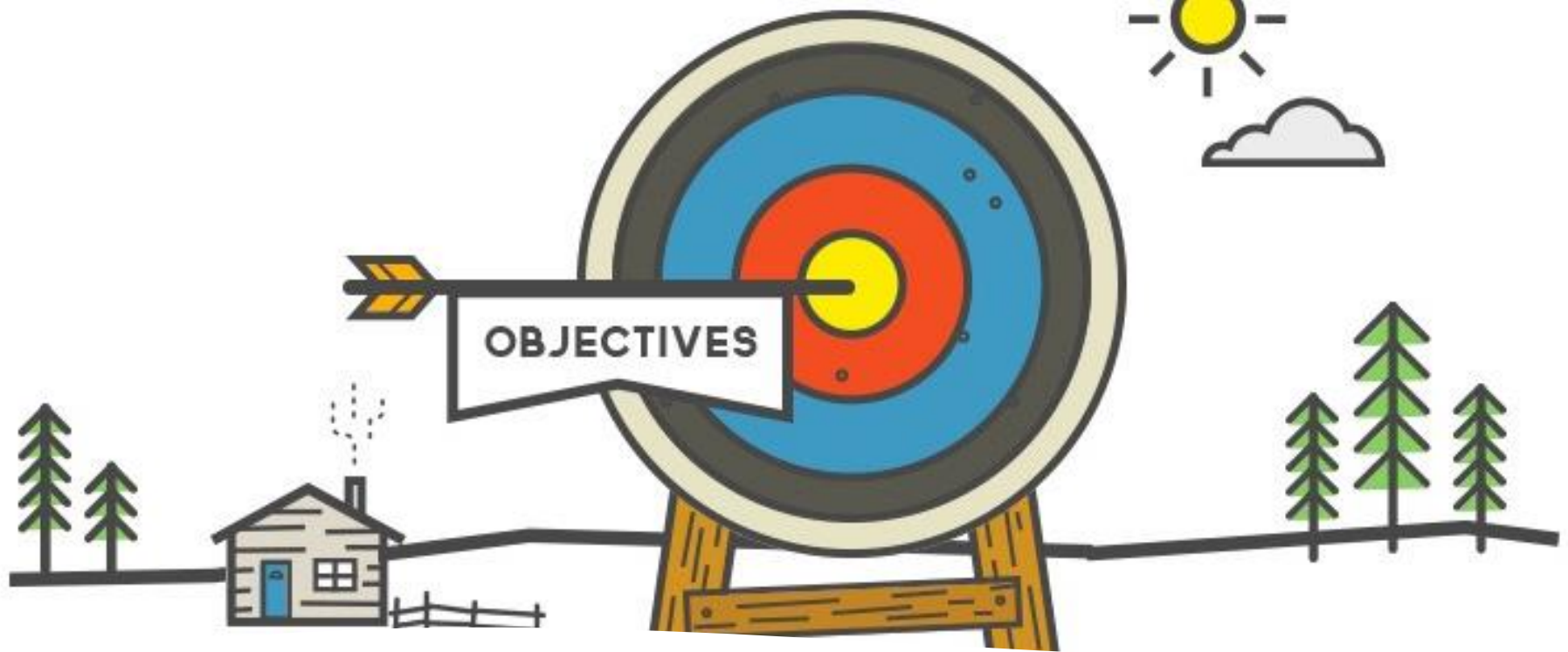


Lisbon School  
of Economics  
& Management  
Universidade de Lisboa



# Pandas

Prof. Carlos J. Costa, PhD



## Learning Goals

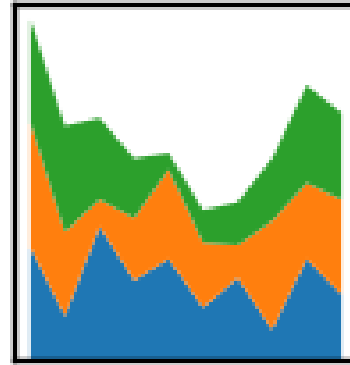
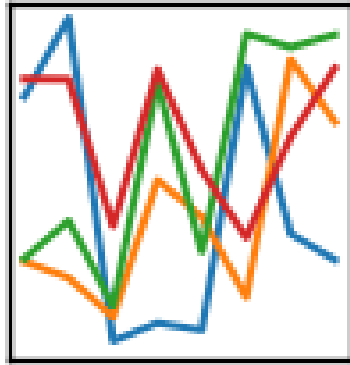
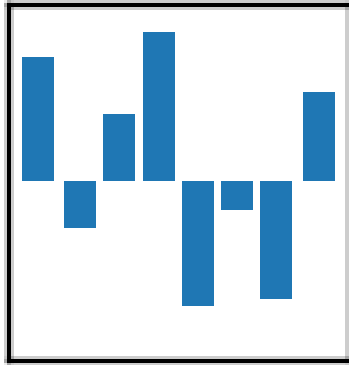
- Understand main characteristics of Pandas
- Manipulate date with Pandas
- Use pandas in the context of data science problems



# Pandas

- <https://pandas.pydata.org/>
- Open source library,
- BSD License
- High performance
- Easy to use
- Includes data structures and data analysis tools

;



# Data Structures

- Series
- DataFrame

# DataFrame

- Labelled data structure
- Columns with potentially different data types
- Similar to spreadsheet or SQL table
- Most used object by Pandas

	Country	Area_km2	Birth rate(births/1000 population)	Current account balance
0	Afghanistan	647500	47.02	NaN
1	Akrotiri	123	NaN	NaN
2	Albania	28748	15.08	-5.040000e+08
3	Algeria	2381740	17.13	1.190000e+10
4	American Samoa	199	23.13	NaN
5	Andorra	468	9.00	NaN
6	Angola	1246700	44.64	-3.788000e+07
7	Anguilla	102	14.26	NaN

# Create DataFrame

- Create dataframe from dictionary

```
import pandas as pd
d = {'col1': [1,2,1,3,1,2], 'col2':
     [1,2,3,4,5,6]}
df = pd.DataFrame(data=d)
df.count()
df['col1'].value_counts()
df['col1'][1]=5
```

# Copy DataFrames

- Copy column

```
col1=df['col1']
```

```
col1[2]=99
```

- What is the result in col1 and df?

```
new_col1 = col1.copy()
```

```
new_col[2]=9999
```

# Read and Save

- Read and save into csv file:

```
import pandas as pd
df = pd.read_csv('worlddata.csv')
...
df.to_csv('worlddata1.csv')
```





# Read and Save

- Read and save into csv file

```
url='https://shorturl.at/fi389'  
df = pd.read_csv(url, error_bad_lines=False, index_col=0, sep=",")
```

---

## Read and Save

- In collaboratoy:  

```
from google.colab import files  
files.upload()
```
- At the end  

```
files.download('file name')
```





# Dataframe Information

- Analyze information
  - `df.head()`
  - `df.info()`
  - `df.describe()`
  - `df.columns`

# Access to Rows and Columns

- DataFrame.at - single value for a pair of row/column labels.
- DataFrame.iloc - integer-location based indexing for selection by position.
- DataFrame.xs returns cross-section from the Series/DataFrame.
- DataFrame.loc allow accessing a group of rows and columns by label(s) or a Boolean array.

# Access to Rows and Columns

```
import pandas as pd
a={'name':["Ann","Ariana","Catarina","João","Patrick"],'address':[13,16,15,13,12]}
df=pd.DataFrame(a)
print(df)
```

```
   name  address
0   Ann        13
1 Ariana        16
2 Catarina       15
3   João        13
4 Patrick        12
```

```
df.at[4, 'name']
```

```
'Patrick'
```

```
df.iloc[4,0]
```

```
0   Ann
1 Ariana
2 Catarina
3   João
Name: name, dtype: object
```

```
df.loc[:3, 'name']
```

```
0   Ann
1 Ariana
2 Catarina
3   João
Name: name, dtype: object
```

```
df.xs(2)
```

```
name      Catarina
address      15
Name: 2, dtype: object
```

```
df.set_index('name',inplace=True)
```

# Access to Row and Columns

- Cells:

```
df.iloc[195][0]
```

- Rows:

```
df.iloc[[195][0]]
```

- Columns:

```
df.loc[:, 'GDPpercapita']
```

# Copy

- Assignment

```
df1=df2
```

- Shallow copy

```
copydf = df.copy(deep=False)
```

- Deep copy

```
copydf = df.copy(deep=True)
```

# Convert Data to Numeric

- Data types

```
df.dtypes
```

- If the result is object, we need to convert a complete column with specific label to numeric

```
df['GDP']=pd.to_numeric(df['GDP'], errors='coerce')
```

```
pd.to_numeric(args, errors)
```



# Convert Data to Numeric

- Suppose you have “,” instead of “.”
- Or you have \$ or € symbols

```
df['GDP'] = df['GDP'].str.replace(',', '.', '')
```

```
df['GDP'] = df['GDP'].replace(['\$', '\€'], '', regex=True)
```

# Create New Columns

- To create a column corresponding to the “internet per capita” it is necessary to do simply:

```
df[ 'IntPC' ]=df[ 'Internet users' ]/df[ 'Pop' ]
```

# Removing Missing Values

- Create a new dataframe

```
YX = df[['GDP', 'MilGDP', 'Unemploy rate(%)']]
```

- And

```
YX.dtypes
```

- All numerical of course



# Removing Missing Values

- Delete missing values from the entire array

```
YX=YX.dropna()
```

- Create X and Y:

```
Y = YX[['GDP']]
```

```
X = YX[['MilGDP','Unemploy rate(%)']]
```



# Remove missing values

- Delete missing values from the entire array

```
YX=YX.dropna()
```

- Create X and Y:

```
Y = YX[['GDPpercapita']]
```

```
X = YX[['MilitPercentGDP','Unemploy rate(%)']]
```



# Statistic Methods

- Using the previous dataframe, the following met
- X.mean()
- X.median()
- X.max()
- X.min()
- X.cov()
- X.corr()
- X.kurt()
- X.kurtosis()
- X.skew()

# Conclusion

- Data structures: dataframe, series
- How to manipulate date
- How to clean and access to data



# Additional Bibliography

- <https://pandas.pydata.org/>
- [https://pandas.pydata.org/pandas-docs/stable/getting\\_started/10min.html](https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html)
- <https://scikit-learn.org/>
- <https://scikit-learn.org/stable/index.html>
- <https://www.statsmodels.org/stable/index.html>