LISBON
SCHOOL OF
ECONOMICS &
MANAGEMENT

UNIVERSIDADE DE LISBOA

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

CARLOS J. COSTA

Pandas

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

Pandas

- **DataFrame**
  - It is a labeled data structure
  - It has columns with potentially different data types
  - Similar to spreadsheet or SQL table
  - It is the most used object by Pandas
  - In addition to the data it is possible to identify columns (column labels) and indexes (row labels)
  -

Pandas

- 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
```

Pandas

- Copy column

```
col1=df['col1’]
col1[2]=99
```

- What is the result in column1 and df?

```
new_col1 = col1.copy()

new_col[2]=9999
```

Pandas

- In collaboratoy:

```
from google.colab import files
files.upload()
```

- At the end

```
files.download(‘nome do ficheiro')
```

Pandas

- On your computer, place the data file in the same folder where the notebook is written
-

Pandas

- Import pandas

```
import pandas as pd
df = pd.read_csv(‘factbook.csv')
```

Pandas

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

Pandas

- DataFrame.loc
- Access a group of rows and columns per label(s) or a Boolean array.
- loc [] is mainly labeled based, but can also be used with a boolean matrix.
-

Pandas

- DataFrame.at
  - Access a single value for a pair of row/column labels.
- DataFrame. iloc
  - access a group of rows and columns per entire position(s).
- DataFrame. Xs
  - Retorna uma seção transversal (linha (s) ou coluna (s)) da série/DataFrame.
- Série. Loc
  - Acede a um grupo de valores utilizando etiquetas.

Pandas

- Cells:

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

- Lines:

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

- Columns:

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

Pandas

- Data types
- `df.dtypes`

- If the result is object there is a need to convert a complete column with specific label to numeric
- `df.loc[:,'GDPpercapita']=pd.to_numeric(df['GDPpercapita'], errors='coerce’)`

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

Can be list, tuple, array, 1D series

Errors can be ignore, raise, or coerce. The latter converts into NAN

Pandas

- Obviously, if necessary also in other variables:
- It can be however

```
df.loc[:,'GDPpercapita']=pd.to_numeric(df['GDPpercapita'], errors='coerce')
df.loc[:,'Military_percent_GDP']=pd.to_numeric(df['Military_percent_GDP'], errors='coerce')
df.loc[:,'Unemployment rate(%)']=pd.to_numeric(df['Unemployment rate(%)'], errors='coerce’)
```

- You can see the result:

```
df.dtypes
```

Pandas

- Create a new dataframe
- `YX = df[['GDPpercapita','Military_percent_GDP','Unemployment rate(%)’]]`
- And
- `YX.dtypes`
- All numerical of course...

Pandas

- Delete missing values from the entire array
-

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

- Create X and Y:
-
```
Y = YX[['GDPpercapita']]
X = YX[['Military_percent_GDP','Unemployment rate(%)']]
```

Pandas

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

```
df['internetpercapita']=df['Internet
users']/df['Population’]
```

Bibliografia

- https://pandas.pydata.org/
- 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