1921 UN SECOLO DI STORIA DAVANTI A NOI 2021

UNIVERSITÀ CATTOLICA del Sacro Cuore

Time Series Analysis and Forecasting

MGO962

Lab 1: Data Manipulation in Python

Main Python packages

pandas

statsmodels

matplotlib

NumPy

SciPy

`Numpy` library

- introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects
- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy

`Scipy` library

- collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Pandas library

- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

Matplotlib library

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization

Loading Python Libraries

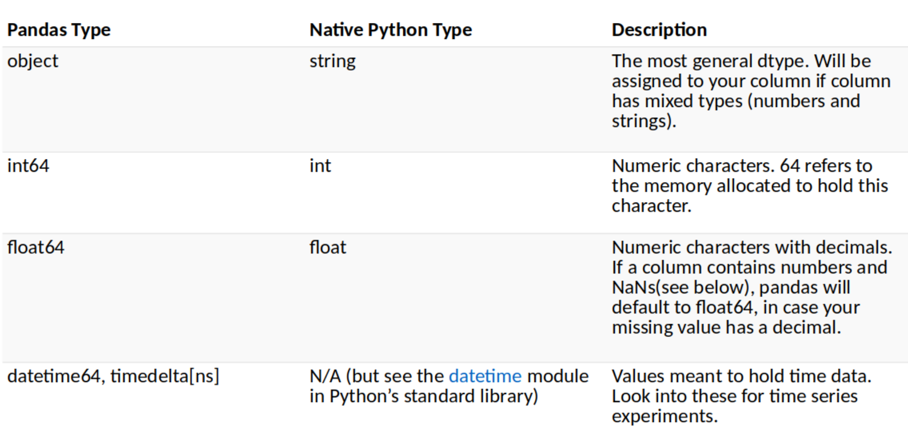
```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
```

Read CSV Files

```
import pandas as pd
df = pd.read_csv("data/tourism.csv", parse_dates=True)
df = df.drop(columns=["Unnamed: 0"])
df.head()
```

```
##       Quarter    Region            State   Purpose       Trips
## 0  1998-01-01  Adelaide  South Australia  Business  135.077690
## 1  1998-04-01  Adelaide  South Australia  Business  109.987316
## 2  1998-07-01  Adelaide  South Australia  Business  166.034687
## 3  1998-10-01  Adelaide  South Australia  Business  127.160464
## 4  1999-01-01  Adelaide  South Australia  Business  137.448533
```

Data Frame Data Types



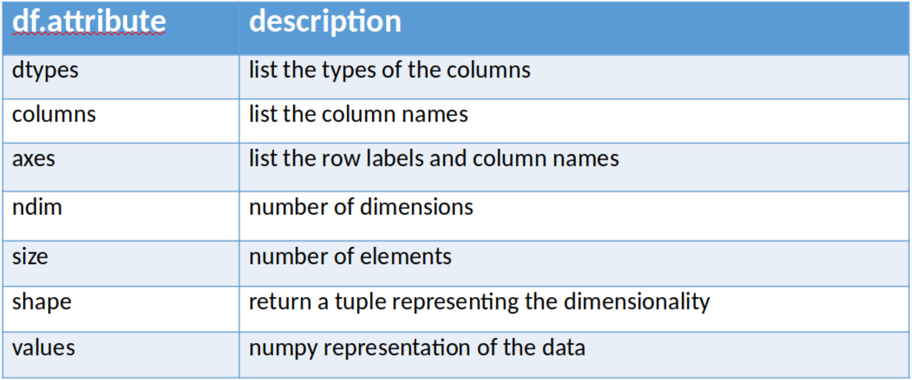
Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python’s standard library)	Values meant to hold time data. Look into these for time series experiments.

Data Frame Data Types

```
import pandas as pd
df = pd.read_csv("data/tourism.csv", parse_dates=True)
df = df.drop(columns=["Unnamed: 0"])
df.dtypes
```

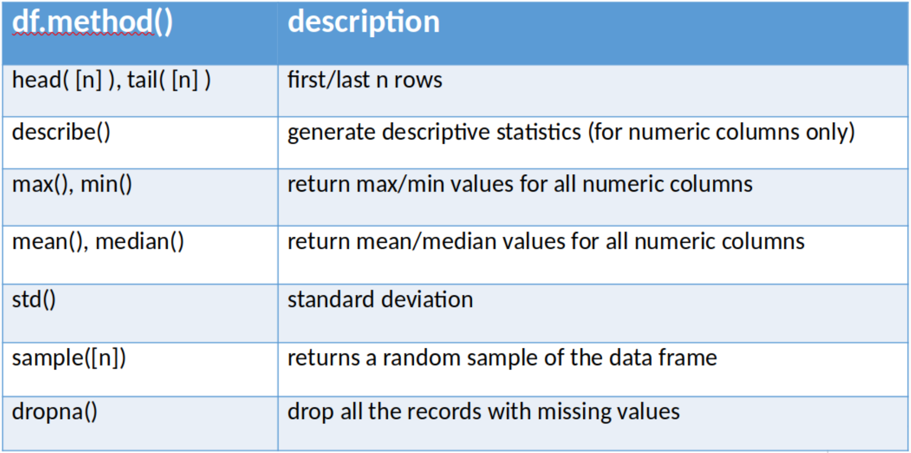
```
## Quarter     object
## Region      object
## State       object
## Purpose     object
## Trips      float64
## dtype: object
```

Data Frame Data Attributes



df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Data Frame Data Methods



df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Selecting a column in a Data Frame 1

Method 1: Subset the data frame using column name:

```
import pandas as pd
df = pd.read_csv("data/tourism.csv", parse_dates=True)
df['State']
```

```
## 0        South Australia
## 1        South Australia
## 2        South Australia
## 3        South Australia
## 4        South Australia
##                ...
## 23403    South Australia
## 23404    South Australia
## 23405    South Australia
## 23406    South Australia
## 23407    South Australia
## Name: State, Length: 23408, dtype: object
```

Selecting a column in a Data Frame 2

Method 2: Use the column name as an attribute:

```
import pandas as pd
df = pd.read_csv("data/tourism.csv", parse_dates=True)
df.State
```

```
## 0        South Australia
## 1        South Australia
## 2        South Australia
## 3        South Australia
## 4        South Australia
##                ...
## 23403    South Australia
## 23404    South Australia
## 23405    South Australia
## 23406    South Australia
## 23407    South Australia
## Name: State, Length: 23408, dtype: object
```

Data Frames `groupby` method 1

Using “groupby” method we can:

- Split the data into groups based on some criteria
- Calculate statistics (or apply a function) to each group
- Similar to dplyr() function in R

Data Frames groupby method 2

```python
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df_state = df.groupby(['rank'])
df_state.mean()
```

```
##                   phd    service         salary
## rank
## AssocProf  15.076923  11.307692   91786.230769
## AsstProf    5.052632   2.210526   81362.789474
## Prof       27.065217  21.413043  123624.804348
```

Data Frames groupby 3

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df_state = df.groupby(['rank'])[['salary']]
df_state.mean()
```

```
##                    salary
## rank
## AssocProf   91786.230769
## AsstProf    81362.789474
## Prof       123624.804348
```

Data Frames groupby 4

groupby performance notes:

- no grouping/splitting occurs until it’s needed. Creating the groupby object only verifies that you have passed a valid mapping
- by default the group keys are sorted during the groupby operation. You may want to pass sort=False for potential speedup:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df_state = df.groupby(['rank'], sort=False)[['salary']]
df_state.mean()
```

```
##                     salary
## rank
## Prof        123624.804348
## AssocProf    91786.230769
## AsstProf     81362.789474
```

Data Frames `filtering` 1

To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter. For example if we want to subset the rows in which the salary value is greater than $120K:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df_sub = df[ df['salary'] > 120000 ]
df_sub.mean()
```

```
## phd              28.8
## service          24.6
## salary       141722.4
## dtype: float64
```

Data Frames `filtering` 2

Any Boolean operator can be used to subset the data:

- “>” greater; >= greater or equal;
- “<” less; <= less or equal;
- “==” equal; != not equal;

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df_f = df[ df['sex'] == 'Female' ]
df_f.mean()
```

```
## phd                16.512821
## service            11.564103
## salary         101002.410256
## dtype: float64
```

Data Frames `slicing` 1

There are a number of ways to subset the Data Frame:

- one or more columns
- one or more rows
- a subset of rows and columns

Rows and columns can be selected by their position or label

Data Frames `slicing` 2

When selecting one column, it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame):

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df['salary']
```

```
## 0     186960
## 1      93000
## 2     110515
## 3     131205
## 4     104800
##        ...
## 73    105450
## 74    104542
## 75    124312
## 76    109954
## 77    109646
```

Data Frames `slicing` 3

When we need to select more than one column and/or make the output to be a DataFrame, we should use double brackets:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df[['rank','salary']]
```

```
##           rank  salary
## 0         Prof  186960
## 1         Prof   93000
## 2         Prof  110515
## 3         Prof  131205
## 4         Prof  104800
## ..         ...     ...
## 73        Prof  105450
## 74   AssocProf  104542
## 75        Prof  124312
## 76        Prof  109954
```

Data Frames `slicing` 4

If we need to select a range of rows, we can specify the range using ":"

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df[10:15]
```

```
##         rank discipline  phd  service   sex  salary
## 10      Prof          B   39       33  Male  128250
## 11      Prof          B   23       23  Male  134778
## 12  AsstProf          B    1        0  Male   88000
## 13      Prof          B   35       33  Male  162200
## 14      Prof          B   25       19  Male  153750
```

Notice that the first row has a position 0, and the last value in the range is omitted: So for 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9

Data Frames `loc` method

If we need to select a range of rows, using their labels we can use method loc:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df.loc[10:20,['rank','sex','salary']]
```

```
##         rank   sex  salary
## 10      Prof  Male  128250
## 11      Prof  Male  134778
## 12  AsstProf  Male   88000
## 13      Prof  Male  162200
## 14      Prof  Male  153750
## 15      Prof  Male  150480
## 16  AsstProf  Male   75044
## 17  AsstProf  Male   92000
## 18      Prof  Male  107300
## 19      Prof  Male  150500
## 20  AsstProf  Male   92000
```

Data Frames `iloc` method

If we need to select a range of rows and/or columns, using their positions we can use method iloc:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df.iloc[10:20,[0, 3, 4, 5]]
```

```
##          rank  service   sex  salary
## 10       Prof       33  Male  128250
## 11       Prof       23  Male  134778
## 12   AsstProf        0  Male   88000
## 13       Prof       33  Male  162200
## 14       Prof       19  Male  153750
## 15       Prof        3  Male  150480
## 16   AsstProf        3  Male   75044
## 17   AsstProf        0  Male   92000
## 18       Prof        7  Male  107300
## 19       Prof       27  Male  150500
```

Data Frames `iloc` method 2

We can sort the data by a value in the column. By default the sorting will occur in ascending order and a new data frame is return.

```
df.iloc[0]  # First row of a data frame
df.iloc[5]  #(i+1)th row
df.iloc[-1] # Last row

df.iloc[:, 0]  # First column
df.iloc[:, -1] # Last column

df.iloc[0:7]       #First 7 rows
df.iloc[:, 0:2]    #First 2 columns
df.iloc[1:3, 0:2]  #Second through third rows and first 2 columns
df.iloc[[0,5], [1,3]]  #1st and 6th rows and 2nd and 4th columns
```

Data Frames sorting method

We can sort the data using 2 or more columns:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
# Create a new data frame sorted by the column Salary
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

```
##         rank discipline  phd  service     sex  salary
## 55  AsstProf          A    2        0  Female   72500
## 23  AsstProf          A    2        0    Male   85000
## 43  AsstProf          B    5        0  Female   77000
## 17  AsstProf          B    4        0    Male   92000
## 12  AsstProf          B    1        0    Male   88000
```

Data Frames sorting method 2

We can sort the data using 2 or more columns:

```
import pandas as pd
df = pd.read_csv("data/salaries.csv")
df_sorted = df.sort_values( by =['service', 'salary'], ascending =
df_sorted.head(10)
```

```
##          rank discipline  phd  service     sex  salary
## 52       Prof          A   12        0  Female  105000
## 17   AsstProf          B    4        0    Male   92000
## 12   AsstProf          B    1        0    Male   88000
## 23   AsstProf          A    2        0    Male   85000
## 43   AsstProf          B    5        0  Female   77000
## 55   AsstProf          A    2        0  Female   72500
## 57   AsstProf          A    3        1  Female   72500
## 28   AsstProf          B    7        2    Male   91300
## 42   AsstProf          B    4        2  Female   80225
## 68   AsstProf          A    4        2  Female   77500
```

Missing Values

Missing values are marked as NaN

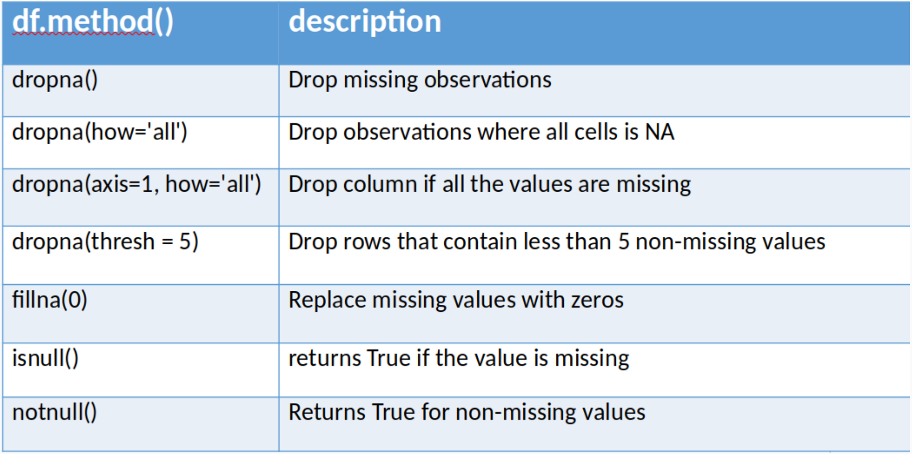
```
import pandas as pd
df = pd.read_csv("data/flights.csv")
```

```
## sys:1: DtypeWarning: Columns (7,8) have mixed types. Specify dt
```

```
df[df.isnull().any(axis=1)].head()
```

```
##    YEAR  MONTH  DAY  ...  AIRLINE_DELAY LATE_AIRCRAFT_DELAY  WE
## 0  2015      1    1  ...            NaN                 NaN
## 1  2015      1    1  ...            NaN                 NaN
## 2  2015      1    1  ...            NaN                 NaN
## 3  2015      1    1  ...            NaN                 NaN
## 4  2015      1    1  ...            NaN                 NaN
##
## [5 rows x 31 columns]
```

Missing Values 2



df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Missing Values 3

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- cumsum() and cumprod() methods ignore missing values but preserve them in the resulting arrays
- Missing values in GroupBy method are excluded (just like in R)
- Many descriptive statistics methods have skipna option to control if missing data should be excluded . This value is set to True by default (unlike R)

Aggregation Functions in Pandas

Aggregation - computing a summary statistic about each group, i.e.

- compute group sums or means
- compute group sizes/counts

Common aggregation functions:

- min, max
- count, sum, prod
- mean, median, mode, mad
- std, var

Aggregation Functions in Pandas 2

agg() method are useful when multiple statistics are computed per column:

```
import pandas as pd
df = pd.read_csv("data/flights.csv")
df[['DEPARTURE_DELAY','ARRIVAL_DELAY']].agg(['min','mean','max'])
```

```
##       DEPARTURE_DELAY  ARRIVAL_DELAY
## min        -82.000000     -87.000000
## mean         9.370158       4.407057
## max       1988.000000    1971.000000
```