Data Analysis and Hypothesis Testing Using the Python ecosystem

pandas

Stavros Demetriadis
Assc. Prof., School of Informatics,
Aristotle University of Thessaloniki

sdemetri@csd.auth.gr
http://mlab.csd.auth.gr/sdemetri

pandas

- What is pandas?
  - Importing pandas
  - The Series and DataFrame objects

- The 'Series' object
  - What is a Series object
  - Constructing a Series object
    - from a list
    - from an array
    - from a dictionary
  - Accessing Series values
  - Series supports vectorized operations

- Indexes and NaN values
  - (1) Handling indexes
    - Setting the index
    - Changing index labels when using a dict
    - rename()
    - reindex()
  - Automatic alignment
  - (2) Handling the 'NaN' (missing) values
    - Adding Series objects with data alignment

pandas

- The 'DataFrame' object
  - What is a DataFrame object
  - Constructing a DataFrame object
  - DataFrame from 'nested' dictionary
- Loading data from external file
  - Import from xls, xlsx
  - Import from csv

- Data access in DataFrame
  - 1. Accessing data by column
    - a) Using column names
    - b) Using column positions
  - 2. Accessing data by row
    - a) Using the .loc indexer for selection by label
    - b) Using the .iloc indexer for selection by position
    - c) Using the .ix indexer
  - Accessing scalars (single data items)
    - Using the [ ] notation
    - Using the .at indexer
    - Using the .iat indexer

Review exercises

- Access data in a pandas DataFrame

- Preparation: run the code below to get your dataframe ready (adjust the path to correctly read the csv or xls file).

```
import pandas as pd
df = pd.read_csv("data/sampledata.csv",sep=',',skiprows=3, header=0, index_col=0)
```

- Work through the following data access exercises. In many of them there are more than one solutions. The answers are available at pytolearn (follow 'Links/Downloads' link).

- ##1. ==== **Accessing by Columns** ==========
- #1.1 Get the first two columns

- #1.2 Get the last two columns

- # 1.3 Get one column by name (write three ways that this can be accomplished)
- # Explore the type of the extracted part in each case

- ## 1.4 Get more than one columns by name
- #What is the type of the extracted data set?

- ## 1.5 Get three or more columns by position
- #Try to slice columns - What happens?

- ## 1.6 Try to slice columns by writing a list comprehension for selecting columns
- # What happens?

- # 2 ==== **.loc** ==========
- ## 2.1 Get one row by label (write two ways to do it and explore the type of the new object)

- ## 2.2 Get two rows by label and explore the type of the new object

- ## 2.3 Slice the dataframe by rows (for example from 'FRA to 'ITA')

- ## 2.4 Slice two rows by name and slice two columns also by name
- ## 2.5 Then produce the same outcome (two rows and two columns) without slicing
- ## 3 ==== **.iloc** ==========
- ## 3.1 Get the odd indexed columns(1,3,5..)
- ## 3.2 Slice the odd position rows and the even position columns
- ## 3.3 Slice rows by label (for example odd indexed labels) and columns by position (even)
- ## 4 ===== **[]** ==========
- ## 4.1 Access a single item without any []
- ## 4.2 Access a single item with []

- ## 5 ===== **.at** ==========
- ## 5.1 Access a single item with row - column labels

- ## 6 ===== **.iat** ==========
- ## 6.1 Access a single item with row - column indexes

- ## 6.2 Use len() to get the number of columns
- #Then get a single item from the last column using row - column indexes

- ## 6.3 Use len() to get the number of columns
- # Use again len() to get the number of rows
- # Then get the item in the intersection of last row/column using the previous values as indexes

- ## 7 ======== **Combining first-column then-row indexers** ==========
- ## 7.1 In column 'Country' get the value in the first row

- ## 7.2 In column '2010' get the value in the last row

- ## 7.3 In the last column get the item for the row 'ESP‘

- ## 8 ======== **Get the index when a value is known** ==========
- ## 8.1 Which are the indexes of the df DataFrame?

- ## 8.2 Which are the indexes for column '2010'?

- ## 8.3 Which is the index for the Country with name 'Greece'?

- ## 8.4 Which country has population 4586897 in the year '2012'?

- ## 8.5 Which country has population 4586897 and in which year?
- #Hint: Iterate over columns and rows to locate value 4586897

- ## 9 ====== **Applying Functions** ==========
- ## 9.1 Which country has the highest population in 2006?

- ## 9.2 In which year did Spain had the highest population?

- ## 9.3 Compute the mean for all rows and all columns using the 'axis' property
- # explore the type of the new object constructed

- ## 9.4 Apply the scipy.stats describe() function in various rows to get some descriptive statistics

- Modifying Columns in DataFrame
  - 1. Rename columns
  - 2. Add columns
  - 3. Delete columns
  - 4. Insert/Rearrange columns
  - 5. Replace column contents

- Modifying Rows in DataFrame
  - 1. Adding Rows and Concatenating DataFrames

Review exercises

- Modifying Columns and Rows

- **‘Modify by Columns’ Preparation**: run the code below to get your dataframe ready (adjust the path to correctly read the csv or xls file).

```
import pandas as pd
df = pd.read_csv("data/sampledata.csv",sep=',',skiprows=3, header=0, index_col=0)
```

- Work through the following Column modifying exercises. The answers are available at pytolearn (follow ‘Links/Downloads’ link).

- ## 1) **RENAME**:
- ## 1.1 Rename column 'Country' to 'Country Name'

- ## 2) **ADD**:
- ## 2.1 Add a new column at the end with the average population for each country.
- ## Name this column as 'Average'

- ## 3) **DELETE**:
- ## 3.1 Delete first column with label "Country Name"

- ## 4) **INSERT**:
- ## 4.1 Construct a new list (or Series with same labels) having in each position the highest value of population for the country. Insert this object as the FIRST column n the dataframe.
- ## Label the column as 'Max Popu‘

- ## 5) **REARRANGE**:
- ## 5.1 Move the first column ('Max Popu') so that it becomes the last in the DataFrame

- ## 6) **REPLACE**:
- ## 6.1 Increase the values in '2006' column by 10%

- **'Modify by Rows' Preparation**: run the code below to get your dataframe ready (adjust the path to correctly read the csv or xls file).

```
import pandas as pd
df = pd.read_csv("data/sampledata.csv",sep=',',skiprows=3, header=0, index_col=0)
```

- Work through the following Row modifying exercises. The answers are available at pytolearn (follow 'Links/Downloads' link).

- ## 1) ADD & CONCATENATE ==========
- *Step-1*) Create a **dfadd** dataframe by slicing the last two columns of df ('2014' and '2015'). Then rename these columns so that '2014 becomes '2016' and '2015' becomes '2017'.
- *Step-2*) Use **.loc** to add a new line to dfadd as follows:
    - dfadd.loc['ALB']=[2780000,2900000]
- *Step-3*) Slice the first two columns of dfnew and construct the **dfnewp**
- *Step-4*) Finally **concatenate dfnewp with dfadd** (first along axis=1 and then along axis=0). What is your conclusion?

- Plotting a Series/DataFrame object
  - 1. Plot Series object
  - 2. Plot DataFrame object
    - Extract and plot one Column data
    - Extract and plot one Row data
    - Plotting DataFrame Column vs. Column

pandas

- The power of 'GroupBy'
  - What is 'GroupBy'?
  - Exploring 'groupby'
    - useful methods of 'groupby' object
  - Retrieving groups
  - Aggregate: applying functions to grouped data
  - GroupBy with 2 factors

Review exercises

- Groupby

- **‘Modify by Rows’ Preparation**: run the code below to get your dataframe ready (adjust the path to correctly read the csv or xls file).

```
import pandas as pd
df = pd.read_csv("data/sampledata.csv",sep=',',skiprows=3, header=0, index_col=0)
```

- Work through the following Row modifying exercises. The answers are available at pytolearn (follow ‘Links/Downloads’ link).

- Step-1) Grouping by 4 groups and use describe() to get some descriptive stats (as shown in the pytolearn example)
- Step-2) Iterate (for loop) over the 4 groups to present their names and data distribution
- Step-3) Use get_group() to get the two boys groups (call them ‘boysK6’ and ‘boysK9’
- Step-4) Apply the aggregate method to get mean and std over all groups
- Step-5) Access a specific stat measure (for example std) through multi-indexing both in Columns and Rows