ECOM90025 Advanced Data Analysis - Pandas Basic

Zheng Fan

July 31, 2023

1 Basic commands for pandas

- Just like a library in R, “pandas” is a popular open-source data manipulation and analysis library. It provides easy-to-use data structures and functions for efficiently working with structured data.
- A 10-minuts to pandas, covering nearly everything we need. https://pandas.pydata.org/docs/user_guide/10min.html
- Ask ChatGPT

To use pandas in your Python environment, you need to install it first using the following command: - remove ! in front to install

[1]:

```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\anaconda3\lib\site-packages (1.2.4)
Requirement already satisfied: pytz>=2017.3 in c:\anaconda3\lib\site-packages
(from pandas) (2021.1)
Requirement already satisfied: numpy>=1.16.5 in c:\anaconda3\lib\site-packages
(from pandas) (1.20.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\anaconda3\lib\site-
packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\anaconda3\lib\site-packages (from
python-dateutil>=2.7.3->pandas) (1.15.0)
```

Once installed, you can import it in your Python scripts or Jupyter notebooks using:

[2]:

```
import pandas as pd
```

- pd is therefore just a shortcut pandas, you can choose what you like.
- we can use it to make a simple data frame

[3]:

```
df_sample = pd.DataFrame(
    {
        "Var_1": 1.0,
        "Var_2": pd.Categorical(["test", "train", "test", "train"]),
        "Var_3": "foo",
    }
)
print(df_sample)
```

```
   Var_1  Var_2 Var_3
0    1.0   test   foo
1    1.0  train   foo
2    1.0   test   foo
3    1.0  train   foo
```

We need to check the data type before doing any numerical exercise

- check python data type here: https://www.w3schools.com/python/python_datatypes.asp

```
[4]: df_sample.dtypes
```

```
[4]: Var_1      float64
     Var_2     category
     Var_3       object
     dtype: object
```

Check data dimension

```
[5]: # the dimension of the data set
     df_sample.shape
```

```
[5]: (4, 3)
```

For illustration purpose, I will use a publicly available dataset from a online Monash python class

```
[6]: df = pd.read_csv("https://monashdatafluency.github.io/python-workshop-base/
      ↪modules/data/surveys.csv")
```

Viewing data

```
[7]: df.head(2)
```

```
[7]:    record_id  month  day  year  site_id species_id sex  hindfoot_length  \
     0          1      7   16  1977        2         NL   M             32.0
     1          2      7   16  1977        3         NL   M             33.0

        weight
     0     NaN
     1     NaN
```

```
[8]: df.tail(2)
```

```
[8]:        record_id  month  day  year  site_id species_id  sex  hindfoot_length  \
     35547      35548     12   31  2002        7         DO    M             36.0
     35548      35549     12   31  2002        5        NaN  NaN              NaN
```

```
       weight
35547    51.0
35548     NaN
```

[9]:

```
df.columns
```

[9]:

```
Index(['record_id', 'month', 'day', 'year', 'site_id', 'species_id', 'sex',
       'hindfoot_length', 'weight'],
      dtype='object')
```

Shows a quick statistic summary of your data:

[10]:

```
df.describe()
```

[10]:

```
          record_id         month           day          year       site_id  \
count  35549.000000  35549.000000  35549.000000  35549.000000  35549.000000
mean   17775.000000      6.474022     16.105966   1990.475231     11.397001
std    10262.256696      3.396583      8.256691      7.493355      6.799406
min        1.000000      1.000000      1.000000   1977.000000      1.000000
25%     8888.000000      4.000000      9.000000   1984.000000      5.000000
50%    17775.000000      6.000000     16.000000   1990.000000     11.000000
75%    26662.000000      9.000000     23.000000   1997.000000     17.000000
max    35549.000000     12.000000     31.000000   2002.000000     24.000000

       hindfoot_length        weight
count     31438.000000  32283.000000
mean         29.287932     42.672428
std           9.564759     36.631259
min           2.000000      4.000000
25%          21.000000     20.000000
50%          32.000000     37.000000
75%          36.000000     48.000000
max          70.000000    280.000000
```

Selecting a single column, which yields a Series

[11]:

```
df["weight"]
```

[11]:

```
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
         …
35544      NaN
35545      NaN
```

```
35546    14.0
35547    51.0
35548     NaN
Name: weight, Length: 35549, dtype: float64
```

Get some rows

```
[12]: df[0:3]
```

```
[12]:    record_id  month  day  year  site_id species_id sex  hindfoot_length  \
      0          1      7   16  1977        2         NL   M             32.0
      1          2      7   16  1977        3         NL   M             33.0
      2          3      7   16  1977        2         DM   F             37.0

         weight
      0     NaN
      1     NaN
      2     NaN
```

Selecting by label

```
[13]: df.loc[:, ["day", "site_id"]]
```

```
[13]:        day  site_id
      0       16        2
      1       16        3
      2       16        2
      3       16        7
      4       16        3
      …     …       …
      35544   31       15
      35545   31       15
      35546   31       10
      35547   31        7
      35548   31        5

      [35549 rows x 2 columns]
```

Performing a descriptive statistic:

```
[14]: df.mean()
```

```
[14]: record_id          17775.000000
      month                  6.474022
      day                   16.105966
      year                1990.475231
```

```
site_id                11.397001
hindfoot_length        29.287932
weight                 42.672428
dtype: float64
```

[15]:

```
df.mean(1) # calculate row mean
```

[15]:

```
0          339.166667
1          339.666667
2          340.333333
3          341.166667
4          340.500000
              …
35544    7521.000000
35545    7521.200000
35546    5375.857143
35547    5383.857143
35548    7519.800000
Length: 35549, dtype: float64
```

[16]:

```
df.month.mean()
```

[16]:

```
6.474021772764353
```