Homework 1 - Grant Jackson

September 6, 2024

1 Introduction to Python

1.0.1 Jupyter notebook

https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

- Each box is called a cell; if click a cell - a green line around the cell appears, which means you are activating/working on the cell; if you run the cell - a number will appear on the left side of the cell, which is the execution number telling the order of executions, which can be useful when working on complex coding
- To start, set up a working directory to the folder where your datasets are stored
  - This will let your program knows where your datasets are
  - The figures, etc that are produced from your program will be stored in the folder as well
- Python is a modular program, which means it consists of many different modules
  - If you want to use a function, you have to import the module containing the function
  - It takes time to figure out what modules include what functions

[1]:
```
import os
os.chdir('C:\\Users\gmoor\Documents\Economic Analytics 1\Data')
# The path to a folder can be found by right-clicking the folder (window:␣
↪"Properties"; Mac: "Get Info")
# Be careful the difference: "/" (Mac) and "\" (Window)
```

[2]:
```
# Install modules and packages
import numpy as np # module for data manipulation (similar to matlab)
import pandas as pd # module for data manipulation (similar to stata)
import math # math fns
```

[3]:
```
# Load data in pandas dataframe using pd.read_csv (data.csv will be uploaded to␣
↪your workspace)
# csv file contains data with data points separated by commas, taking up less␣
↪space than other file formats
# read_csv contains many parameters: https://pandas.pydata.org/docs/reference/
↪api/pandas.read_csv.html
# It's a good habit to check the manual of a new function!

raw0 = pd.read_csv('Police.csv')
```

```
# "=" assignment operator - most functions in python create a temportary object␣
↪when executed
# In order to put the object permanently in your program you have to assign it␣
↪to a variable
# "==" identity operator
```

1.0.2 Differences between Numpy and Pandas

https://discuss.codecademy.com/t/what-are-some-differences-between-pandas-numpy-and-matplotlib/354475

- The biggest difference is that Panda creates and uses tabular data ("Pandas DataFrame" table form -> user friendly) that can be easily visualized. However, since it accompanies many built-in functions (e.g. row, column names), it may be very slow in optimization/computation. In contrast, Numpy creates and uses very simple numerical arrays (e.g. vector and matrix), which are superior in mathmatical/numerical computation.
- We will learn how to convert panda dataframe to numpy array using raw0.values, and see the differences between the two in accessing and manipulating data.

[4]:
```
# Check how data is uploaded using .head()
# The observation number (row number) is not part of data
# Python index starts from 0 (0-indexed)
raw0.head()
```

[4]:
```
         date        census person_gender person_race  person_dob  traffic  \
0  2008-04-30  3.606700e+10             M           W  1981-06-13      1.0
1  2007-01-23  3.606700e+10             M           B  1960-03-27      1.0
2  2009-01-22  3.606700e+10             M           B  1968-10-20      1.0
3  2006-03-22  3.606701e+10             M           B  1982-09-18      1.0
4  2007-07-27  3.606700e+10             F           W  1981-08-30      1.0

   frisk_search  arrest  male  offid       dob_1 race_1 sex_1  apptdate_1  \
0             0       0     1      1  1978-02-07      W     M  2005-07-25
1             0       0     1      1  1978-02-07      W     M  2005-07-25
2             0       0     1      1  1978-02-07      W     M  2005-07-25
3             0       0     1      1  1978-02-07      W     M  2005-07-25
4             0       0     0      1  1978-02-07      W     M  2005-07-25

   blacktract
0         1.0
1         1.0
2         1.0
3         1.0
4         1.0
```

[5]:
```
# Converting a panda frame to a numpy array
raw0.values
```

[5]:
```
array([['2008-04-30', 36067003000.0, 'M', …, 'M', '2005-07-25', 1.0],
       ['2007-01-23', 36067003000.0, 'M', …, 'M', '2005-07-25', 1.0],
       ['2009-01-22', 36067003800.0, 'M', …, 'M', '2005-07-25', 1.0],
       …,
       ['2008-11-14', 36067002200.0, 'M', …, 'M', '1997-11-14', 1.0],
       ['2009-01-20', 36067003200.0, 'F', …, 'M', '1985-09-16', 1.0],
       ['2006-01-24', 36067003200.0, 'M', …, 'M', '1989-03-31', 1.0]],
      dtype=object)
```

[6]:
```
# Drop/remove the rows (observations) containing any missing values
# In practice, how to take care of missing values is not simple
raw0=raw0.dropna()

# Important to remember: check the last row number after implementing dropna()␣
 ↪function
# You will see that row numbers don't change even though some of the rows were␣
 ↪dropped
# The row number (or row index) in panda frame is permanent!
```

[7]:
```
# check the length of data (= n)
# what
len(raw0)
```

[7]:
```
99278
```

[8]:
```
# check the shape of data (n,p)
raw0.shape
```

[8]:
```
(99278, 15)
```

[9]:
```
raw0.tail()
```

[9]:
```
              date        census person_gender person_race  person_dob  \
102505  2008-08-28  3.606700e+10             M           W  1961-12-01
102506  2006-10-21  3.606701e+10             F           W  1959-09-05
102507  2008-11-14  3.606700e+10             M           B  1952-11-03
102508  2009-01-20  3.606700e+10             F           W  1986-06-07
102509  2006-01-24  3.606700e+10             M           W  1950-04-25

        traffic  frisk_search  arrest  male  offid       dob_1 race_1 sex_1  \
102505      1.0             0       0     1    518  1958-08-22      W     M
102506      1.0             0       0     0    519  1954-10-28      B     F
102507      1.0             0       0     1    520  1966-04-09      W     M
102508      1.0             0       0     0    521  1960-11-02      W     M
```

```
102509      1.0             0       0     1    522  1964-03-23      W     M

        apptdate_1  blacktract
102505  1993-12-16         1.0
102506  1981-12-16         0.0
102507  1997-11-14         1.0
102508  1985-09-16         1.0
102509  1989-03-31         1.0
```

1.0.3 Datatypes

https://pbpython.com/pandas_dtypes.html

https://docs.python.org/3/tutorial/floatingpoint.html

- Most frequently used data types are integer, float (similar to decimal number; see the second reference for some issues with float) and object (including string).
- Each type takes different space in computer and some function only work for specific types, so sometimes we need to change the type of data.

[10]:
```python
# check the datatypes of data
raw0.dtypes
```

[10]:
```
date              object
census           float64
person_gender     object
person_race       object
person_dob        object
traffic          float64
frisk_search       int64
arrest             int64
male               int64
offid              int64
dob_1             object
race_1            object
sex_1             object
apptdate_1        object
blacktract       float64
dtype: object
```

[11]:
```python
# change data types (census, traffic and blacktract to integer)
# astype() creates a copy, so the copy should be assigned to the original␣
 ↪variable for the change to be effective
raw0.census = raw0.census.astype(int)
raw0['traffic'] = raw0.traffic.astype(int)
raw0['blacktract'] = raw0.blacktract.astype(int)
```

[12]:
```
# check the datatypes of data again
raw0.dtypes
```

[12]:
```
date              object
census             int32
person_gender     object
person_race       object
person_dob        object
traffic            int32
frisk_search       int64
arrest             int64
male               int64
offid              int64
dob_1             object
race_1            object
sex_1             object
apptdate_1        object
blacktract         int32
dtype: object
```

1.0.4 Accessing/selecting row(s), column(s) and cell(s) in Panda

https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/

https://towardsdatascience.com/a-python-beginners-look-at-loc-part-1-cb1e1e565ec2

- iloc - locate data points based on "updated/current" row & column numbers when some rows or columns are removed
- variable name - locate data points based on the "original" row & column numbers assigned when data was read
- loc - useful when selecting rows or columns that satisfies certain conditions (e.g. >, ==)

[13]:
```
# select a cell using iloc
raw0.iloc[99277, 0]
```

[13]:
```
'2006-01-24'
```

[14]:
```
raw0.tail()
```

[14]:
```
              date      census person_gender person_race  person_dob  traffic  \
102505  2008-08-28 -2147483648             M           W  1961-12-01        1
102506  2006-10-21 -2147483648             F           W  1959-09-05        1
102507  2008-11-14 -2147483648             M           B  1952-11-03        1
102508  2009-01-20 -2147483648             F           W  1986-06-07        1
102509  2006-01-24 -2147483648             M           W  1950-04-25        1

        frisk_search  arrest  male  offid       dob_1 race_1 sex_1  \
102505             0       0     1    518  1958-08-22      W     M
102506             0       0     0    519  1954-10-28      B     F
```

```
102507              0       0     1    520  1966-04-09      W     M
102508              0       0     0    521  1960-11-02      W     M
102509              0       0     1    522  1964-03-23      W     M

        apptdate_1  blacktract
102505  1993-12-16           1
102506  1981-12-16           0
102507  1997-11-14           1
102508  1985-09-16           1
102509  1989-03-31           1
```

[15]:
```
# select a cell using variable name
raw0.date[102509]
```

[15]:
```
'2006-01-24'
```

[16]:
```
raw0.iloc[-1,0]
# the last element in a list can be accessed using index -1 (this, however,␣
 ↪doesn't work when variable name is used to select the column)
# try raw0['date'][-1]
```

[16]:
```
'2006-01-24'
```

[17]:
```
raw0.iloc[0, 0]
```

[17]:
```
'2008-04-30'
```

[18]:
```
# access a part of a string in a cell using iloc
# ":x" selects the first x elements; "x:" selects from the (x+1)th element to␣
 ↪the last;
# "-x:" selects the last x elements; ":-x" select from the first to the xth␣
 ↪elements from the last;
raw0.iloc[0, 0][:-2]
```

[18]:
```
'2008-04-'
```

[19]:
```
raw0['date'][0][:-2]
```

[19]:
```
'2008-04-'
```

[20]:
```
# select columns using a variable names
raw0[['date','census']]
```

[20]:
```
              date       census
0       2008-04-30 -2147483648
1       2007-01-23 -2147483648
2       2009-01-22 -2147483648
3       2006-03-22 -2147483648
```

```
4       2007-07-27 -2147483648
…             …          …
102505  2008-08-28 -2147483648
102506  2006-10-21 -2147483648
102507  2008-11-14 -2147483648
102508  2009-01-20 -2147483648
102509  2006-01-24 -2147483648

[99278 rows x 2 columns]
```

[21]:
```
# access a part of a string in cell
raw0['date'][0][:3]
```

[21]:
```
'200'
```

[22]:
```
raw0['date'][102509]
```

[22]:
```
'2006-01-24'
```

1.0.5 In-Class Exercise 1: Create a dummy for person's gender

- We want to create a dummy variable that assigns 1 if driver is female (F), 0 otherwise

[23]:
```
# check unque elements (categories)
set(raw0['person_gender'])
```

[23]:
```
{'F', 'M', 'U'}
```

1.0.6 Important Operators in Python

https://www.programiz.com/python-programming/operators

https://www.geeksforgeeks.org/python-operators/

- Arithmetic
- Logical/Identity/Comparison
- Assignment

[24]:
```
raw0.person_gender == 'F'
```

[24]:
```
0         False
1         False
2         False
3         False
4          True
          …
102505    False
102506     True
102507    False
```

```
102508     True
102509    False
Name: person_gender, Length: 99278, dtype: bool
```

[25]:
```
raw0.person_gender == 'M'
```

[25]:
```
0          True
1          True
2          True
3          True
4         False
          …
102505     True
102506    False
102507     True
102508    False
102509     True
Name: person_gender, Length: 99278, dtype: bool
```

[26]:
```
(raw0.person_gender == 'F') | (raw0.person_gender == 'M')
```

[26]:
```
0         True
1         True
2         True
3         True
4         True
          …
102505    True
102506    True
102507    True
102508    True
102509    True
Name: person_gender, Length: 99278, dtype: bool
```

[27]:
```
# remove the rows with "u"
raw0 = raw0.loc[(raw0.person_gender == 'F') | (raw0.person_gender == 'M')]
```

[28]:
```
set(raw0['person_gender'])
```

[28]:
```
{'F', 'M'}
```

[29]:
```
# replace "person_gender" with a dummy that returns 1 if F, 0 otherwise
# This is an informal way to create a dummy and we will learn a python function␣
 ↪to create dummies next time
raw0.person_gender = (raw0.person_gender == 'F')*1
```

[30]:
```
raw0.person_gender
```

[30]:
```
0         0
1         0
2         0
3         0
4         1
         ..
102505    0
102506    1
102507    0
102508    1
102509    0
Name: person_gender, Length: 99266, dtype: int32
```

[31]:
```python
set(raw0['sex_1'])
```

[31]:
```
{'F', 'M'}
```

[32]:
```python
# do the same for sex_1
raw0['sex_1']= (raw0.sex_1 == 'F')*1
```

[33]:
```python
raw0.head()
```

[33]:
```
         date      census  person_gender person_race  person_dob  traffic  \
0  2008-04-30 -2147483648              0           W  1981-06-13        1
1  2007-01-23 -2147483648              0           B  1960-03-27        1
2  2009-01-22 -2147483648              0           B  1968-10-20        1
3  2006-03-22 -2147483648              0           B  1982-09-18        1
4  2007-07-27 -2147483648              1           W  1981-08-30        1

   frisk_search  arrest  male  offid       dob_1 race_1  sex_1  apptdate_1  \
0             0       0     1      1  1978-02-07      W      0  2005-07-25
1             0       0     1      1  1978-02-07      W      0  2005-07-25
2             0       0     1      1  1978-02-07      W      0  2005-07-25
3             0       0     1      1  1978-02-07      W      0  2005-07-25
4             0       0     0      1  1978-02-07      W      0  2005-07-25

   blacktract
0           1
1           1
2           1
3           1
4           1
```

1.0.7 In-Class Exercise 2: Create a set of dummies for person's race

- we want to create two dummies
  - First dummy returns 1 if driver is B, 0 otherwise
  - Second dummy returns 1 if driver is A or I or O or U (W is the baseline group)

```
[34]: # check the unque elements in person_race and create/add a dummy, D_B, to data
      set(raw0['person_race'])
```

```
[34]: {'A', 'B', 'I', 'O', 'U', 'W'}
```

```
[35]: # there are many other ways to add new columns to data in Panda (see https://
       ↪www.geeksforgeeks.org/adding-new-column-to-existing-dataframe-in-pandas/)
      raw0['D_B'] = (raw0.person_race == 'B')*1
```

```
[36]: # do the same for the other dummy
      raw0['D_Other'] = ((raw0.person_race == 'A')|(raw0.person_race == 'I')|(raw0.
       ↪person_race == 'O')|(raw0.person_race == 'U'))*1
```

```
[37]: raw0.head()
```

```
[37]:          date      census  person_gender person_race  person_dob  traffic  \
      0  2008-04-30 -2147483648              0           W  1981-06-13        1
      1  2007-01-23 -2147483648              0           B  1960-03-27        1
      2  2009-01-22 -2147483648              0           B  1968-10-20        1
      3  2006-03-22 -2147483648              0           B  1982-09-18        1
      4  2007-07-27 -2147483648              1           W  1981-08-30        1

         frisk_search  arrest  male  offid       dob_1 race_1  sex_1  apptdate_1  \
      0             0       0     1      1  1978-02-07      W      0  2005-07-25
      1             0       0     1      1  1978-02-07      W      0  2005-07-25
      2             0       0     1      1  1978-02-07      W      0  2005-07-25
      3             0       0     1      1  1978-02-07      W      0  2005-07-25
      4             0       0     0      1  1978-02-07      W      0  2005-07-25

         blacktract  D_B  D_Other
      0           1    0        0
      1           1    1        0
      2           1    1        0
      3           1    1        0
      4           1    0        0
```

1.0.8 In-Class Exercise 3: create age variable for driver

- Definition of age: age = date(string, yyyy-mm-dd) -person_dob(string, yyyy-mm-dd) (i.e., the age of driver at the time of stop)

```
[38]: # [Step 1]
      # get the years of "person_dob" and "date," and store them in "dyear" and␣
       ↪"byear"
      # covert the strings to integers and calculate the difference
      dyear=raw0['date'][0][:4] # alternatively, raw0.iloc[0,0][:4]
      byear=raw0['person_dob'][0][:4]
      dyearn = int(dyear)
```

```
byearn = int(byear)
age = dyearn - byearn
age
```

[38]: 27

[39]:
```
dyearns = str(dyearn)
```

[40]:
```
dyearns
```

[40]: '2008'

[41]:
```
# [Step 2]
# get the months of "person_dob" and "date," and store them in "dmon" and "bmon"
# covert the strings to integers and calculate the difference
# if the difference in month is negative (i.e., his/her birthday hadn't yet␣
 ↪passed at the time of stop),\
# then subtract one from age
dmon=raw0['date'][0][5:7]
bmon=raw0['person_dob'][0][5:7]
dmonn = int(dmon)
bmonn = int(bmon)
mond = dmonn - bmonn
if mond < 0:
    age = age -1
age
```

[41]: 26

1.0.9 For-Loops in Python

https://www.w3schools.com/python/python_for_loops.asp

1.0.10 If statements in Python

https://www.w3schools.com/python/python_conditions.asp

[42]:
```
# Three different ways to repeat the calculation for all the observations

# (1) Using for-loop and iloc
D_age = np.zeros((len(raw0),),dtype=int)

for i in range(0,len(raw0)):
    age=int(raw0.iloc[i,0][:4]) - int(raw0.iloc[i,4][:4])
    mond=int(raw0.iloc[i,0][5:7]) - int(raw0.iloc[i,4][5:7])
    if mond < 0:
        age = age -1
    D_age[i]=age
```

[43]:
```
print(D_age)
```

```
[26 46 40 … 56 22 55]
```

[44]:
```
# add it to raw0
raw0['D_age'] = D_age
```

[45]:
```
# (2) Using for-loop and variable name
raw0['D_age2'] = 0 # create a colum of zeros in raw0

#for i in raw0.index:
    #age=int(raw0.date[i][:4]) - int(raw0.person_dob[i][:4])
    #mond=int(raw0.date[i][5:7]) - int(raw0.person_dob[i][5:7])
    #if mond < 0:
        #age = age -1
    #raw0.D_age2[i]=age


# Note that we use "raw0.index" instead of "range(0,len(raw0))"
# This is because of the difference in how "iloc" and "variable name" method␣
 ↪access the data (current v.s. original row numbers)
```

[46]:
```
# (3) Using "Apply" function
# https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.apply.html
# if calculations are rather complex, looping is better (also, the apply␣
 ↪function includes a looping precedure)

D_age3 = raw0.apply(lambda x: int(x['date'][:4]) - int(x['person_dob'][:4]) - 1␣
 ↪if int(x['date'][5:7]) - int(x['person_dob'][5:7]) < 0 else int(x['date'][:
 ↪4]) - int(x['person_dob'][:4]), axis=1)
```

[47]:
```
print(D_age3)
```

```
0         26
1         46
2         40
3         23
4         25
          ..
102505    46
102506    47
102507    56
102508    22
102509    55
Length: 99266, dtype: int64
```

[48]:
```
raw0['D_age3'] = D_age3
```

[49]:
```
raw0.head()
```

[49]:
```
         date      census  person_gender person_race  person_dob  traffic  \
0  2008-04-30 -2147483648              0           W  1981-06-13        1
1  2007-01-23 -2147483648              0           B  1960-03-27        1
2  2009-01-22 -2147483648              0           B  1968-10-20        1
3  2006-03-22 -2147483648              0           B  1982-09-18        1
4  2007-07-27 -2147483648              1           W  1981-08-30        1

   frisk_search  arrest  male  offid       dob_1 race_1  sex_1  apptdate_1  \
0             0       0     1      1  1978-02-07      W      0  2005-07-25
1             0       0     1      1  1978-02-07      W      0  2005-07-25
2             0       0     1      1  1978-02-07      W      0  2005-07-25
3             0       0     1      1  1978-02-07      W      0  2005-07-25
4             0       0     0      1  1978-02-07      W      0  2005-07-25

   blacktract  D_B  D_Other  D_age  D_age2  D_age3
0           1    0        0     26       0      26
1           1    1        0     46       0      46
2           1    1        0     40       0      40
3           1    1        0     23       0      23
4           1    0        0     25       0      25
```

1.0.11 HW1: Similarly as we have done for D_age,

1. create an age variable for officer: O_age
2. create a tenure variable for officer: Exp, defined as exp = date - apptdate_1
3. Append the two variables to raw0

To submit your HW, go to File -> Download as PDF via Latex; for this to work, "pandoc" should be installed: https://github.com/jgm/pandoc/tree/3.1.6.1

[50]:
```
# Number 1

# Using 'Apply' function
# Calculating officer age
raw0['O_age'] = raw0.apply(lambda x: int(x['date'][:4]) - int(x['dob_1'][:4]) -␣
↪1 if int(x['date'][5:7]) - int(x['dob_1'][5:7]) < 0 else int(x['date'][:4])␣
↪- int(x['dob_1'][:4]), axis=1)

print(raw0[['date', 'dob_1', 'O_age']].head())

print("Shape of raw0:", raw0.shape)
```

```
         date       dob_1  O_age
0  2008-04-30  1978-02-07     30
1  2007-01-23  1978-02-07     28
2  2009-01-22  1978-02-07     30
3  2006-03-22  1978-02-07     28
```

```
4  2007-07-27  1978-02-07     29
Shape of raw0: (99266, 21)
```

[51]:
```
# Number 2

# Using for-loop and iloc
import numpy as np

# Create numpy array to store the results
Exp = np.zeros((len(raw0),),dtype=int)

# Getting column index for appointment date
apptdate_index = raw0.columns.get_loc('apptdate_1')

# Looping through each row in dataset
for i in range(0,len(raw0)):

    # Calculates years of expierence
    exp = int(raw0.iloc[i, 0][:4]) - int(raw0.iloc[i, apptdate_index][:4])

    # Calculates month difference
    mond = int(raw0.iloc[i, 0][5:7]) - int(raw0.iloc[i, apptdate_index][5:7])

    # Condition if current month is earlier than the appointment date month,␣
  ↪subtracts one from 'Exp'
    if mond < 0:
        exp = exp -1
    Exp[i] = exp

# Add the results as a new column to dataset
raw0['Exp'] = Exp

print(raw0[['date', 'apptdate_1', 'Exp']].head())
```

```
         date  apptdate_1  Exp
0  2008-04-30  2005-07-25    2
1  2007-01-23  2005-07-25    1
2  2009-01-22  2005-07-25    3
3  2006-03-22  2005-07-25    0
4  2007-07-27  2005-07-25    2
```

[52]:
```
# Number 3
print(raw0[['date', 'O_age', 'apptdate_1', 'Exp']].head())
```

```
         date  O_age  apptdate_1  Exp
0  2008-04-30     30  2005-07-25    2
1  2007-01-23     28  2005-07-25    1
2  2009-01-22     30  2005-07-25    3
3  2006-03-22     28  2005-07-25    0
```

```
4  2007-07-27      29  2005-07-25     2
```

```
[ ]:
```