Python_Module8

September 27, 2020

0.1 Author: Joseph Vargovich

[80]:

```
#Import libraries
import pandas as pd
import numpy as np
from patsy import dmatrices
import statsmodels.api as sm
```

1 Exercise 1: Create vectors of length three and add them.

[4]:

```
 #Create a numpy vector
vec_a = np.array([2,4,6])
vec_b = np.array([8,10,12])


vec_c = vec_a + vec_b

vec_c
```

[4]:

```
array([10, 14, 18])
```

[ ]:

```
# Exercise 2: Create vectors of differing size and add them.
```

[5]:

```
#Hmm, does Python have a recyling rule?
vec_d = np.array([14,20])

vec_a + vec_d #It does not! We get an error instead of recyling.
```

```
           ␣
↪---------------------------------------------------------------------------


       ValueError                                Traceback (most recent call␣
↪last)


       <ipython-input-5-ee969c48c52e> in <module>
         2 vec_d = np.array([14,20])
```

```
      3
----> 4 vec_a + vec_d


    ValueError: operands could not be broadcast together with shapes (3,)␣
↪(2,)
```

2 Exercise 3 - Add a constant to a vector

[6]:

```
vec_a + 5
```

[6]:

```
array([ 7,  9, 11])
```

3 Exercise 4 - Generate a vector of integers with two methods

[15]:

```
#1 with numpy arange
print(np.arange(1,6)) #This is not an inclusive upper range value, unlike R.

#2 with the built-in range function (No simple colon operator in Python :( )
print(list(range(1,6)))
```

```
[1 2 3 4 5]
[1, 2, 3, 4, 5]
```

4 Exercise 5 - Generate a vector of even integers with two methods

[18]:

```
#1 Use numpy's arange in place of R's seq function
listNp = np.arange(2,22, 2)
print(listNp)

#2 with the built-in range function
list1 = list(range(1,11))
list2 = [i * 2 for i in list1]
print(list2)
```

```
[ 2  4  6  8 10 12 14 16 18 20]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

5 Exercise 6 - Generate a vector of 21 evenly spaced elements between 0 and 1

[19]:
```
# This utilizes np.linspace()
x = np.linspace(0,1,21)
print(x)
```

```
[0.   0.05 0.1  0.15 0.2  0.25 0.3  0.35 0.4  0.45 0.5  0.55 0.6  0.65
 0.7  0.75 0.8  0.85 0.9  0.95 1.  ]
```

6 Exercise 7 - Generate a vector by repeating another one

[32]:
```
#This is the default behavior of np.tile
to_repeat = np.array([2,4,8])

result1 = np.tile(to_repeat,3)

print(result1)
```

```
[2 4 8 2 4 8 2 4 8]
```

7 Exercise 8 - Generate a vector by repeating another one, keep the elements in order this time

[37]:
```
#This is the default behavior of np.repeat()
letters = np.array

print(result2)
```

```
[2 2 2 2 4 4 4 4 8 8 8 8]
```

8 Exercise 9 - Work with the letters dataset

[61]:
```
letters = np.
 ↪array(['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u',

#a. Get the 9th element (indexing starts at 0 in Python)
print(letters[8])

#b. Get the subvector of the 9th, 11th, and 19th elements
print(letters[[8,10,17]])

#c. Get the subvector of elements that exclude the last 2 elements
print(letters[0:24]) #This is identical to Python as the second element of the␣
 ↪range is not inclusive. [0,24) in this example.
```

```
i
['i' 'k' 'r']
['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r'
 's' 't' 'u' 'v' 'w' 'x']
```

9 Exercise 10 - Create a matrix and perform operations

[42]:

```
#a. Create an nd matrix to hold a sequence of numbers using ndarray()
matrix1 = np.ndarray(shape=(3,5), buffer=np.arange(2,32,2), order='C', dtype=np.
 ↪int32)
print("Matrix1: \n")
print(matrix1)

#b. Create anotehr matrix by stacking vectors using np.vstack
row1 = np.arange(2,12, 2)
row2 = np.arange(12,22, 2)
row3 = np.arange(22,32, 2)

#Build the matrix
matrix2 = np.vstack((row1,row2,row3))
#matrix2 = np.concatenate((matrix2,row3), axis=1)

print("\nMatrix2: \n")
print(matrix2)


#c. Grab the second row of the matrix
print("\n", matrix1[1])

#d. Grab the second element of the third row
print("\n", matrix[2][1])
```

```
Matrix1:

[[ 2  4  6  8 10]
 [12 14 16 18 20]
 [22 24 26 28 30]]

Matrix2:

[[ 2  4  6  8 10]
 [12 14 16 18 20]
 [22 24 26 28 30]]

 [12 14 16 18 20]

 24
```

10 Exercise 11 - Create and manipulate a data frame.

[89]:

```
#a. Create a trees dataframe
dataDict = {'Girth': [8.3, 8.6,8.8,10.5, 10.7, 10.8, 11.0], 'Height': [70, 65,␣
 ↪63, 72, 81, 83, 66], 'Volume': [10.3, 10.3, 10.2, 16.4, 18.8, 19.7, 15.6]}
my_trees = pd.DataFrame(data=dataDict)

#b. Extract the third element (row) from the dataframe
print("Thrid row: \n",my_trees[2:3], "\n\n")

#c. Extract the Girth column by name
print("Girth column: \n",my_trees['Girth'])

#d. Grab every row but the fourth one
print("My_Trees with row 4 dropped: \n", my_trees.drop(3,axis=0))

#e. Select from the df based on condition
my_trees2 = my_trees[my_trees['Girth'] > 10]
print("Girth Greater than 10 selected: \n", my_trees2)

#f. Create a data set with just the large trees
dataDictLarge = {'Girth': my_trees['Girth'], 'Height': my_trees['Height'],␣
 ↪'Volume': my_trees['Volume']}

my_trees_large = pd.DataFrame(data=dataDictLarge)
print("Large Trees: \n", my_trees_large )

#g. Create a data set with just the small trees
my_trees_small = my_trees[my_trees['Girth'] < 10]
print("Small Trees: \n", my_trees_small )
```

```
Thrid row:
    Girth  Height  Volume
2    8.8      63    10.2



Girth column:
 0     8.3
1     8.6
2     8.8
3    10.5
4    10.7
5    10.8
6    11.0
Name: Girth, dtype: float64
My_Trees with row 4 dropped:
    Girth  Height  Volume
```

```
0    8.3      70    10.3
1    8.6      65    10.3
2    8.8      63    10.2
4   10.7      81    18.8
5   10.8      83    19.7
6   11.0      66    15.6
Girth Greater than 10 selected:
    Girth  Height  Volume
3   10.5      72    16.4
4   10.7      81    18.8
5   10.8      83    19.7
6   11.0      66    15.6
Large Trees:
    Girth  Height  Volume
0    8.3      70    10.3
1    8.6      65    10.3
2    8.8      63    10.2
3   10.5      72    16.4
4   10.7      81    18.8
5   10.8      83    19.7
6   11.0      66    15.6
Small Trees:
    Girth  Height  Volume
0    8.3      70    10.3
1    8.6      65    10.3
2    8.8      63    10.2
```

11 Exercise 14 - Create and manipulate a list

[90]:

```
#Not really a true equivalent, but we can still store various sized items with␣
 ↪a dictionary
dict_struct = {'x': [4,5,6,7,8,9,10], 'y': [34,35,41,40,45,47,51], 'slope': [2.
 ↪82], 'p-value': [0.000131]}

#b. Grab the second value of the dictionary
print("y: ",dict_struct['y'])

#c. Grab the p-value
print("p-val: ", dict_struct['p-value'])
```

```
y:  [34, 35, 41, 40, 45, 47, 51]
p-val:  [0.000131]
```

12 Exercise 15 - Examine the data structures used with linear models lm()

[88]:
```
#a. Load the trees dataset
trees = pd.read_csv("trees.csv")

#b. Examine the dataset head
print(trees.head())

#c. Perform a linear regression relating volume of lumber to girth and height
#b.  Create a linear regression model for the dataframe.
Volume, GirthHeight = dmatrices('Volume ~ Girth + Height ', data=trees,␣
 ↪return_type='dataframe')
model = sm.OLS(Volume, GirthHeight)

fitModel = model.fit()
print(fitModel.summary(), "\n\n\n")

print(fitModel.params)
```

```
   Unnamed: 0  Girth  Height  Volume
0           1    8.3      70    10.3
1           2    8.6      65    10.3
2           3    8.8      63    10.2
3           4   10.5      72    16.4
4           5   10.7      81    18.8
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Volume   R-squared:                       0.948
Model:                            OLS   Adj. R-squared:                  0.944
Method:                 Least Squares   F-statistic:                     255.0
Date:                Sun, 27 Sep 2020   Prob (F-statistic):           1.07e-18
Time:                        20:46:26   Log-Likelihood:                -84.455
No. Observations:                  31   AIC:                             174.9
Df Residuals:                      28   BIC:                             179.2
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -57.9877      8.638     -6.713      0.000     -75.682     -40.293
Girth          4.7082      0.264     17.816      0.000       4.167       5.249
Height         0.3393      0.130      2.607      0.014       0.073       0.606
==============================================================================
Omnibus:                        0.923   Durbin-Watson:                   1.266
Prob(Omnibus):                  0.630   Jarque-Bera (JB):                0.950
Skew:                           0.310   Prob(JB):                        0.622
```

```
Kurtosis:                       2.408   Cond. No.                         959.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
Intercept   -57.987659
Girth         4.708161
Height        0.339251
dtype: float64
```

[ ]: