Python_Module4

September 4, 2020

1 Author: Joseph Vargovich

[1]:
```
#Import libraries
import pandas as pd
```

2 Exercise 1 - Manipulations and filters on the ChickWeight dataset imported from R.

[2]:
```
#Read the address csv and store it here.
chickWeight = pd.read_csv("chickWeight.csv")

#a. Apply the filter function to get measurements from day 10 OR day 20.
chickWeight = chickWeight[chickWeight['Time'].isin([10, 20])]

#Calculate the mean and std deviation of each diet group from this modified␣
 ↪dataframe
chickWeightGrps = chickWeight.groupby(['Diet'])
chickWeightMean = chickWeightGrps['weight'].mean()
print("Mean weight across the 4 diets\n", chickWeightMean)
chickWeightSd = chickWeightGrps['weight'].std()
print("\n")
print("Std deviation across the 4 diets\n", chickWeightSd)
```

```
Mean weight across the 4 diets
 Diet
1    129.583333
2    157.050000
3    188.000000
4    177.105263
Name: weight, dtype: float64


Std deviation across the 4 diets
 Diet
1    56.571257
2    71.404316
```

```
3    86.607402
4    61.284305
Name: weight, dtype: float64
```

3 Exercise 2 - OpenIntro body dimensions pipeline.

[3]:
```
%%capture
#a. The OpenIntro textbook on statistics includes a data set on body dimensions.
 ↪ Unfortunately the server seems to be blocking my request, so I will␣
 ↪manually download it from the link instead.
"""url = 'http://www.openintro.org/stat/data/bdims.csv'
response = requests.get(url).content"""
```

[4]:
```
#Read the address csv and store it here.
body = pd.read_csv("bdims.csv")

# b. The column sex is coded as a 1 if the individual is male and 0 if female.␣
 ↪This is a non-intuitive labeling system. Create a new column sex.MF that␣
 ↪uses labels Male and Female.
body = body.assign(sexMF = ['Male' if a == 1 else 'Female' for a in␣
 ↪body['sex']])

#c. The columns wgt and hgt measure weight and height in kilograms and␣
 ↪centimeters (respectively). Use these to calculate the Body Mass Index (BMI)␣
 ↪for each individual.
#Sort of a hackish way to append the BMI, I would look to see if there is a␣
 ↪better way to do this that is more concise.
bmi = []
wgt = 0
hgt = 0
#Index is used with iterrows, row is our current row to pull from body.
for index, row in body.iterrows():
    hgt = row['hgt']
    wgt = row['wgt']
    bmi.append((wgt)/(hgt/100)**2)
#Once we built a bmi list, append it to body to serve as a column for BMI␣
 ↪calculations.
body['BMI'] = bmi

#e. Create a new column of in the data frame that divides the age into decades␣
 ↪(10-19, 20-29, 30-39, etc).
body['Decades'] = pd.cut(body.age, [10, 20, 30, 40, 50, 60, 70], right=False)

#f.   Find the average BMI for each Sex by decades combination
body = body.groupby(['sexMF', 'Decades'])
# Store the mean of the groups in a new data frame.
```

```
meanDF = body['BMI'].mean()
print(meanDF)
```

```
sexMF   Decades
Female  [10, 20)    21.840488
        [20, 30)    21.766430
        [30, 40)    22.531684
        [40, 50)    24.255105
        [50, 60)    22.666894
        [60, 70)    23.694256
Male    [10, 20)    25.508854
        [20, 30)    24.166745
        [30, 40)    24.887053
        [40, 50)    26.368220
        [50, 60)    24.753149
        [60, 70)    23.908969
Name: BMI, dtype: float64
```