Dingyi Duan

Module 6 Assignment

ADS-500B

04/11/2021

Module 6 Assignment Questions

Note that the answers to each of these questions should be the direct result of running appropriate Python or R code and not involve any manual processing of dataset files. Answers without either the code or results will not receive any grade.

1. For the next exercise, you are going to use the “airline_costs.csv” dataset.

The dataset has the following attributes:

i. Airline name

ii. Length of flight in miles

iii. Speed of plane in miles per hour

iv. Daily flight time per plane in hours

v. Customers served in 1000s

vi. Total operating cost in cents per revenue ton-mile

vii. Revenue in tons per aircraft mile

viii. Ton-mile load factor

ix. Available capacity

x. Total assets in $100,000s

xi. Investments and special funds in $100,000s

xii. Adjusted assets in $100,000s

(Implement this exercise in Python language; import “pandas”, “statsmodels.api” libraries) Use a linear regression model to predict the number of customers each airline serves from its length of flight and daily flight time per plane.

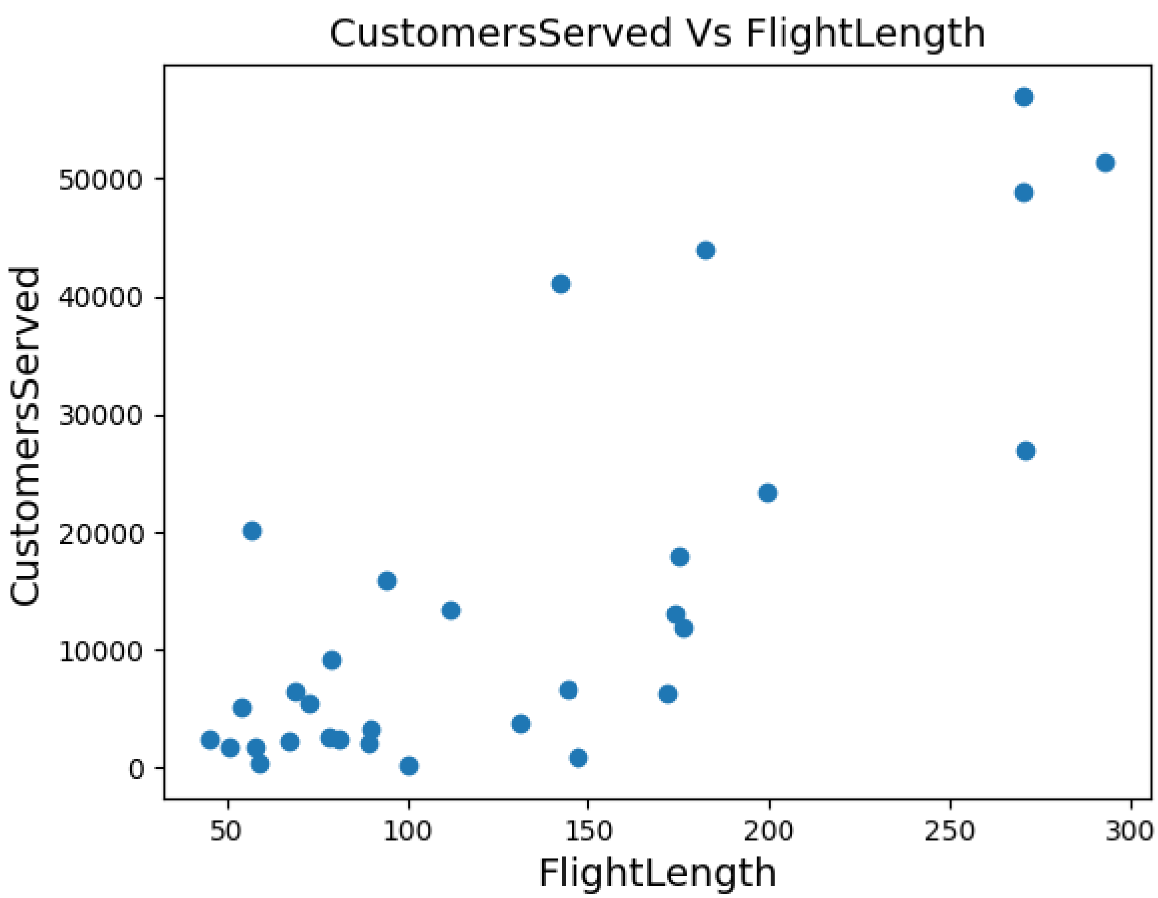
```
# Q6.1

import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Set max display
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)


# Read the tsv file
df = pd.read_csv('C:/Users/DDY/Desktop/2021-Spring-textbooks/ADS-500B/Module6/airline_costs.csv',header=0)

print (df)

# Check for nulls for preprocessing
print (df.isna().sum())
```

```
Airline                0
FlightLength           0
PlaneSpeed             0
DailyFlightTime        0
CustomersServed        0
TotalOperatingCost     0
Revenue                0
LoadFactor             0
AvailableCapacity      0
TotalAssets            0
Investments            0
AdjustedAssets         0
dtype: int64
```

```
# Check for Multilinearity between 'DailyFlightTime' and 'FlightLength'
print (round(df['DailyFlightTime']. corr(df['FlightLength']),2))
```

Check for multicollinearity between two independent variables: `0.48` (weak)

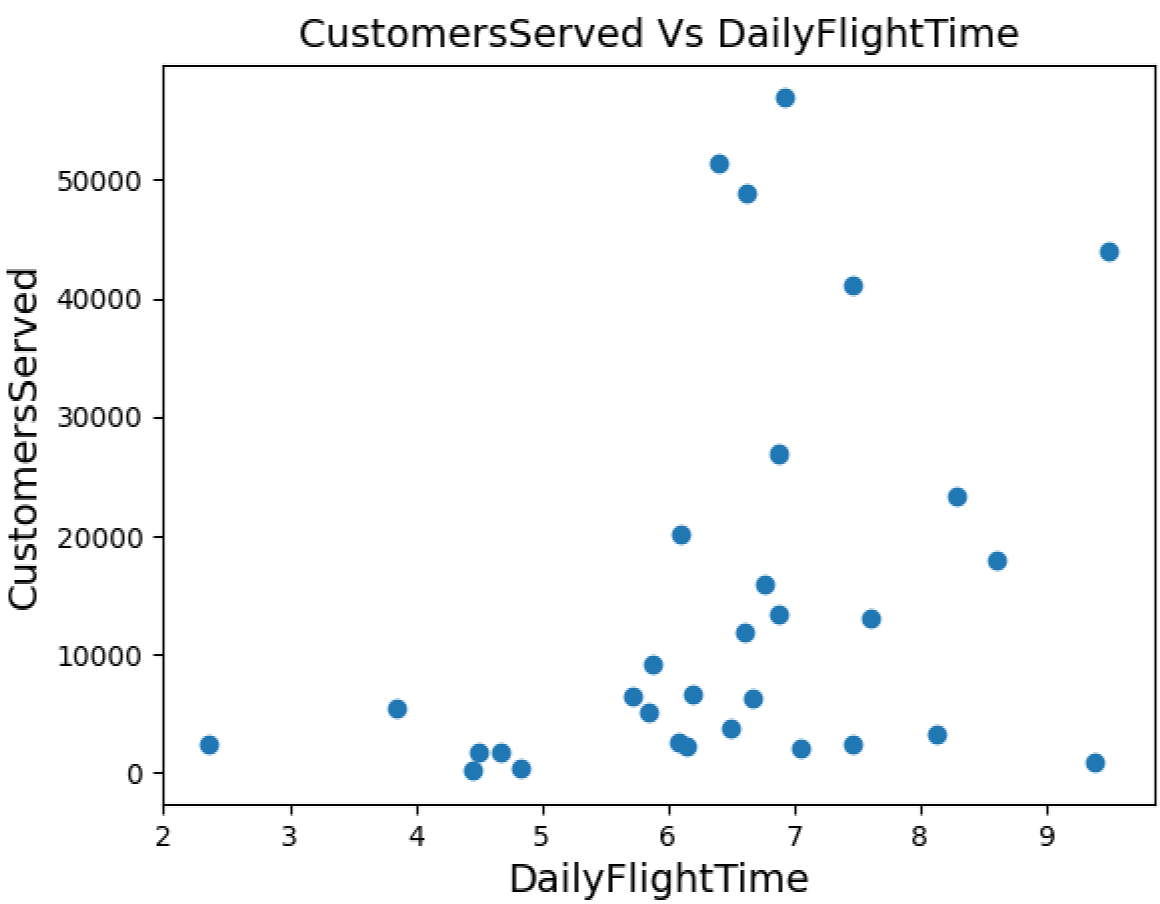
```python
# Check for linearity between dependent and independent variables

# Linearity between 'CustomersServed' and 'FlightLength'

print (round(df['CustomersServed']. corr(df['FlightLength']),2))
# Correlation coefficient: 0.79 (strong)

plt.scatter(df['FlightLength'], df['CustomersServed'])
plt.title('CustomersServed Vs FlightLength', fontsize=14)
plt.xlabel('FlightLength', fontsize=14)
plt.ylabel('CustomersServed', fontsize=14)
plt.show()
```



Correlation coefficients for 'CustomersServed' and 'FlightLength' and 'CustomersServed' and 'DailyFlightTime' :

```
0.79
0.36
```

```
# Linearity between 'CustomersServed' and 'DailyFlightTime'

print (round(df['CustomersServed'].corr(df['DailyFlightTime']),2))
# Correlation coefficient: 0.36 (weak) as we see outliers

plt.scatter(df['DailyFlightTime'], df['CustomersServed'])
plt.title('CustomersServed Vs DailyFlightTime', fontsize=14)
plt.xlabel('DailyFlightTime', fontsize=14)
plt.ylabel('CustomersServed', fontsize=14)
plt.show()
```



```
# =================== Detect and clean up outliers =====================


# Check outliers for 'FlightLength'
FlightLength_Q1 = df.FlightLength.quantile(0.25)
FlightLength_Q3 = df.FlightLength.quantile(0.75)
FlightLength_IQR = FlightLength_Q3 - FlightLength_Q1
FlightLength_out = (df.FlightLength < (FlightLength_Q1 - 1.5 * FlightLength_IQR))|(df.FlightLength
                                                                                   > (FlightLength_Q3 + 1.5 * FlightLength_IQR))

print (FlightLength_out) # No outliers
```

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
30     False
Name: FlightLength, dtype: bool
```

```
DailyFlightTime_Q1 = df.DailyFlightTime.quantile(0.25)
DailyFlightTime_Q3 = df.DailyFlightTime.quantile(0.75)
DailyFlightTime_IQR = DailyFlightTime_Q3 - DailyFlightTime_Q1
DailyFlightTime_out = (df.DailyFlightTime < (DailyFlightTime_Q1 - 1.5 * DailyFlightTime_IQR))|(df.DailyFlightTime
                                                                                               > (DailyFlightTime_Q3 + 1.5 * DailyFlightTime_IQR))

print(DailyFlightTime_out) # Outliers: row10,28,29


df = df[~(DailyFlightTime_out)] # Revove outliers from df
```

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10      True
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28      True
29      True
30     False
Name: DailyFlightTime, dtype: bool
```

Multiple Linear Regression Model:

```python
# Perform Multiple Linear Regression using statsmodels

X = df[['FlightLength','DailyFlightTime']]
y = df['CustomersServed']

X = sm.add_constant(X)  # adding a constant

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)

# Equation: CustomersServed = 179.69 * FlightLength - 244.11 * DailyFlightTime - 7372.77
# R-squared = 0.654
```

Multiple Linear model:

CustomersServed = 179.69 * FlightLength - 244.11 * DailyFlightTime - 7372.77

R-squared = 0.654

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         CustomersServed   R-squared:                       0.654
Model:                             OLS   Adj. R-squared:                  0.626
Method:                  Least Squares   F-statistic:                     23.64
Date:                 Sun, 11 Apr 2021   Prob (F-statistic):           1.73e-06
Time:                         16:26:04   Log-Likelihood:                -296.16
No. Observations:                   28   AIC:                             598.3
Df Residuals:                       25   BIC:                             602.3
Df Model:                            2
Covariance Type:             nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const           -7372.7739   1.08e+04     -0.682      0.502   -2.96e+04    1.49e+04
FlightLength      179.6901     29.009      6.194      0.000     119.945     239.436
DailyFlightTime  -244.1079   1844.128     -0.132      0.896   -4042.160    3553.944
==============================================================================
Omnibus:                        4.032   Durbin-Watson:                   1.876
Prob(Omnibus):                  0.133   Jarque-Bera (JB):                2.736
Skew:                           0.751   Prob(JB):                        0.255
Kurtosis:                       3.301   Cond. No.                         859.
==============================================================================
```

Next, build another regression model to predict the total assets of an airline from the customers served by the airline.

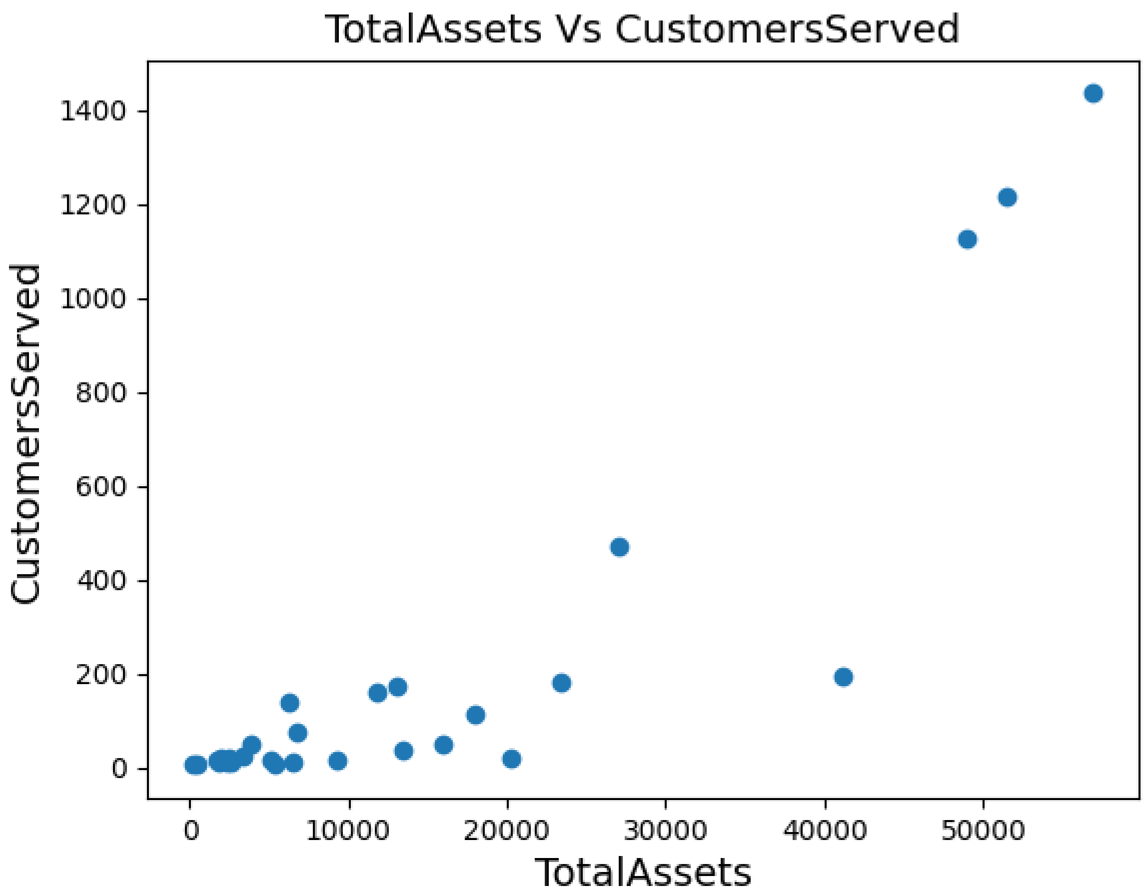
Simple Linear Regression Model:

```
# ========================== Second linear regression model ========================

# Linearity between 'CustomersServed' and 'FlightLength'

print (round(df['TotalAssets']. corr(df['CustomersServed']),2))
# Correlation coefficient: 0.89 (strong)

plt.scatter(df['CustomersServed'], df['TotalAssets'])
plt.title('TotalAssets Vs CustomersServed', fontsize=14)
plt.xlabel('TotalAssets', fontsize=14)
plt.ylabel('CustomersServed', fontsize=14)
plt.show()
```

Correlation coefficients for 'CustomersServed' and 'FlightLength' is 0.89

```python
# Check outliers for 'CustomersServed'
CustomersServed_Q1 = df.CustomersServed.quantile(0.25)
CustomersServed_Q3 = df.CustomersServed.quantile(0.75)
CustomersServed_IQR = CustomersServed_Q3 - CustomersServed_Q1
CustomersServed_out = (df.CustomersServed
                       < (CustomersServed_Q1 - 1.5 * CustomersServed_IQR)) | (df.CustomersServed
                                                                              > (CustomersServed_Q3 + 1.5 * CustomersServed_IQR))

print (CustomersServed_out) # Outliers: row1,24,25

df = df[~(CustomersServed_out)] # Revove outliers from df
```

```
0.89
0      False
1       True
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24      True
25      True
26     False
27     False
30     False
Name: CustomersServed, dtype: bool
```




```
# Perform Multiple Linear Regression

x = df[['CustomersServed']]
y = df['TotalAssets']

x = sm.add_constant(x)  # adding a constant

model = sm.OLS(y, x).fit()
predictions = model.predict(x)

print_model = model.summary()
print(print_model)

# Equation: CustomersServed = 0.0072 * FlightLength + 2.86
# R-squared = 0.472
```

Linear model:

TotalAssets = 0.0072 * CustomersServed + 2.86

R-squared = 0.472

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalAssets   R-squared:                       0.472
Model:                            OLS   Adj. R-squared:                  0.449
Method:                 Least Squares   F-statistic:                     20.58
Date:                Sun, 11 Apr 2021   Prob (F-statistic):           0.000148
Time:                        16:40:36   Log-Likelihood:                -143.19
No. Observations:                  25   AIC:                             290.4
Df Residuals:                      23   BIC:                             292.8
Df Model:                           1
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const               2.8593     21.936      0.130      0.897     -42.519      48.238
CustomersServed     0.0072      0.002      4.536      0.000       0.004       0.010
==============================================================================
Omnibus:                       22.944   Durbin-Watson:                   2.353
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               40.383
Skew:                           1.764   Prob(JB):                     1.70e-09
Kurtosis:                       8.131   Cond. No.                     1.96e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.96e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Do you have any insight about the data from the last two regression models? (20 points)

Answer: For multiple linear regression models, we are dealing with multiple variables that may have weak linear relationships with our predictor. From the first summary table we can see that the p value for 'DailyFlightTime' is 0.896 which means this variable is not statistically significant, which also matches the weak correlation coefficient 0.36. However, when we finish building the model, we obtain a better R-squared value of 0.654; Compared to the simple linear regression model where we have the dependent and independent variables are highly correlated to each other, we only obtain an R-squared value of 0.472 which leads to a less efficient model than multiple linear one. To conclude, it is more accurate to build a linear regression model with multiple variables than a single independent variable.

2. For this clustering exercise, you are going to use the data on women professional golfers performance on the LPGA, 2008 tour (‘lpga2008.csv’ dataset). The dataset has the following attributes:

i. Golfer: name of the player

ii. Average Drive distance

iii. Fairway Percentage

iv. Greens in regulation: in percentage

v. Average putts per round

vi. Sand attempts per round

vii. Sand saves: in percentage

viii. Total Winnings per round

ix. Log: Calculated as (Total Win/Round)

x. Total Rounds

xi. Id: Unique ID representing each player

(Implement this exercise in R language; import ‘cluster’ library)

Use agglomerative clustering and divisive clustering on this dataset to find out which players have similar performance in the same season. Visualize the clusters using dendrograms for both types of clustering models. (20 points)

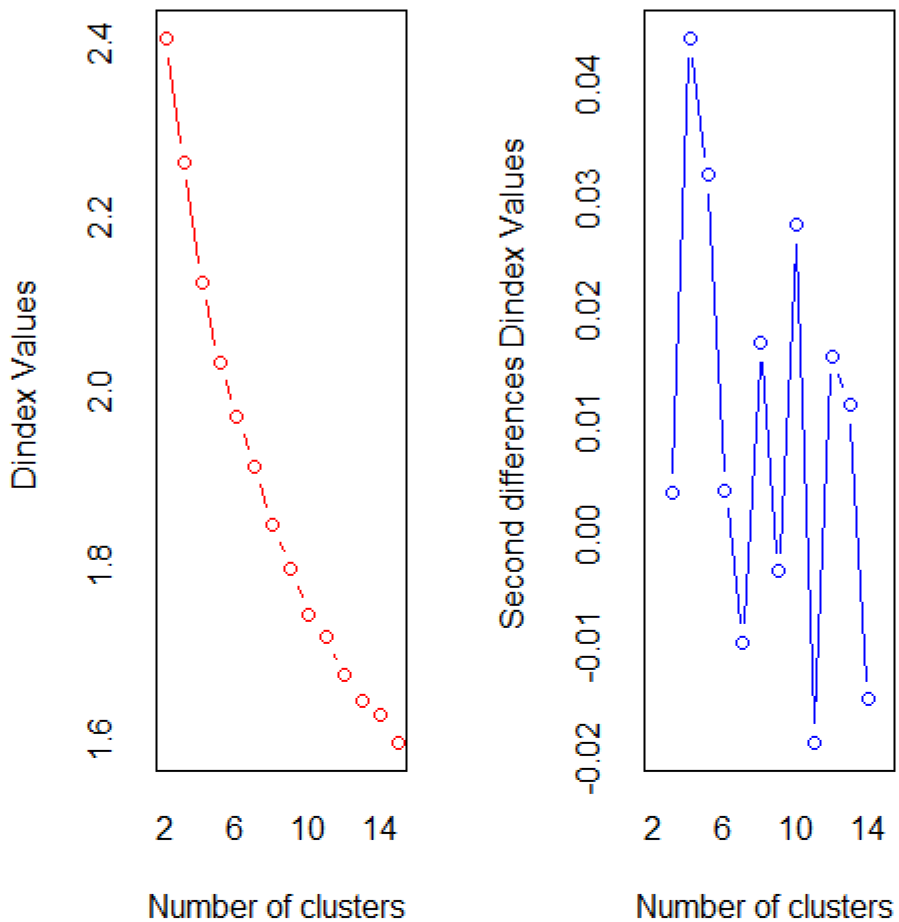
```
# Q6.2

library(cluster)
library(factoextra)
library(NbClust)


# Read the file
lpga = read.table('C:/Users/DDY/Desktop/2021-Spring-textbooks/ADS-500B/Module6/lpga2008.csv',header=T,sep=',')

# Remove '?' and '0's
lpga[lpga == '?' && '0'] = NA
lpga = na.omit(lpga)

# Count nulls >>> no nulls
sum(is.na(lpga))
```

```
> # Count nulls >>> no nulls
> sum(is.na(lpga))
[1] 0
>
```

```
# Standardizing the dataset
newlpga = lpga[-1]
row.names(newlpga)=lpga$Golfer
newlpga$Id = NULL
View(newlpga)
lpga_st = scale(newlpga)
```

```
# Determining 'k'

# Using nbClust() to determine 'k' value
NbClust(lpga_st,distance='euclidean', method='kmeans') # According to majority rule,
# the best number of clusters is 2
```

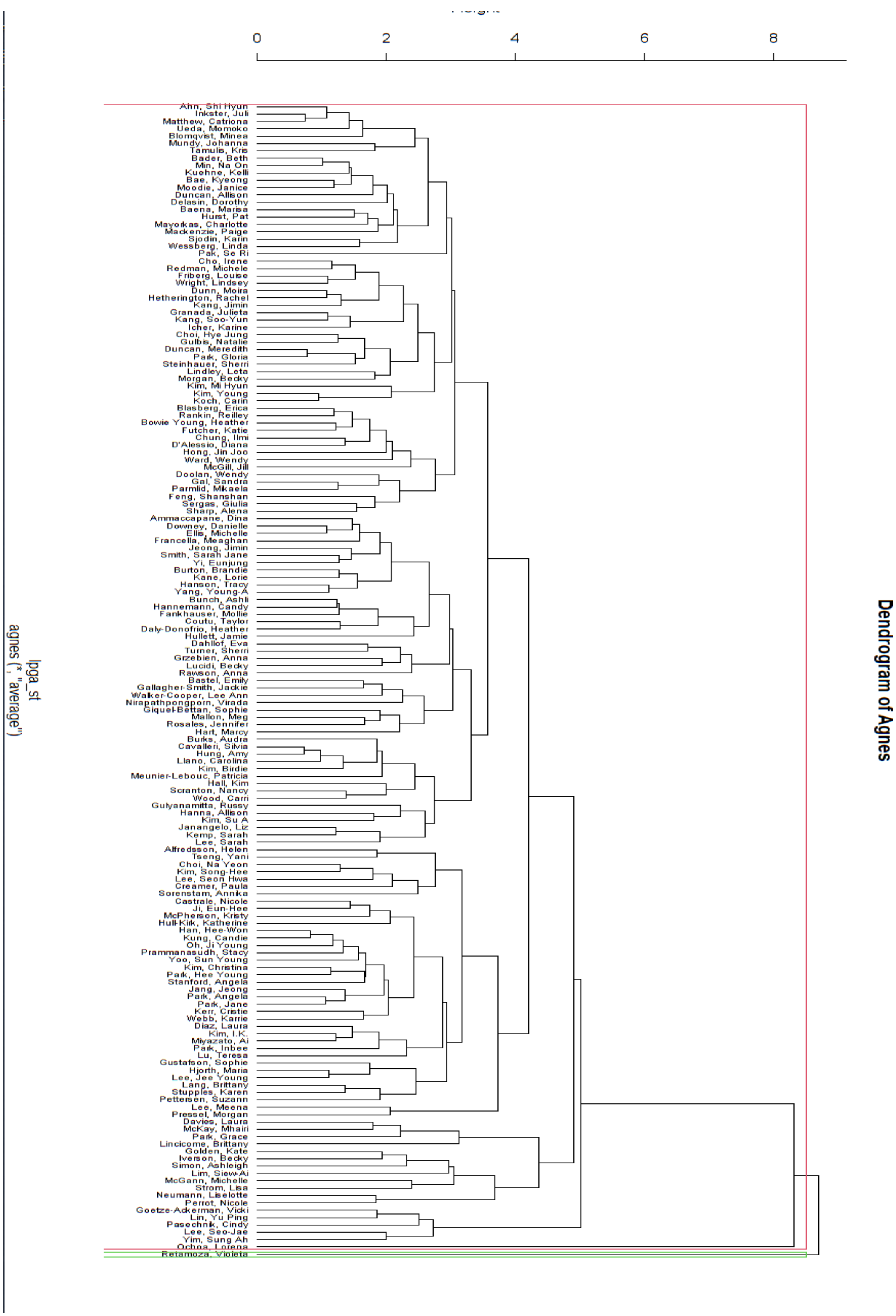
```
*******************************************************************
* Among all indices:
* 7 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 3 proposed 4 as the best number of clusters
* 4 proposed 5 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 3 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2
```



Dindex Values
2.4
2.2
2.0
1.8
1.6
2
6
10
14
Number of clusters
Second differences Dindex Values
0.04
0.03
0.02
0.01
0.00
-0.01
-0.02
2
6
10
14
Number of clusters

```r
# ===================== Agglomerative (Agnes) clustering =================

# Build the clusters in the Agglomerative method using the agnes() function
aclusters = agnes(lpga_st,metric = 'euclidean',stand=FALSE)

# Agglomerative coefficient
aclusters$ac
```

```
> # Agglomerative coefficient
> aclusters$ac
[1] 0.8088495
>
```

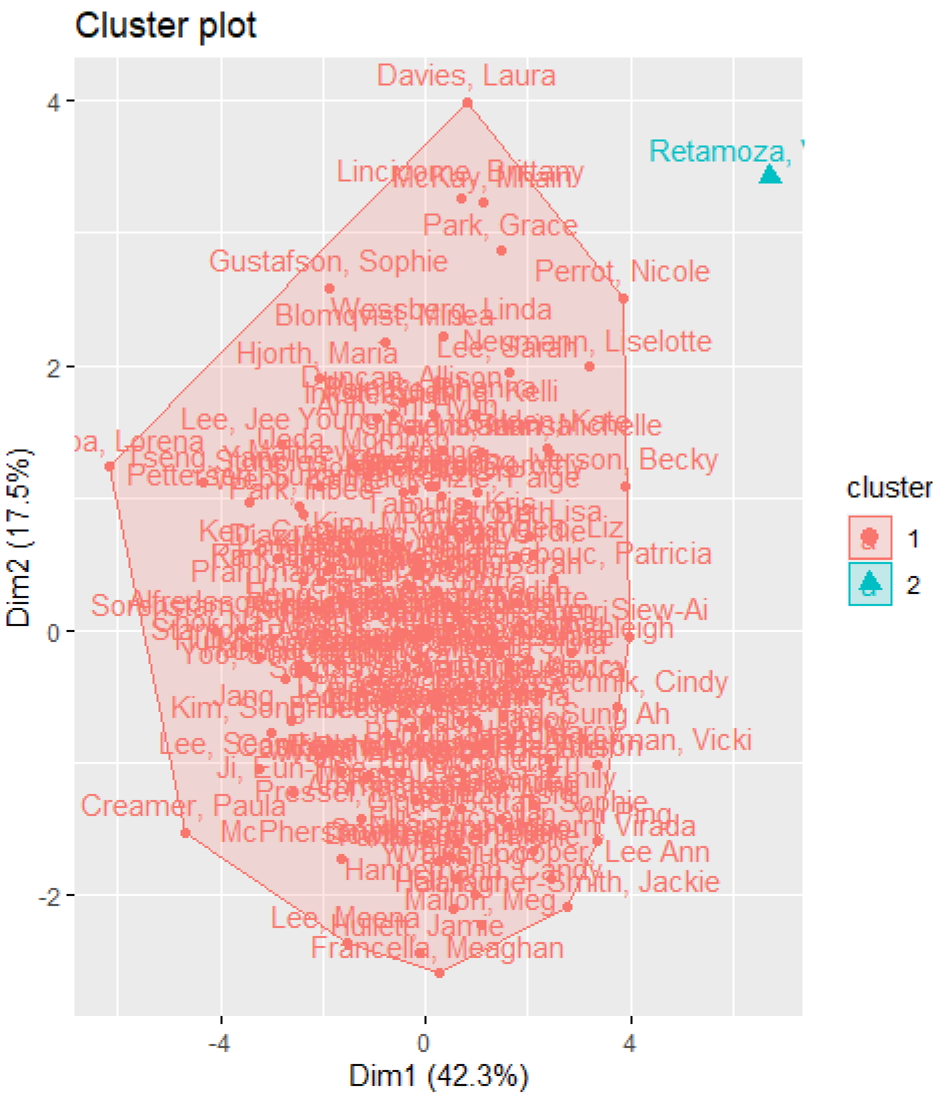


Dendrogram of Agnes
Height
0
2
4
6
8
Ahn, Shi Hyun
Inkster, Juli
Matthew, Catriona
Ueda, Momoko
Blomqvist, Minea
Mundy, Johanna
Tamulis, Kris
Bader, Beth
Min, Na On
Kuehne, Kelli
Bae, Kyeong
Moodie, Janice
Duncan, Allison
Delasin, Dorothy
Baena, Marisa
Hurst, Pat
Mayorkas, Charlotte
Mackenzie, Paige
Sjodin, Karin
Wessberg, Linda
Pak, Se Ri
Cho, Irene
Redman, Michele
Friberg, Louise
Wright, Lindsey
Dunn, Moira
Hetherington, Rachel
Kang, Jimin
Granada, Julieta
Kang, Soo-Yun
Icher, Karine
Choi, Hye Jung
Gulbis, Natalie
Duncan, Meredith
Park, Gloria
Steinhauer, Sherri
Lindley, Leta
Morgan, Becky
Kim, Mi Hyun
Kim, Young
Koch, Carin
Blasberg, Erica
Rankin, Reilley
Bowie Young, Heather
Futcher, Katie
Chung, Ilmi
D'Alessio, Diana
Hong, Jin Joo
Ward, Wendy
McGill, Jill
Doolan, Wendy
Gal, Sandra
Parmlid, Mikaela
Feng, Shanshan
Sergas, Giulia
Sharp, Alena
Ammaccapane, Dina
Downey, Danielle
Ellis, Michelle
Francella, Meaghan
Jeong, Jimin
Smith, Sarah Jane
Yi, Eunjung
Burton, Brandie
Kane, Lorie
Hanson, Tracy
Yang, Young-A
Bunch, Ashli
Hannemann, Candy
Fankhauser, Mollie
Coutu, Taylor
Daly-Donofrio, Heather
Hullett, Jamie
Dahllof, Eva
Turner, Sherri
Grzebien, Anna
Lucidi, Becky
Rawson, Anna
Bastel, Emily
Gallagher-Smith, Jackie
Walker-Cooper, Lee Ann
Nirapathpongporn, Virada
Giquel-Bettan, Sophie
Mallon, Meg
Rosales, Jennifer
Hart, Marcy
Burks, Audra
Cavalleri, Silvia
Hung, Amy
Llano, Carolina
Kim, Birdie
Meunier-Lebouc, Patricia
Hall, Kim
Scranton, Nancy
Wood, Carri
Gulyanamitta, Russy
Hanna, Allison
Kim, Su A
Janangelo, Liz
Kemp, Sarah
Lee, Sarah
Alfredsson, Helen
Tseng, Yani
Choi, Na Yeon
Kim, Song-Hee
Lee, Seon Hwa
Creamer, Paula
Sorenstam, Annika
Castrale, Nicole
Ji, Eun-Hee
McPherson, Kristy
Hull-Kirk, Katherine
Han, Hee-Won
Kung, Candie
Oh, Ji Young
Prammanasudh, Stacy
Yoo, Sun Young
Kim, Christina
Park, Hee Young
Stanford, Angela
Jang, Jeong
Park, Angela
Park, Jane
Kerr, Cristie
Webb, Karrie
Diaz, Laura
Kim, I.K.
Miyazato, Ai
Park, Inbee
Lu, Teresa
Gustafson, Sophie
Hjorth, Maria
Lee, Jee Young
Lang, Brittany
Stupples, Karen
Pettersen, Suzann
Lee, Meena
Pressel, Morgan
Davies, Laura
McKay, Mhairi
Park, Grace
Lincicome, Brittany
Golden, Kate
Iverson, Becky
Simon, Ashleigh
Lim, Siew-Ai
McGann, Michelle
Strom, Lisa
Neumann, Liselotte
Perrot, Nicole
Goetze-Ackerman, Vicki
Lin, Yu Ping
Pasechnik, Cindy
Lee, Seo-Jae
Yim, Sung Ah
Ochoa, Lorena
Retamoza, Violeta
lpga_st
agnes (*, "average")

```r
# Ploting Agnes Dedogram
pltree(aclusters, cex=0.6, hang=-1, main = 'Dendogram of Agnes')

# Indicating rectangles on the plot to visualize 2 clusters
rect.hclust(aclusters, k=2, border = 2:5)
```

```r
# Grouping clusters using cutree() function
grp_agnes = cutree(aclusters, k=2)

# Forming table to see the cluster size
table(grp_agnes)

# Visualization using fviz_cluster
fviz_cluster(list(data=lpga_st,cluster=grp_agnes))
```

```
> table(grp_agnes)
grp_agnes
  1   2
156   1
```

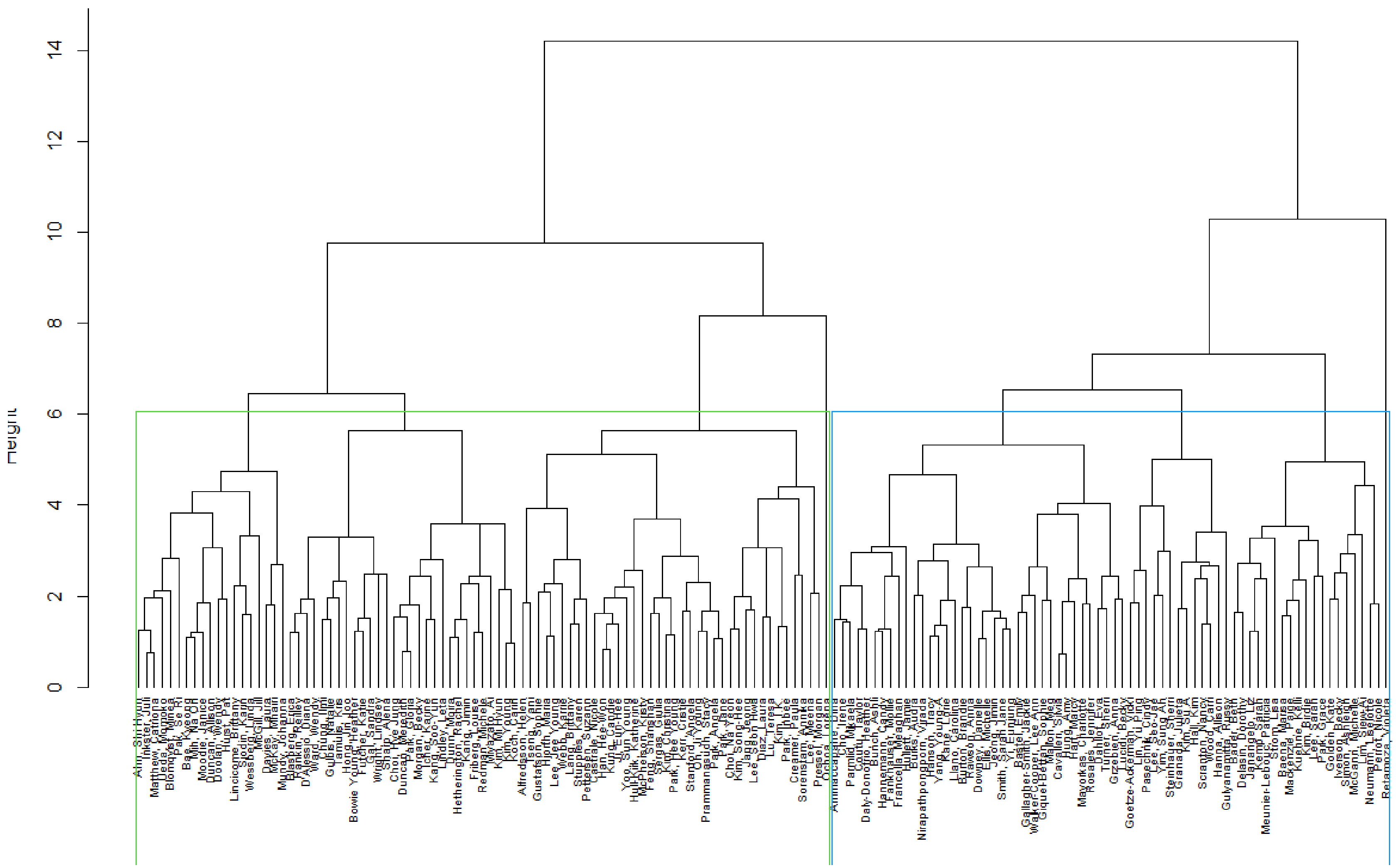



```
# ==================== Divisive (Diana) clustering ================

# Build the clusters in the Divisive method using the diana() function
diclusters = diana(lpga_st,metric = 'euclidean',stand=FALSE)

# Divisive coefficient
diclusters$dc
```

```
> # Divisive coefficient
> diclusters$dc
[1] 0.8719543
>
```

```
# Ploting Diana Dedogram
pltree(diclusters, cex=0.6, hang=-1, main = 'Dendogram of Diana')

# Indicating rectangles on the plot to visualize 2 clusters
rect.hclust(diclusters, k=2, border = 3:5)
```

Dendrogram of Diana



lpga_st
diana (*, "NA")

```r
# Grouping clusters using cutree() function
grp_diana = cutree(diclusters, k=2)

# Forming table to see the cluster size
table(grp_diana)

# Visualization using fviz_cluster
fviz_cluster(list(data=lpga_st,cluster=grp_diana))
```

```
> table(grp_diana)
grp_diana
 1  2 
87 70 
```