CS544_HW1_Huang

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Part1

The data set rivers contains the lengths (in miles) of "major" rivers in North America, as compiled by the US Geological Survey. Use the data set to answer the following questions using R:

```
data("rivers")
rivers
```

```
##   [1]  735  320  325  392  524  450 1459  135  465  600  330  336  280  315
##  [15]  870  906  202  329  290 1000  600  505 1450  840 1243  890  350  407
##  [29]  286  280  525  720  390  250  327  230  265  850  210  630  260  230
##  [43]  360  730  600  306  390  420  291  710  340  217  281  352  259  250
##  [57]  470  680  570  350  300  560  900  625  332 2348 1171 3710 2315 2533
##  [71]  780  280  410  460  260  255  431  350  760  618  338  981 1306  500
##  [85]  696  605  250  411 1054  735  233  435  490  310  460  383  375 1270
##  [99]  545  445 1885  380  300  380  377  425  276  210  800  420  350  360
## [113]  538 1100 1205  314  237  610  360  540 1038  424  310  300  444  301
## [127]  268  620  215  652  900  525  246  360  529  500  720  270  430  671
## [141] 1770
```

a) How many data points are there in the data set?

```
numberOfData_rivers <- length(rivers)
numberOfData_rivers
```

```
## [1] 141
```

b) Compute the mean, median, and mode.

```
mean_rivers <- mean(rivers)
median_rivers <- median(rivers)
mode_rivers <- as.integer(names(sort(-table(rivers)))[1])

mean_rivers
```

```
## [1] 591.1844
```

```
median_rivers
```

```
## [1] 425
```

```
mode_rivers
```

```
## [1] 350
```

c) Compute the variance and the standard deviation

```
var_rivers <- var(rivers)
sd_rivers <- sd(rivers)

var_rivers
```

```
## [1] 243908.4
```

```
sd_rivers
```

```
## [1] 493.8708
```

d) Compute the five number summary, the interquartile range, and outliers, if any.

```
fiveNum_rivers <- fivenum(rivers)

quantile(rivers)
```

```
##   0%  25%  50%  75% 100%
##  135  310  425  680 3710
```

```
Q1_rivers<- quantile(rivers, 0.25)
Q3_rivers<- quantile(rivers, 0.75)
IQR_rivers <- IQR(rivers)

fiveNum_rivers
```

```
## [1]  135  310  425  680 3710
```

```
Q1_rivers
```

```
## 25%
## 310
```

```
Q3_rivers
```

```
## 75%
## 680
```

```
IQR_rivers
```

```
## [1] 370
```

```
# define a function to remove outliers
FindOutliers <- function(data) {
  q1 <- quantile(data)[2]
  q3 <- quantile(data)[4]
  iqr <- IQR(data)
```

```
  threshold.upper = (iqr * 3) + q3
  threshold.lower = q1 - (iqr * 3)
  index <- which(data > threshold.upper | data < threshold.lower) #retrun positions that are TRUE in th
  return(data[index])
}

# use the function to identify outliers
outliers_rivers <- FindOutliers(rivers)
outliers_rivers
```

```
## [1] 2348 3710 2315 2533 1885
```

e) Compute the standardized version (z-scores) of the above data.

```
#method 1: use built-in function
zScore_rivers<- scale(rivers)
zScore_rivers
```

```
##               [,1]
##   [1,]  0.29120084
##   [2,] -0.54909983
##   [3,] -0.53897573
##   [4,] -0.40331273
##   [5,] -0.13603637
##   [6,] -0.28587312
##   [7,]  1.75717116
##   [8,] -0.92369170
##   [9,] -0.25550080
##  [10,]  0.01785002
##  [11,] -0.52885162
##  [12,] -0.51670270
##  [13,] -0.63009267
##  [14,] -0.55922394
##  [15,]  0.56455166
##  [16,]  0.63744521
##  [17,] -0.78802870
##  [18,] -0.53087645
##  [19,] -0.60984446
##  [20,]  0.82777837
##  [21,]  0.01785002
##  [22,] -0.17450797
##  [23,]  1.73894778
##  [24,]  0.50380703
##  [25,]  1.31980985
##  [26,]  0.60504808
##  [27,] -0.48835521
##  [28,] -0.37294042
##  [29,] -0.61794374
##  [30,] -0.63009267
##  [31,] -0.13401155
##  [32,]  0.26082852
##  [33,] -0.40736237
##  [34,] -0.69083730
```

```
##  [35,] -0.53492609
##  [36,] -0.73133371
##  [37,] -0.66046498
##  [38,]  0.52405524
##  [39,] -0.77183013
##  [40,]  0.07859464
##  [41,] -0.67058909
##  [42,] -0.73133371
##  [43,] -0.46810700
##  [44,]  0.28107673
##  [45,]  0.01785002
##  [46,] -0.57744733
##  [47,] -0.40736237
##  [48,] -0.34661774
##  [49,] -0.60781964
##  [50,]  0.24058032
##  [51,] -0.50860342
##  [52,] -0.75765639
##  [53,] -0.62806785
##  [54,] -0.48430556
##  [55,] -0.67261391
##  [56,] -0.69083730
##  [57,] -0.24537670
##  [58,]  0.17983569
##  [59,] -0.04289461
##  [60,] -0.48835521
##  [61,] -0.58959625
##  [62,] -0.06314282
##  [63,]  0.62529629
##  [64,]  0.06847054
##  [65,] -0.52480198
##  [66,]  3.55723694
##  [67,]  1.17402275
##  [68,]  6.31504300
##  [69,]  3.49041785
##  [70,]  3.93182881
##  [71,]  0.38231778
##  [72,] -0.63009267
##  [73,] -0.36686595
##  [74,] -0.26562491
##  [75,] -0.67058909
##  [76,] -0.68071319
##  [77,] -0.32434471
##  [78,] -0.48835521
##  [79,]  0.34182136
##  [80,]  0.05429679
##  [81,] -0.51265306
##  [82,]  0.78930678
##  [83,]  1.44737357
##  [84,] -0.18463207
##  [85,]  0.21223282
##  [86,]  0.02797412
##  [87,] -0.69083730
##  [88,] -0.36484113
```

```
##  [89,]  0.93711870
##  [90,]  0.29120084
##  [91,] -0.72525925
##  [92,] -0.31624543
##  [93,] -0.20488028
##  [94,] -0.56934804
##  [95,] -0.26562491
##  [96,] -0.42153612
##  [97,] -0.43773468
##  [98,]  1.37448002
##  [99,] -0.09351513
## [100,] -0.29599722
## [101,]  2.61974487
## [102,] -0.42761058
## [103,] -0.58959625
## [104,] -0.42761058
## [105,] -0.43368504
## [106,] -0.33649364
## [107,] -0.63819195
## [108,] -0.77183013
## [109,]  0.42281420
## [110,] -0.34661774
## [111,] -0.48835521
## [112,] -0.46810700
## [113,] -0.10768888
## [114,]  1.03026046
## [115,]  1.24286666
## [116,] -0.56124876
## [117,] -0.71715997
## [118,]  0.03809823
## [119,] -0.46810700
## [120,] -0.10363924
## [121,]  0.90472157
## [122,] -0.33851846
## [123,] -0.56934804
## [124,] -0.58959625
## [125,] -0.29802204
## [126,] -0.58757143
## [127,] -0.65439052
## [128,]  0.05834643
## [129,] -0.76170603
## [130,]  0.12314070
## [131,]  0.62529629
## [132,] -0.13401155
## [133,] -0.69893658
## [134,] -0.46810700
## [135,] -0.12591227
## [136,] -0.18463207
## [137,]  0.26082852
## [138,] -0.65034088
## [139,] -0.32636954
## [140,]  0.16161230
## [141,]  2.38689046
## attr(,"scaled:center")
```

```
## [1] 591.1844
## attr(,"scaled:scale")
## [1] 493.8708
```

```
#method 2: define a function to compute z-scores of data
computeZscore <- function(data) {
  mean <- mean(data)
  sd <- sd(data)
  zScore <- (data-mean)/sd
  return(zScore)
}
zScore_rivers <- computeZscore (rivers)
zScore_rivers
```

```
##   [1]  0.29120084 -0.54909983 -0.53897573 -0.40331273 -0.13603637
##   [6] -0.28587312  1.75717116 -0.92369170 -0.25550080  0.01785002
##  [11] -0.52885162 -0.51670270 -0.63009267 -0.55922394  0.56455166
##  [16]  0.63744521 -0.78802870 -0.53087645 -0.60984446  0.82777837
##  [21]  0.01785002 -0.17450797  1.73894778  0.50380703  1.31980985
##  [26]  0.60504808 -0.48835521 -0.37294042 -0.61794374 -0.63009267
##  [31] -0.13401155  0.26082852 -0.40736237 -0.69083730 -0.53492609
##  [36] -0.73133371 -0.66046498  0.52405524 -0.77183013  0.07859464
##  [41] -0.67058909 -0.73133371 -0.46810700  0.28107673  0.01785002
##  [46] -0.57744733 -0.40736237 -0.34661774 -0.60781964  0.24058032
##  [51] -0.50860342 -0.75765639 -0.62806785 -0.48430556 -0.67261391
##  [56] -0.69083730 -0.24537670  0.17983569 -0.04289461 -0.48835521
##  [61] -0.58959625 -0.06314282  0.62529629  0.06847054 -0.52480198
##  [66]  3.55723694  1.17402275  6.31504300  3.49041785  3.93182881
##  [71]  0.38231778 -0.63009267 -0.36686595 -0.26562491 -0.67058909
##  [76] -0.68071319 -0.32434471 -0.48835521  0.34182136  0.05429679
##  [81] -0.51265306  0.78930678  1.44737357 -0.18463207  0.21223282
##  [86]  0.02797412 -0.69083730 -0.36484113  0.93711870  0.29120084
##  [91] -0.72525925 -0.31624543 -0.20488028 -0.56934804 -0.26562491
##  [96] -0.42153612 -0.43773468  1.37448002 -0.09351513 -0.29599722
## [101]  2.61974487 -0.42761058 -0.58959625 -0.42761058 -0.43368504
## [106] -0.33649364 -0.63819195 -0.77183013  0.42281420 -0.34661774
## [111] -0.48835521 -0.46810700 -0.10768888  1.03026046  1.24286666
## [116] -0.56124876 -0.71715997  0.03809823 -0.46810700 -0.10363924
## [121]  0.90472157 -0.33851846 -0.56934804 -0.58959625 -0.29802204
## [126] -0.58757143 -0.65439052  0.05834643 -0.76170603  0.12314070
## [131]  0.62529629 -0.13401155 -0.69893658 -0.46810700 -0.12591227
## [136] -0.18463207  0.26082852 -0.65034088 -0.32636954  0.16161230
## [141]  2.38689046
```

f) Create a matrix of size 2 x 30 using the first 60 data points in rivers. The first 30 values belong to the first row of the matrix. Assign the result to the variable,rivers.60, and display the result.

```
rivers.60 <- matrix(head(rivers, n=60), nrow=2, ncol=30, byrow=TRUE )
rivers.60
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]  735  320  325  392  524  450 1459  135  465   600   330   336   280
## [2,]  525  720  390  250  327  230  265  850  210   630   260   230   360
```

```
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]   315   870   906   202   329   290  1000   600   505  1450   840
## [2,]   730   600   306   390   420   291   710   340   217   281   352
##      [,25] [,26] [,27] [,28] [,29] [,30]
## [1,]  1243   890   350   407   286   280
## [2,]   259   250   470   680   570   350
```

g) Without hardcoding, displaying the first and last columns of the matrix.

```
firstCol_riversMatrix <- rivers.60[,1]
lastCol_riversMatrix <- rivers.60[,30]

firstCol_riversMatrix
```

```
## [1] 735 525
```

```
lastCol_riversMatrix
```

```
## [1] 280 350
```

h) Assign row names for the rivers.60 as Row_1 and Row_2 and column names as Length_1, Length_2, ...Length_30. The code should not hard code the values of the numbers in the row and column names.

```
rowNames_rivers.60 <- c("Row_1", "Row_2")
colNames_rivers.60 <- paste("Length",1:30, sep="_")

dimnames(rivers.60) <- list(rowNames_rivers.60, colNames_rivers.60)
rivers.60
```

```
##       Length_1 Length_2 Length_3 Length_4 Length_5 Length_6 Length_7
## Row_1      735      320      325      392      524      450     1459
## Row_2      525      720      390      250      327      230      265
##       Length_8 Length_9 Length_10 Length_11 Length_12 Length_13 Length_14
## Row_1      135      465       600       330       336       280       315
## Row_2      850      210       630       260       230       360       730
##       Length_15 Length_16 Length_17 Length_18 Length_19 Length_20
## Row_1       870       906       202       329       290      1000
## Row_2       600       306       390       420       291       710
##       Length_21 Length_22 Length_23 Length_24 Length_25 Length_26
## Row_1       600       505      1450       840      1243       890
## Row_2       340       217       281       352       259       250
##       Length_27 Length_28 Length_29 Length_30
## Row_1       350       407       286       280
## Row_2       470       680       570       350
```

Part2

The data file Johnson.csv contains quarterly earnings (dollars) per Johnson & Johnson share 1960–80.

a) Read the data from johnson.csv into a data frame. In the data frame, the data in "Year" column should be used as row names and "Qtr1", 'Qtr2", "Qtr3", and "Qtr4" should be column names.

```
johnson<- read.csv("Johnson.csv", row.names=1)
johnson
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1960  0.71  0.63  0.85  0.44
## 1961  0.61  0.69  0.92  0.55
## 1962  0.72  0.77  0.92  0.60
## 1963  0.83  0.80  1.00  0.77
## 1964  0.92  1.00  1.24  1.00
## 1965  1.16  1.30  1.45  1.25
## 1966  1.26  1.38  1.86  1.56
## 1967  1.53  1.59  1.83  1.86
## 1968  1.53  2.07  2.34  2.25
## 1969  2.16  2.43  2.70  2.25
## 1970  2.79  3.42  3.69  3.60
## 1971  3.60  4.32  4.32  4.05
## 1972  4.86  5.04  5.04  4.41
## 1973  5.58  5.85  6.57  5.31
## 1974  6.03  6.39  6.93  5.85
## 1975  6.93  7.74  7.83  6.12
## 1976  7.74  8.91  8.28  6.84
## 1977  9.54 10.26  9.54  8.73
## 1978 11.88 12.06 12.15  8.91
## 1979 14.04 12.96 14.85  9.99
## 1980 16.20 14.67 16.02 11.61
```

b) Show the summary for earnings for each quarter.

```
summary(johnson)
```

```
##       Qtr1             Qtr2             Qtr3             Qtr4
##  Min.   : 0.610   Min.   : 0.630   Min.   : 0.850   Min.   : 0.440
##  1st Qu.: 1.160   1st Qu.: 1.300   1st Qu.: 1.450   1st Qu.: 1.250
##  Median : 2.790   Median : 3.420   Median : 3.690   Median : 3.600
##  Mean   : 4.791   Mean   : 4.966   Mean   : 5.254   Mean   : 4.188
##  3rd Qu.: 6.930   3rd Qu.: 7.740   3rd Qu.: 7.830   3rd Qu.: 6.120
##  Max.   :16.200   Max.   :14.670   Max.   :16.020   Max.   :11.610
```

```
sumOfQuarterEarnings<-c(sum(johnson$Qtr1),sum(johnson$Qtr2),sum(johnson$Qtr3),sum(johnson$Qtr4))

names(sumOfQuarterEarnings) <-c("Qtr1","Qtr2","Qtr3","Qtr4")
sumOfQuarterEarnings
```

```
##   Qtr1   Qtr2   Qtr3   Qtr4
```

```
## 100.62 104.28 110.33  87.95
```

c) Add a new column, Yearly, showing the earnings for the whole year (the sum of earnings for the 4 quarters). Display the new resulting data frame.

```
johnson$Yearly <- johnson$Qtr1+ johnson$Qtr2+johnson$Qtr3+johnson$Qtr4
johnson
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4 Yearly
## 1960  0.71  0.63  0.85  0.44   2.63
## 1961  0.61  0.69  0.92  0.55   2.77
## 1962  0.72  0.77  0.92  0.60   3.01
## 1963  0.83  0.80  1.00  0.77   3.40
## 1964  0.92  1.00  1.24  1.00   4.16
## 1965  1.16  1.30  1.45  1.25   5.16
## 1966  1.26  1.38  1.86  1.56   6.06
## 1967  1.53  1.59  1.83  1.86   6.81
## 1968  1.53  2.07  2.34  2.25   8.19
## 1969  2.16  2.43  2.70  2.25   9.54
## 1970  2.79  3.42  3.69  3.60  13.50
## 1971  3.60  4.32  4.32  4.05  16.29
## 1972  4.86  5.04  5.04  4.41  19.35
## 1973  5.58  5.85  6.57  5.31  23.31
## 1974  6.03  6.39  6.93  5.85  25.20
## 1975  6.93  7.74  7.83  6.12  28.62
## 1976  7.74  8.91  8.28  6.84  31.77
## 1977  9.54 10.26  9.54  8.73  38.07
## 1978 11.88 12.06 12.15  8.91  45.00
## 1979 14.04 12.96 14.85  9.99  51.84
## 1980 16.20 14.67 16.02 11.61  58.50
```

d) Which was the best performing year (in terms of highest earning) and worst performing year?

```
bestYear <- rownames(johnson)[which.max(johnson$Yearly)]
worstYear <- rownames(johnson)[which.min(johnson$Yearly)]

bestYear
```

```
## [1] "1980"
```

```
worstYear
```

```
## [1] "1960"
```

e) Show all rows of the data frame whose "Yearly" is greater than 20.

```
goodYears<- subset(johnson, Yearly >20, select=Qtr1:Yearly)
goodYears
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4 Yearly
```

```
## 1973  5.58  5.85  6.57  5.31  23.31
## 1974  6.03  6.39  6.93  5.85  25.20
## 1975  6.93  7.74  7.83  6.12  28.62
## 1976  7.74  8.91  8.28  6.84  31.77
## 1977  9.54 10.26  9.54  8.73  38.07
## 1978 11.88 12.06 12.15  8.91  45.00
## 1979 14.04 12.96 14.85  9.99  51.84
## 1980 16.20 14.67 16.02 11.61  58.50
```