STAT 385 Homework Assignment 02

Tianli Ding

Due by 12:00 PM 09/28/2019

Grader Comment

Efficiency:

The code and document is written with lots of logical thought about the tasks with few unnecessary or unnecessarily verbose lines and is written in a least brute force way.

Correctness:

The files run with no errors and the coding is written in a most accurate way.

Documentation:

The explanations and comments contain reasonable clarity and exhibit mid-tier (to be expected) fundamentals of usefulness, grammar, smoothness, and structure.

Beauty:

The document is most visually appealing and has the strongest readability. The coding output and visualizations are very attractive.

Please note that the summary above applies to the completed work. If the submission was incomplete, the ultimate mark for the submission will be scaled down proportionally.

Please take care to ensure that one's repo includes the image files that are included in one's .Rmd file. Otherwise, the graded file that is returned to the student may not contain the images.

You may find that your returned work (.Rmd and .pdf) has had sections removed. This is only for accelerating the grading process by removing code chunks that are impeding knitting. If you believe that you were incorrectly penalized for missing work, please do contact the TA and/or professor.

Please see comments below for specifics.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

The Homework Problems

Below you will find problems for you to complete as an individual. It is fine to discuss the homework p

1. Do the following:

a. create two vectors labelled x and y, each with 500 elements. You can do this any way you like. Combine these vectors into a matrix called mat. Run the command head(mat).

```r
x = c(1:500)
y = c(501:1000)
mat = cbind(x, y)
head(mat)
```

```
##      x   y
## [1,] 1 501
## [2,] 2 502
## [3,] 3 503
## [4,] 4 504
## [5,] 5 505
## [6,] 6 506
```

b. run the command z <- rep(c(1,3), each = 250). Combine mat and z. Convert the matrix mat with columns x, y, and z into a data frame called dat.

```r
z <- rep(c(1,3), each = 250)
newc = cbind(mat, z)
dat = data.frame(newc)
```

c. reassign z as a factor variable within the data frame dat. Run the command str(dat). You should see something like:

```
'data.frame':   500 obs. of  3 variables:<br>
$ x: num  1.4176 1.25 -1.4528 -0.3946 0.0264 ...<br>
$ y: num  0.0736 -0.193 -0.3286 -0.9563 -0.1975 ...<br>
$ z: Factor w/ 2 levels "1","3": 1 1 1 1 1 1 1 1 1 ...
```

```
dat$z = factor(dat$z)
is.factor(dat$z)
```

```
## [1] TRUE
```

```
str(dat)
```

```
## 'data.frame':    500 obs. of  3 variables:
##  $ x: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ y: num  501 502 503 504 505 506 507 508 509 510 ...
##  $ z: Factor w/ 2 levels "1","3": 1 1 1 1 1 1 1 1 1 ...
```

d. run the following commands:

```
2 + dat$z
2 + as.numeric(dat$z)
2 + as.numeric(levels(dat$z)[dat$z])
```

Carefully describe why the first line returned a warning message and both the second and third lines did not. Which of the three lines properly added 2 to the levels of the factor variable z? Why?

```
2 + dat$z
```

```
## Warning in Ops.factor(2, dat$z): '+' not meaningful for factors
```

```
##   [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [24] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [47] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [70] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [93] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [116] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [139] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [162] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [185] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [208] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [231] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [254] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [277] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [300] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [323] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [346] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [369] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [392] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [415] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [438] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [461] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [484] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
```

```
    2 + as.numeric(dat$z)
```

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [211] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [246] 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [281] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [316] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [351] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [386] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [421] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [456] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [491] 4 4 4 4 4 4 4 4 4 4
```

```
    2 + as.numeric(levels(dat$z)[dat$z])
```

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [211] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [246] 3 3 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [281] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [316] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [351] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [386] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [421] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [456] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [491] 5 5 5 5 5 5 5 5 5 5
```

The first line returned a warning since `dat$z` is a factor, and it cannot be calculated as numeric. Both the second and third lines add 2 to numeric number so it is valid operation. The third line properly added 2 to the levels of the factor variable `z` since the third line pull out the levels of `dat$z` and then add 2.

e. remove the rows of `dat` corresponding to lowest 100 values of `x`.

```
dat = dat[order(dat$x)[101:500], ]
```

2. With the `dat` dataset in problem 1, do the following:

a. run the commands:

```
set.seed(13)
e <- rnorm(400)
dat$x <- rnorm(400)
dat$y <- dat$x * 2 + as.numeric(levels(dat$z)[dat$z]) + e
```

```
zlist <- split(dat, f = dat$z)
first = mean(dat$y[levels(dat$z)[dat$z] == 1])
second = mean(dat$y[levels(dat$z)[dat$z] == 3])
first < second
```

```
## [1] TRUE
```

Determine which level of z has a higher mean in y. Report your results.

Level 3 of z has a higher mean in y than level 1.

b. The fourth line of the code in part a generates y as a linear function of the numeric variable x and the factor variable z. Can you determine that the linear modeling assumptions are satisfied for the regression of y on x and z? Explain why or why not in detail. **Hint**: read the documentation of the `rnorm` function and examine its defaults. The linear modeling assumptions are: 1) there is a linear relationship between the response variable and the predictor variables; 2) there is no auto-correlation (cases are independent); 3) errors are normally distributed; 4) the standard deviation of the errors are constant across all predictor variables (the values of x and the levels of z in this problem). There is no need to use numerical or graphical summaries to verify whether or not these modeling assumptions hold, all of the necessary information is contained in the code chunk in part **a**.

Solution: the linear modeling assumptions are satisfied for the regression of y on x and z. There is a linear relationship between the response variable and the predictor variables and there is no correlations. The errors are random and normally distributed, and the standard deviation of the errors are constant across all predictor variables.

3. Using functions, lists, matrices and vectors, complete the following:

a. place the first 5 letters of the alphabet into a list

```
five = list(letters[1:5])
```

b. place a random set of 2 integers as a new component of the list made in part **a**

```
append(five, c(1,2))
```

```
## [[1]]
## [1] "a" "b" "c" "d" "e"
##
## [[2]]
## [1] 1
##
## [[3]]
## [1] 2
```

c. create a matrix with two columns such that column 1 contains the 5 letters repeating, column 2 contains the 2 integers repeating, and column 3 that shows all possible pairs of the elements of the first 2 columns

```
rep1 = rep(letters[1:5], each = 2)
rep2 = rep(c(1,2), 5)
com3 = cbind(rep(letters[1:5], each = 2), rep(letters[1:5], each = 2), paste(rep1, rep2))
com3
```

```
##       [,1] [,2] [,3]
```

```
##  [1,] "a"  "a"  "a 1"
##  [2,] "a"  "a"  "a 2"
##  [3,] "b"  "b"  "b 1"
##  [4,] "b"  "b"  "b 2"
##  [5,] "c"  "c"  "c 1"
##  [6,] "c"  "c"  "c 2"
##  [7,] "d"  "d"  "d 1"
##  [8,] "d"  "d"  "d 2"
##  [9,] "e"  "e"  "e 1"
## [10,] "e"  "e"  "e 2"
```

d. add a 4th column to the matrix from part **c** that is a randomized ordering of the pairs of the 3rd column

```
cbind(com3, sample(paste(rep1, rep2)))
```

```
##       [,1] [,2] [,3]  [,4]
##  [1,] "a"  "a"  "a 1" "c 1"
##  [2,] "a"  "a"  "a 2" "b 1"
##  [3,] "b"  "b"  "b 1" "c 2"
##  [4,] "b"  "b"  "b 2" "a 1"
##  [5,] "c"  "c"  "c 1" "d 2"
##  [6,] "c"  "c"  "c 2" "e 2"
##  [7,] "d"  "d"  "d 1" "e 1"
##  [8,] "d"  "d"  "d 2" "b 2"
##  [9,] "e"  "e"  "e 1" "a 2"
## [10,] "e"  "e"  "e 2" "d 1"
```

4. Use the CCSO Bookings Data to accomplish the following:

a. read in the data using R's default method for comma separated files and show the data dimension

```
ccso = read.csv("https://uofi.box.com/shared/static/9elozjsg99bgcb7gb546wlfr3r2gc9b7.csv")
dim(ccso)
```

```
## [1] 67764    35
```

b. show only the 2012 bookings for people ages 17-23 years old not residing in Illinois and show the da

```
newb = filter(ccso,  Age.at.Arrest >= 17 & Age.at.Arrest <= 23 & ccso$STATE != "ILLINOIS" & as.Date(ccs
dim(newb)
```

```
## [1] 102  35
```

c. show only the bookings for people who have employment status as "student" booked after the year 2012

```
newc = filter(ccso, EMPLOYMENT.STATUS == "Student" & CITY == "DANVILLE" & as.Date(ccso$BOOKING.DATE, "%
dim(newc)
```

```
## [1] 22 35
```

d. show only the bookings for Asian people residing in the cities of Champaign or Urbana and show the da

```
newd = filter(ccso, CITY == "CHAMPAIGN" | CITY == "URBANA", RACE == "Asian/Pacific Islander" )
dim(newd)
```

```
## [1] 593  35
```

5. Use the CCSO Bookings Data to accomplish the following:

a. read in the data using the **data.table** package and show the data dimension

```
library(data.table)
ccso2 = fread("https://uofi.box.com/shared/static/9elozjsg99bgcb7gb546wlfr3r2gc9b7.csv")
```

```
## Warning in require_bit64(): Some columns are type 'integer64' but
## package bit64 is not installed. Those columns will print as strange
## looking floating point data. There is no need to reload the data. Simply
## install.packages('bit64') to obtain the integer64 print method and print
## the data again.
```

```
dim(ccso2)
```

```
## [1] 67764    35
```

b. show only the 2012 bookings for people ages 17-23 years old not residing in Illinois and show the da

```
bb = filter(ccso2, `Age at Arrest` >= 17 & `Age at Arrest` <= 23 & ccso2$STATE != "ILLINOIS" & as.Date(
dim(bb)
```

```
## [1] 102  35
```

c. show only the bookings for people who have employment status as "student" booked after the year 2012

```
bc = filter(ccso2, ccso2$`EMPLOYMENT STATUS` == "Student" & as.Date(ccso2$`BOOKING DATE`, "%m/%d/%Y") >
dim(bc)
```

```
## [1] 22 35
```

d. show only the bookings for Asian people residing in the cities of Champaign or Urbana and show the da

```
bd = filter(ccso2, ccso2$RACE == "Asian/Pacific Islander" & (CITY == "CHAMPAIGN"| CITY == "URBANA") )
dim(bd)
```

```
## [1] 593  35
```