STAT 2200: Problem Set 2

Jack Ambery

Due: Monday, 2/12 at the beginning of class

- You may discuss this assignment with other students in the class, but you may not sit down and type it up with them or show them your code. You also may not discuss the assignment with anyone who isn't in our class, nor may you look up anything online.
- You must type up your homework using R Markdown. I want to see all of your code and output, and any answers you provide that aren't code must be typed above the respective `R` code chunk.
- Make sure none of your code runs off the page, otherwise you will lose points.
- You must print and turn in a PDF of your homework. I won't accept anything else (e.g., a Word document). All pages must be stapled together.
- This assignment is worth 100 points.

Part 1

In this part you will practice working with vectors and strings.

1. Create and store a vector that consists of the following movie lengths (in minutes): 94, 109, 110, 123, 125, 108, 92, 106, 84, 119, 110, and 140.

```
lengths <- c(94, 109, 110, 123, 125, 108, 92, 106, 84,
119, 110, 140)
```

2. Using your entire movie length vector from problem 1, as well as the `cat()` function, reproduce the output below *exactly* as it's displayed. Note that the `round()` function allows you to round to a specified number of decimal places. Also note that you may not manually type the number 110.

Mean movie length (in min):
110 minutes

```r
cat("Mean movie length (in min):\n",mean(lengths),"minutes")
```

```
## Mean movie length (in min):
##  110 minutes
```

3. Using your entire movie length vector from problem 1, as well as the `paste()` and/or `paste0()` functions, reproduce the output below *exactly* as it's displayed. You may not manually type the number 110.

"The mean movie length is 110 minutes."

```r
paste0("The mean movie length is ",mean(lengths)," minutes.")
```

```
## [1] "The mean movie length is 110 minutes."
```

Part 2

In this part you will practice working with data frames that you will create manually.

1. Create and print a data frame that consists of the following variables and values for recipes:

   - type of recipe: entree, appetizer, appetizer, entree, entree, appetizer, dessert, dessert, entree, entree,
   - number of ingredients: 8, 4, 5, 10, 6, 8, 7, 15, 10, 9,
   - total prep time (in min): 15, 15, 5, 35, 20, 40, 25, 30, 10, 20,
   - total cook time (in min): 30, 15, 20, 55, 25, 10, 120, 25, 45, 60, and
   - contains meat (yes/no): yes, yes, yes, no, no, yes, no, no, yes, yes

Note: Two of the variables should be factors! Also, make sure your variable names are descriptive but not too long.

```
recipies <- data.frame(
  type = factor(c("entree", "appetizer", "appetizer", "entree", "entree", "appetizer", "
  ingredients = c(8, 4, 5, 10, 6, 8, 7, 15, 10, 9),
  prepTime = c(15, 15, 5, 35, 20, 40, 25, 30, 10, 20),
  cookTime = c(30, 15, 20, 55, 25, 10, 120, 25, 45, 60),
  hasMeat = factor(c("yes", "yes", "yes", "no", "no", "yes", "no", "no", "yes", "yes")
))
recipies
```

```
##         type ingredients prepTime cookTime hasMeat
## 1     entree           8       15       30     yes
## 2  appetizer           4       15       15     yes
## 3  appetizer           5        5       20     yes
## 4     entree          10       35       55      no
## 5     entree           6       20       25      no
## 6  appetizer           8       40       10     yes
## 7    dessert           7       25      120      no
## 8    dessert          15       30       25      no
## 9     entree          10       10       45     yes
## 10    entree           9       20       60     yes
```

2. Create a new variable that measures the total time (prep + cook) it takes to make each recipe. Add this new variable to the data frame from problem 1, and then print the updated data frame. You may not manually type the numbers.

```
recipies$totalTime <- recipies$prepTime + recipies$cookTime
recipies
```

```
##          type ingredients prepTime cookTime hasMeat totalTime
## 1     entree           8       15       30     yes        45
## 2  appetizer           4       15       15     yes        30
## 3  appetizer           5        5       20     yes        25
## 4     entree          10       35       55      no        90
## 5     entree           6       20       25      no        45
## 6  appetizer           8       40       10     yes        50
## 7    dessert           7       25      120      no       145
## 8    dessert          15       30       25      no        55
## 9     entree          10       10       45     yes        55
## 10    entree           9       20       60     yes        80
```

3. Create and print a second data frame that consists of the following variables and values for four additional recipes:

- type of recipe: appetizer, entree, dessert, appetizer
- number of ingredients: 3, 15, 8, 5,
- total prep time (in min): 10, 35, 45, 10,
- total cook time (in min): 0, 90, 150, 20, and
- contains meat (yes/no): no, no, no, yes

```
recipies2 <- data.frame(
  type = factor(c("appetizer", "entree", "dessert", "appetizer")),
  ingredients = c(3, 15, 8, 5),
  prepTime = c(10, 35, 45, 10),
  cookTime = c(0, 90, 150, 20),
  hasMeat = factor(c("no", "no", "no", "yes"))
)
recipies2
```

```
##        type ingredients prepTime cookTime hasMeat
## 1 appetizer           3       10        0      no
## 2    entree          15       35       90      no
## 3   dessert           8       45      150      no
## 4 appetizer           5       10       20     yes
```

4. Create the same total time variable you created in problem 2 for this new data set, and then print the updated data frame.

```
recipies2$totalTime <- recipies2$prepTime + recipies2$cookTime
recipies2
```

```
##        type ingredients prepTime cookTime hasMeat totalTime
## 1 appetizer           3       10        0      no        10
## 2    entree          15       35       90      no       125
```

```
## 3   dessert           8       45      150       no       195
## 4 appetizer           5       10       20      yes        30
```

5. Combine the data frames from problems 1 and 4 so that you have a new data set with 14 observations, and then print the resulting data frame.

```
allRecipies <- rbind(recipies, recipies2)
allRecipies
```

```
##         type ingredients prepTime cookTime hasMeat totalTime
## 1     entree           8       15       30     yes        45
## 2  appetizer           4       15       15     yes        30
## 3  appetizer           5        5       20     yes        25
## 4     entree          10       35       55      no        90
## 5     entree           6       20       25      no        45
## 6  appetizer           8       40       10     yes        50
## 7    dessert           7       25      120      no       145
## 8    dessert          15       30       25      no        55
## 9     entree          10       10       45     yes        55
## 10    entree           9       20       60     yes        80
## 11 appetizer           3       10        0      no        10
## 12    entree          15       35       90      no       125
## 13   dessert           8       45      150      no       195
## 14 appetizer           5       10       20     yes        30
```

6. Print all of the data for the recipes than take under an hour to make (start to finish).

```
allRecipies[allRecipies$totalTime < 60, ]
```

```
##         type ingredients prepTime cookTime hasMeat totalTime
## 1     entree           8       15       30     yes        45
## 2  appetizer           4       15       15     yes        30
## 3  appetizer           5        5       20     yes        25
## 5     entree           6       20       25      no        45
## 6  appetizer           8       40       10     yes        50
## 8    dessert          15       30       25      no        55
## 9     entree          10       10       45     yes        55
## 11 appetizer           3       10        0      no        10
## 14 appetizer           5       10       20     yes        30
```

Part 3

In this part you will practice working with a data frame that has already been created. The data set, named `mtcars`, is included with base `R`.

1. What type of data structure is the data set stored as? Use an appropriate function and then type your answer above your code (outside of the `R` code chunk).

The mtcars dataset is stored as a data frame with 32 observations and 11 variables.

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

2. How many cars are in the data set? How many variables are there? Use appropriate function(s) and then type your answers above your code (outside of the `R` code chunk).

The mtcars dataset has 32 cars and 11 variables. Columns tells us the number of variables and rows tells us the number of cars.

```
# columns
ncol(mtcars)
```

```
## [1] 11
```

```
#rows
nrow(mtcars)
```

```
## [1] 32
```

3. Print the first nine rows of the data set.

```
head(mtcars, n = 9)
```

```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
```

```
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
```

4. Extract the miles per gallon (mpg) variable from the data set.

```
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

5. Find the average and median mpg of the cars.

```
# mean
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```
# median
median(mtcars$mpg)
```

```
## [1] 19.2
```

6. Create a new data frame that consists of only the miles per gallon (mpg) and horsepower (hp) variables for all of the cars. Make sure not to remove the names of the cars.

```
mpgAndHp <- subset(mtcars, select = c(mpg, hp))
```

7. Print all of the data for the cars with at least 105 horsepower.

```
mpgAndHp[mpgAndHp$hp >= 105, ]
```

```
##                    mpg  hp
## Mazda RX4         21.0 110
## Mazda RX4 Wag     21.0 110
## Hornet 4 Drive    21.4 110
## Hornet Sportabout 18.7 175
## Valiant           18.1 105
## Duster 360        14.3 245
## Merc 280          19.2 123
## Merc 280C         17.8 123
## Merc 450SE        16.4 180
```

```
## Merc 450SL          17.3 180
## Merc 450SLC         15.2 180
## Cadillac Fleetwood  10.4 205
## Lincoln Continental 10.4 215
## Chrysler Imperial   14.7 230
## Dodge Challenger    15.5 150
## AMC Javelin         15.2 150
## Camaro Z28          13.3 245
## Pontiac Firebird    19.2 175
## Lotus Europa        30.4 113
## Ford Pantera L      15.8 264
## Ferrari Dino        19.7 175
## Maserati Bora       15.0 335
## Volvo 142E          21.4 109
```

8. Print all of the data for the cars with an mpg under 20 or over 25 from the data set.

```
mpgAndHp[mpgAndHp$mpg < 20 | mpgAndHp$mpg > 25, ]
```

```
##                      mpg  hp
## Hornet Sportabout   18.7 175
## Valiant             18.1 105
## Duster 360          14.3 245
## Merc 280            19.2 123
## Merc 280C           17.8 123
## Merc 450SE          16.4 180
## Merc 450SL          17.3 180
## Merc 450SLC         15.2 180
## Cadillac Fleetwood  10.4 205
## Lincoln Continental 10.4 215
## Chrysler Imperial   14.7 230
## Fiat 128            32.4  66
## Honda Civic         30.4  52
## Toyota Corolla      33.9  65
## Dodge Challenger    15.5 150
## AMC Javelin         15.2 150
## Camaro Z28          13.3 245
## Pontiac Firebird    19.2 175
## Fiat X1-9           27.3  66
## Porsche 914-2       26.0  91
## Lotus Europa        30.4 113
## Ford Pantera L      15.8 264
## Ferrari Dino        19.7 175
## Maserati Bora       15.0 335
```

9. Print all of the data for the cars with an mpg of at least 22 and a horsepower under 95.

```r
mpgAndHp[mpgAndHp$mpg >= 22 & mpgAndHp$hp < 95, ]
```

```
##                  mpg hp
## Datsun 710      22.8 93
## Merc 240D       24.4 62
## Fiat 128        32.4 66
## Honda Civic     30.4 52
## Toyota Corolla 33.9 65
## Fiat X1-9       27.3 66
## Porsche 914-2   26.0 91
```