Metadata

```
Course:  DS 5100
Module:  10 R Programming 1
Topic:   HW Computing Payoff for a Quota Structure
Author:  R.C. Alvarado (adapted)
Date:    7 July (revised)
```

Student Info

```
Name: Courtney Hodge
Ned ID: yss2zv
URL of this file on GitHub: https://github.com/courtneyhodge/DS5100-2024-06-R/blob/main/hw/M10/M10-HW-1-1.pdf
```

Instructions

In your **private course repo** use this notebook to write code that performs the tasks below.

Save your notebook in the `M10` directory.

Remember to add and commit these files to your repo.

Then push your commits to your repo on GitHib.

Be sure to fill out the **Student Info** block above.

To submit your homework, save your results as a PDF and upload it to GradeScope.

TOTAL POINTS: 12

Overview

A salesperson at a large tech firm is faced with a new payment structure.

This salesperson has a quarterly quota of $225,000.

The payment received follows a progressive schedule with four brackets as follows:

1. For the first 40% of quota, the salesperson receives 7% on quota reached
2. For the next 30% of quota, the salesperson receives 10% on quota reached
3. For the next 20% of quota, the salesperson receives 13% on quota reached
4. For the next 10% of quota, the salesperson receives 16% on quota reached

For example, if the salesperson is 50% to quota, reaching $112,500 of sales, then:

- `a` = the first 40% is paid out at 7%, thus payout = $225,000 * 40% * 7%
- `b` = the next 10% is paid out at 10%, thus payout = $225,000 * 10% * 10%

The total payout to the salesperson would be `a + b`.

Notice what does *not* happen: getting to the second bracket does NOT mean the payout is $225,000 * 50% * 10%.

In another example, a salesperson is at 20% quota. Their payout would be $225,000 * 20% * 7%.

This schedule represents earnings up to 100% of quota. We ignore sales above 100% here.

Given this, the salesperson would like to know how much she would earn if she reaches a given percentage of quarterly quota.

Note: The quota structure in this assignment is analogous to how the US tax system works: There are several **brackets** with rate r applied to dollars in bracket i.

Task 1

(4 points)

Create a dataframe that encodes the information presented in the question. That is, assume that each row of the dataframe stands for a bracket, and that the columns stand for the features described in the progressive schedule. Then, using the quarterly quota of $225,000, add columns to the dataframe that apply the encoded parameters to this value for each bracket. You should end up with columns for the earnings in dollars for each bracket, as well as the payout in dollars.

```
quota <- 225000

cut <- c(.4, .3, .2, .1) #percent (expressed as value between 0 and 1)

payout_pct <- c(.07, .1, .13, .16)

cut_sum <- cumsum(cut) #cumulitive sum of cut

amt <- cut * quota #cut * quota

payout <- amt * payout_pct

payout_sum <- cumsum(payout)

amt_sum <- cumsum(amt)

my_df <<- data.frame(cut, payout_pct, cut_sum, amt, payout, payout_sum, amt_sum)

my_df
```

cut <dbl>	payout_pct <dbl>	cut_sum <dbl>	amt <dbl>	payout <dbl>	payout_sum <dbl>	amt_sum <dbl>
0.4	0.07	0.4	90000	6300	6300	90000
0.3	0.10	0.7	67500	6750	13050	157500
0.2	0.13	0.9	45000	5850	18900	202500
0.1	0.16	1.0	22500	3600	22500	225000

4 rows

Task 2

(4 points)

Write a function that takes an argument for the fraction of quarterly quota reached by the salesperson, expressed as a decimal value between 0 and 1 (e.g. 0.8 means 80%), and which returns the dollar amount earned.

This function should use the previously defined dataframe as a global variable. Note that this function is greatly simplified if your first dataframe has cumulative sums for the dollar amount columns.

Do not use for loops in completing this task or the next. Instead, let your dataframe do the work. In your function, match the amount earned to the appropriate row in your first dataframe to get the answer.

```
my_function <- function(y){
  a <- 0
  b <- 0
  c <- 0
  d <- 0
  if(y < .5){
    first_tier <- TRUE
    #print("first")
    a <- y * 0.07 * quota
  }else if (y >= 0.5 && y < 0.8) {
    second_tier <- TRUE
    #print("second")
    a <- my_df$cut[1] * 0.07 * quota

    if (y == 0.5) {
        b <- my_df$cut[4] * 0.1 * quota
        #print("five")
    } else if (0.5 < y  && y < 0.7) {
        b <- my_df$cut[3] * 0.1 * quota
        #print("six")
    } else if (0.6 < y && y < 0.8) {
        b <- my_df$cut[2] * 0.1 * quota
        #print("seven")
    }
  }else if(.8  <= y && y < 1){
    third_tier <- TRUE
    #print("third")

    a <- my_df$cut[1] * 0.07 * quota
    b <- my_df$cut[2] * 0.1 * quota

    if(y == 0.8){
      c <- 0.1 * 0.13 * quota
    }else if(y == 0.9){
      c <- 0.2 * 0.13 * quota
    }

  }else if(y == 1){
    fourth_tier <- TRUE
    #print("fourth")
    a <- my_df$cut[1] * 0.07 * quota
    b <- my_df$cut[2] * 0.1 * quota
    c <- my_df$cut[3] * 0.13 * quota
    d <- my_df$cut[4] * 0.16 * quota
  }

  my_payout <- a + b + c + d

  return(my_payout)
}
```

Task 3

(2 points)

Call the function to get the dollar amount earned in increments of 10% in a range between 0% to 100% earned. Note that you can use `seq()` to generate these increments.

Be sure to put the results of your function at work into a second dataframe. That is, create a dataframe with columns for percent of quota earned and payout for that amount.

```
#based on the path from the Overview, I have to do math below
quota <- 225000
x <- seq(0, 1, length.out = 11) #percent of quota

my_function <- function(y){
  a <- 0
  b <- 0
  c <- 0
  d <- 0
  if(y < .5){
    first_tier <- TRUE
    #print("first")
    a <- y * 0.07 * quota
  }else if (y >= 0.5 && y < 0.8) {
    second_tier <- TRUE
    #print("second")
    a <- my_df$cut[1] * 0.07 * quota

    if (y == 0.5) {
        b <- my_df$cut[4] * 0.1 * quota
        #print("five")
    } else if (0.5 < y  && y < 0.7) {
        b <- my_df$cut[3] * 0.1 * quota
        #print("six")
    } else if (0.6 < y && y < 0.8) {
        b <- my_df$cut[2] * 0.1 * quota
        #print("seven")
    }
  }else if(.8  <= y && y < 1){
    third_tier <- TRUE
    #print("third")

    a <- my_df$cut[1] * 0.07 * quota
    b <- my_df$cut[2] * 0.1 * quota

    if(y == 0.8){
      c <- 0.1 * 0.13 * quota
    }else if(y == 0.9){
      c <- 0.2 * 0.13 * quota
    }

  }else if(y == 1){
    fourth_tier <- TRUE
    #print("fourth")
    a <- my_df$cut[1] * 0.07 * quota
    b <- my_df$cut[2] * 0.1 * quota
    c <- my_df$cut[3] * 0.13 * quota
    d <- my_df$cut[4] * 0.16 * quota
  }

  my_payout <- a + b + c + d
```

```
  return(my_payout)
}

payout_for_amounts <- c(my_function(x[1]), my_function(x[2]), my_function(x[3]), my_function(x
[4]), my_function(x[5]), my_function(x[6]), my_function(x[7]), my_function(x[8]), my_function(x
[9]), my_function(x[10]), my_function(x[11]))

my_df_2 <- data.frame(x * 225000, payout_for_amounts)
my_df_2
```

x...225000 <dbl>	payout_for_amounts <dbl>
0	0
22500	1575
45000	3150
67500	4725
90000	6300
112500	8550
135000	10800
157500	13050
180000	15975
202500	18900

1-10 of 11 rows Previous **1** 2 Next

Task 4

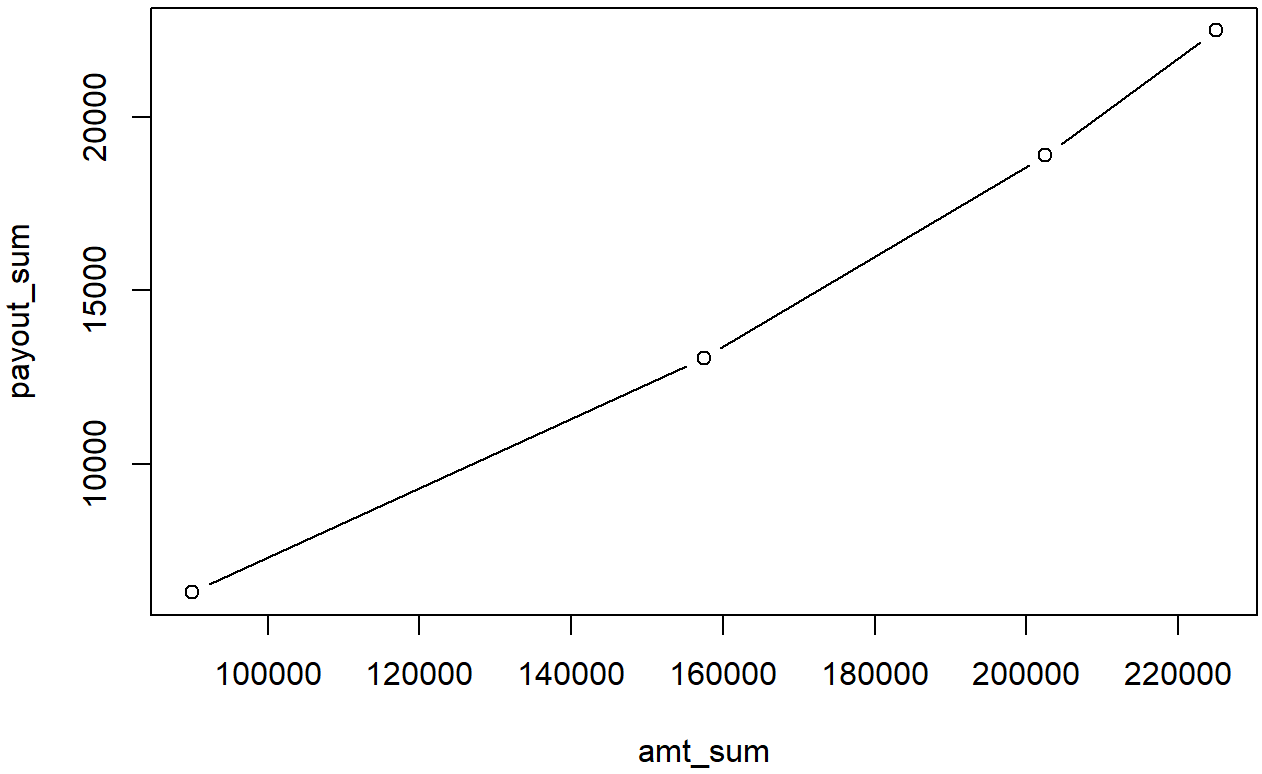
(1 point)

Using the first dataframe, plot the amounts earned (y-axis) versus quarterly quota reached (x-axis).

Display the graph using both points and lines.

Hint: for both axes, use the cumulative sums, which you should have defined above.

```
# CODE HERE
plot(my_df$amt_sum, my_df$payout_sum, type = 'b', xlab = 'amt_sum', ylab = 'payout_sum')
```



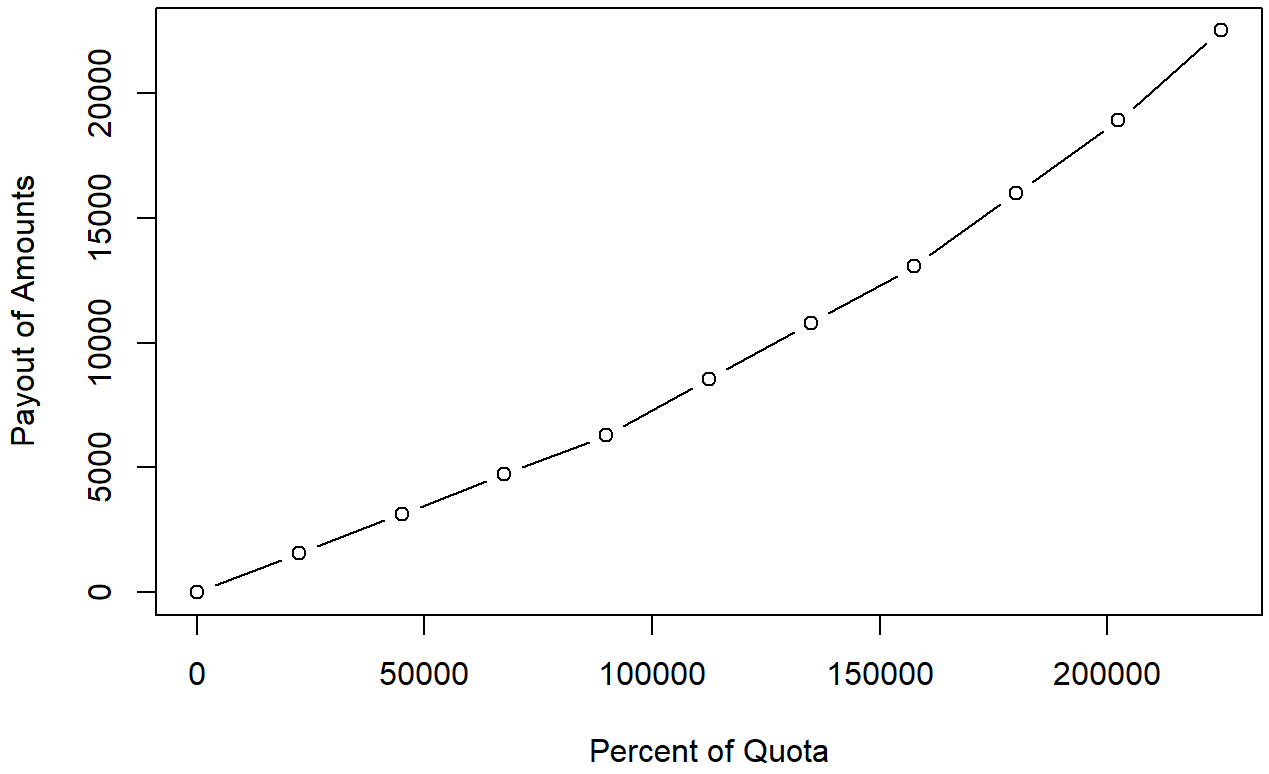
Task 5

(1 point)

Using the second dataframe, plot the dollar amount for each increment (x-axis) versus the payout in dollars (y-axis).

Again, display the graph using both points and lines.

```
# CODE HERE
plot(my_df_2$x, my_df_2$payout_for_amounts, type = 'b', xlab = 'Percent of Quota', ylab = 'Payou
t of Amounts')
```


Payout of Amounts
0
5000
10000
15000
20000
0
50000
100000
150000
200000
Percent of Quota