# Metadata

```
Course:  DS 5100
Module:  10 R Programming 1
Topic:   HW Computing Payoff for a Quota Structure
Author:  R.C. Alvarado (adapted)
Date:    06 October 2022 (revised)
```

# Student Info

```
Name: Ballard
Ned ID: bkq5nt
URL of this file on GitHub:https://github.com/ballard11/DS5100-2022-08-0/upload/main/lessons/M10_RBasics
```

# 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.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
##
## 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
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
quota <- 225000

#Create the dataframe w/ quota schedule

df <- data.frame(
      cut = c(.4,.3,.2,.1),
      payout_pct = c(.07, .1, .13, .16)
```

```
)

#Create additional paramters

df$Amount <- df$cut * quota

df$Payout <- df$Amount * df$payout_pct

df$Payout_Sum <- cumsum(df$Payout)

df$Amount_Sum <- cumsum(df$Amount)

df
```

```
##   cut payout_pct Amount Payout Payout_Sum Amount_Sum
## 1 0.4       0.07  90000   6300       6300      90000
## 2 0.3       0.10  67500   6750      13050     157500
## 3 0.2       0.13  45000   5850      18900     202500
## 4 0.1       0.16  22500   3600      22500     225000
```

## 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. In applying your function, use `apply()` and assign the result as a second column to your second dataframe.

```
get_payout <- function(qq_frac) {
  #Qyote Earned
  earned <-  quota * qq_frac
  payout = min(df[earned <= df$Amount_Sum, "Payout_Sum"])
  return(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.

```r
df2 <- data.frame(i = seq(.1 , 1, by= .1))

df2$Payout <- apply(df2, 1, get_payout)

df2
```

```
##      i Payout
## 1  0.1   6300
## 2  0.2   6300
## 3  0.3   6300
## 4  0.4   6300
## 5  0.5  13050
## 6  0.6  13050
## 7  0.7  18900
## 8  0.8  18900
## 9  0.9  18900
## 10 1.0  22500
```

## Task 4

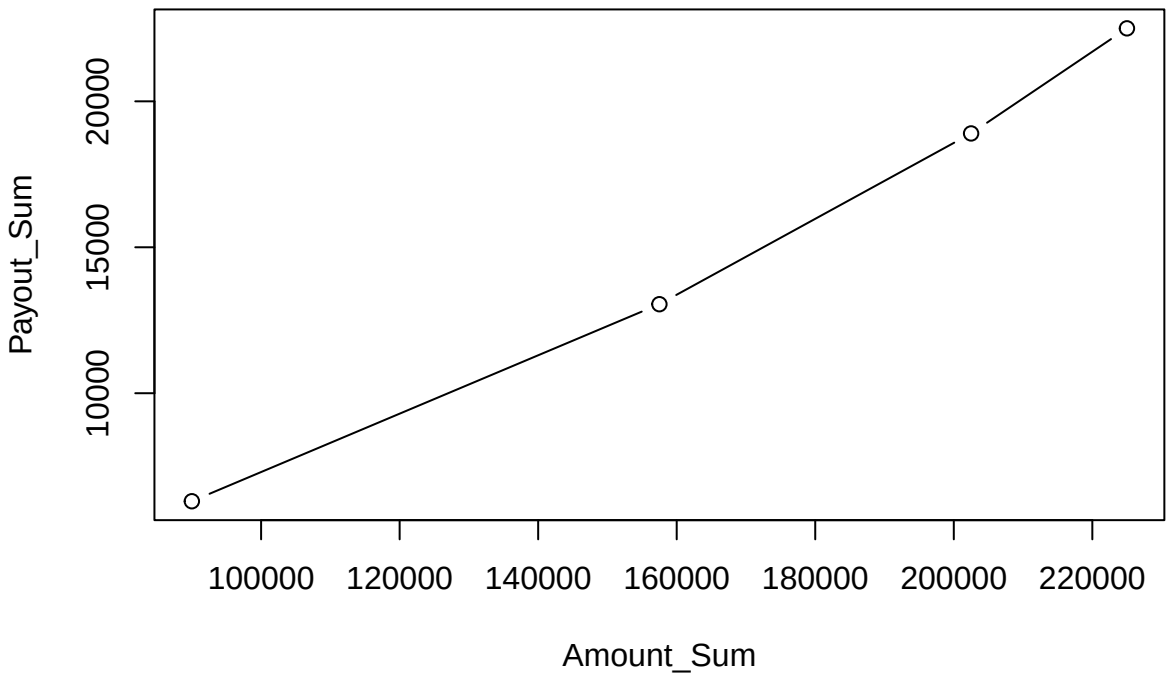
(1 point)

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.

```r
#Plot graphic
plot(df[c('Amount_Sum','Payout_Sum')], type = 'b')
```



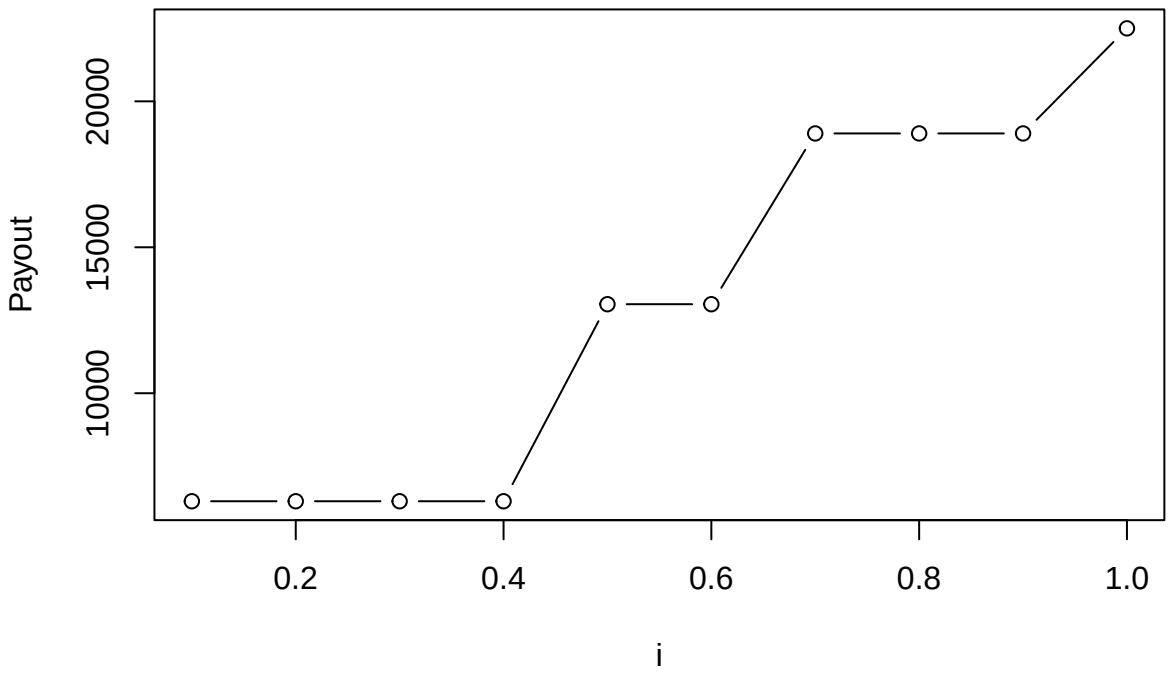

# Task 5

(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.

```
#plot DF 2
plot(df2[c('i','Payout')], type = 'b')
```


Payout
10000
15000
20000
0.2
0.4
0.6
0.8
1.0
i