hw10

Tom Lever

11/03/22

Metadata

Name: hw10

URL: https://github.com/tslever/DS5100-2022-08-tsl2b/blob/main/lessons/M10/hw10.Rmd

Course: DS 5100

Term: Fall 2022 Online

Module: M10: R Programming

Topic: Computing Payoff for a Quota Structure

Author: Tom Lever

Net ID: tsl2b

Created: 2 November 2022

Updated: 3 November 2022

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.

```
upper_percent_of_quota <- c(0.40, 0.30, 0.20, 0.10)
pay_rate <- c(0.07, 0.10, 0.13, 0.16)
data_frame <- data.frame(upper_percent_of_quota, pay_rate)
rownames(data_frame) <- c("bracket_1", "bracket_2", "bracket_3", "bracket_4")
quota <- 225000
upper_sales <- quota * data_frame[, "upper_percent_of_quota"]
data_frame$upper_sales <- upper_sales
upper_pay <- data_frame$upper_sales * data_frame[, "pay_rate"]
data_frame$upper_pay <- upper_pay
print(data_frame)
```

```
##           upper_percent_of_quota pay_rate upper_sales upper_pay
## bracket_1                    0.4     0.07       90000      6300
## bracket_2                    0.3     0.10       67500      6750
## bracket_3                    0.2     0.13       45000      5850
## bracket_4                    0.1     0.16       22500      3600
```

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.

```
library(dplyr)
library(TomLeversRPackage)
calculate_sales_and_pay <- function(proportion_of_quarterly_quota_reached_by_salesperson) {
    cumulative_upper_percent_of_quota <- cumsum(data_frame$upper_percent_of_quota)
    cumulative_upper_sales <- cumsum(data_frame$upper_sales)
    cumulative_upper_pay <- cumsum(data_frame$upper_pay)
    data_frame_with_accumulations <- data.frame(upper_percent_of_quota, cumulative_upper_percent_of_quo
    cumulative_upper_percent_of_quota_is_less_than_or_equal_to_proportion <- (cumulative_upper_percent_
    data_frame_with_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_proportion <- data_
    data_frame_with_minimum_cumulative_upper_percent_of_quota_greater_than_proportion <- data_frame_wit
```

```
    maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_proportion <- 0
    cumulative_upper_sales_for_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_proportio
    cumulative_upper_pay_for_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_proportion
    if (nrow(data_frame_with_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_proportion)
        maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_proportion <- max(data_frame_wi
        cumulative_upper_sales_for_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_propo
        cumulative_upper_pay_for_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_propor
    }
    difference_between_proportion_and_maximum_cumulative_upper_percent_of_quota_less_than_or_equal_to_pr
    if (near(proportion_of_quarterly_quota_reached_by_salesperson, maximum_cumulative_upper_percent_of_
        total_sales <- cumulative_upper_sales_for_maximum_cumulative_upper_percent_of_quota_less_than_or
        total_pay <- cumulative_upper_pay_for_maximum_cumulative_upper_percent_of_quota_less_than_or_equ
        return(c(total_sales, total_pay))
    } else {
        index_of_upper_pay <- get_column_index(data_frame_with_minimum_cumulative_upper_percent_of_quota
        index_of_upper_percent_of_quota <- get_column_index(data_frame_with_minimum_cumulative_upper_per
        index_of_upper_sales <- get_column_index(data_frame_with_minimum_cumulative_upper_percent_of_quo
        total_sales <- cumulative_upper_sales_for_maximum_cumulative_upper_percent_of_quota_less_than_or
        total_pay <- cumulative_upper_pay_for_maximum_cumulative_upper_percent_of_quota_less_than_or_equ
        return(c(total_sales, total_pay))
    }
}
```

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.

```
percent_of_quota <- seq(0.0, 1.0, by = 0.10)
sales_and_pay <- apply(as.array(percent_of_quota), 1, calculate_sales_and_pay)
sales <- sales_and_pay[1, ]
pay <- sales_and_pay[2, ]
percent_of_quota_sales_and_pay <- data.frame(percent_of_quota, sales, pay)
percent_of_quota_sales_and_pay
```

```
##    percent_of_quota  sales   pay
## 1               0.0      0     0
## 2               0.1  22500  1575
## 3               0.2  45000  3150
## 4               0.3  67500  4725
## 5               0.4  90000  6300
## 6               0.5 112500  8550
## 7               0.6 135000 10800
## 8               0.7 157500 13050
## 9               0.8 180000 15975
## 10              0.9 202500 18900
## 11              1.0 225000 22500
```

Task 4

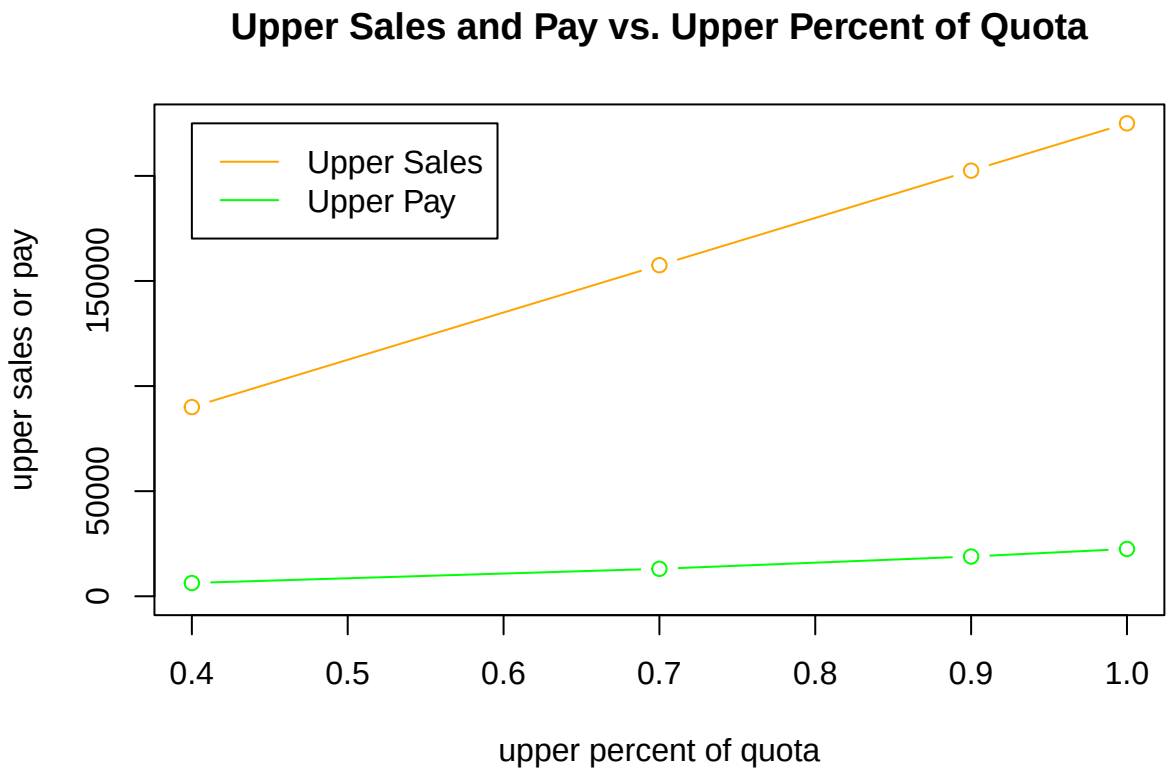
(1 point)

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.

```
cumulative_upper_percent_of_quota <- cumsum(data_frame$upper_percent_of_quota)
cumulative_upper_sales <- cumsum(data_frame$upper_sales)
cumulative_upper_pay <- cumsum(data_frame$upper_pay)
maximum_cumulative_upper_sales <- max(cumulative_upper_sales)
plot(
    cumulative_upper_percent_of_quota,
    cumulative_upper_sales,
    type = "b",
    col = "orange",
    ylim = c(0, maximum_cumulative_upper_sales),
    xlab = "upper percent of quota",
    ylab = "upper sales or pay"
)
lines(cumulative_upper_percent_of_quota, cumulative_upper_pay, type = "b", col = "green")
title("Upper Sales and Pay vs. Upper Percent of Quota")
legend(
    x = 0.4,
    y = maximum_cumulative_upper_sales,
    legend = c("Upper Sales", "Upper Pay"),
    lty = "solid",
    col = c("orange", "green")
)
```



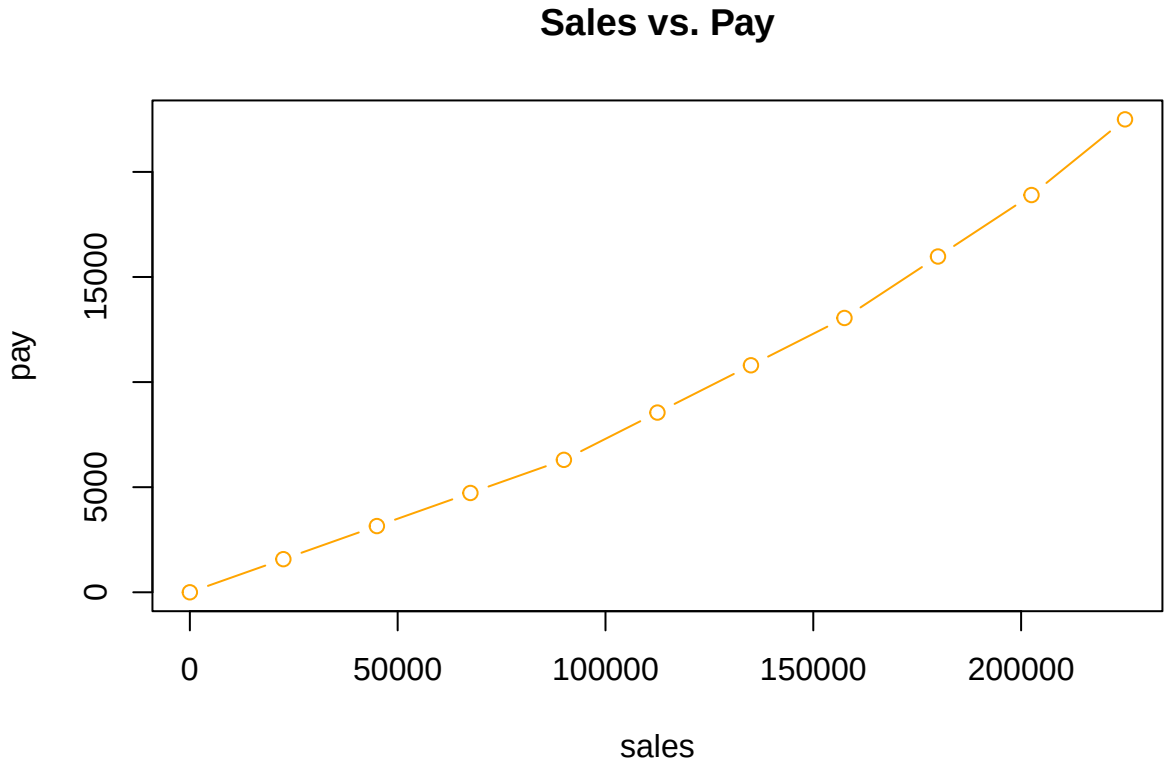
Task 5

(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.

```
maximum_pay <- max(pay)
plot(
    sales,
    pay,
    type = "b",
    col = "orange",
    xlab = "sales",
    ylab = "pay"
)
title("Sales vs. Pay")
```


Sales vs. Pay
pay
0
5000
15000
0
50000
100000
150000
200000
sales