MJDonaldson|MGMT6233|Assignment1

1) Vectors and Computations [Applied] Calculate the following by implementing it as a for() loop in R: X 99 i=10 (i 2 + i 3)

```
answer<-0
for(i in 10:99){
 answer<-answer+(i^2+i^3)
  }

answer
[1] 24828540
```

Is there another way of computing it that does not use a for() loop?

```
x_range<-(10:99)
f.x<-function(i)(i^2+i^3)
answer2<-sum(f.x(x_range))

answer2
[1]24828540
> answer
```

2)A test is graded with an average score of 35 and a standard deviation of 10. For comparison to other tests, it would be convenient to rescale the scores to a mean of 100 and standard deviation of 15.

• How can the scores be linearly transformed to have this new mean and standard deviation?

Because the scores are normally distributed, a linear transformation can be applied without changing the shape of the distribution. This is due to the fact that normal distributions belong to the sliding scale family of distributions.

Slope can be derived using the following formula:

$Var(Y) = m^2 * Var(X)$

$15^2=m^2*10^2$

$225= m^2 * 100$

$m=\sqrt{225/100}$

$m=1.5$

Mean can be calculated by simply adding a constant to every value generated by rnorm() in the later steps of this problem, in this case 47.5:
Y = (1.5)X + b
100=(1.5)*35+b
b=47.5

The linear transformation would be:
Y=1.5x+47.5

• How would you implement this transformation in R? Assume scores are drawn from a normal distribution. Use the rnorm() function to create random numbers drawn from a normal distribution – but make sure to adjust the mean and standard deviation accordingly. Use ?rnorm to find out how.

#Use the Monte Carlo Method to generate 100000 random values based on a mean of 35 and Standard
#Deviation of 10. *100000 was chosen in order to produce a result closer to the theoretical value of m(y)
x<-rnorm(n=100000, mean=35, sd=10)
y<-1.5*(x)+47.5
mean(y)
[1] 99.98441

3)Working with Character Vectors [Applied] Use the function paste() to create the following character vectors of length 30: 1. ("label 1", "label 2", ..., "label 30"). Note that there is a single space between label and the number following.

myCVLabel<-paste("label", 1:30)

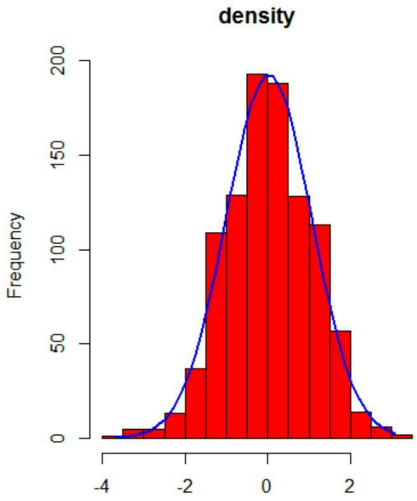
```
> myCVLabel
 [1] "label 1"  "label 2"  "label 3"
 [4] "label 4"  "label 5"  "label 6"
 [7] "label 7"  "label 8"  "label 9"
[10] "label 10" "label 11" "label 12"
[13] "label 13" "label 14" "label 15"
[16] "label 16" "label 17" "label 18"
[19] "label 19" "label 20" "label 21"
[22] "label 22" "label 23" "label 24"
[25] "label 25" "label 26" "label 27"
[28] "label 28" "label 29" "label 30"
> length(myCVLabel)
[1] 30
>
```

("fn1", "fn2", ..., "fn30"). In this case, there is no space between fn and the number following.

myCVFN<-paste('fn',1:30,sep="")

```
> myCVFN<-paste('fn',1:30,sep="")
> myCVFN
 [1] "fn1"  "fn2"  "fn3"  "fn4"  "fn5"  "fn6"
 [7] "fn7"  "fn8"  "fn9"  "fn10" "fn11" "fn12"
[13] "fn13" "fn14" "fn15" "fn16" "fn17" "fn18"
[19] "fn19" "fn20" "fn21" "fn22" "fn23" "fn24"
[25] "fn25" "fn26" "fn27" "fn28" "fn29" "fn30"
> length(myCVFN)
[1] 30
>
```

4) Random Numbers and Histograms [Applied] Let x = x1 + ... + x20, the sum of 20 independent Uniform(0,1) random variables. In R, create 1,000 simulations of x and plot their histogram. On the histogram, overlay a graph of the normal density function. Comment on any differences between the histogram and the curve.



Rcode:

```
x<-rnorm(1000)
h<-hist(x, breaks=10, col="red", xlab="x",
       main="density")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```

5)Use R to simulate a single fair coin toss. Your code should simply print “Heads” or “Tails” to the screen once every time you run it, each time with a probability p = 1/2 for a head or a tail.

```
coinFlip <-function(n) sample(0:1,n,rep=T)
printCoinFlip<-function(){
  coinValue<-coinFlip(1)
  if(coinValue==1){
    print("Heads")
  }
  else{
    print("Tails")
  }

}
>printCoinFlip()
[1] "Tails"
> printCoinFlip()
[1] "Tails"
> printCoinFlip()
[1] "Heads"
> printCoinFlip()
[1] "Tails"
```

Using a for loop, simulate 100 tosses of a fair coin and count how many heads you get in a variable called total which you print out after the loop.

```
Heads<-0
Tails<-0
for(x in 0:99){
  flip<-coinFlip(1)
  if(flip==1){
    Heads=Heads+1
  }
  else{
    Tails=Tails+1
  }
}
print(c(Heads,Tails))
[1] 57 43
print(c(Heads,Tails))
[1] 42 58
```

Optional/advanced Now repeat without resorting to a for loop. Can you do it in just two lines of code?

```
myHundredCoins<-coinFlip(100)
heads<-sum(myHundredCoins)
heads
[1] 52
```

6) **Execute the following line on the R console. What do you get? Can you explain, step by step, how R is interpreting the command? 1:10 > 5**

Result:

```
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
 [9]  TRUE  TRUE
```

Explanation:

This iterates over the range one through ten, returning “FALSE” for values less than 5 and “TRUE” for values greater than 5.

7)**A common business problem that we discussed in class is customer churn: existing customers that leave for the competition. You can download a dataset of churn data on Blackboard in the Datasets menu. Load the data set into R. You can use the below lines of code to help.**

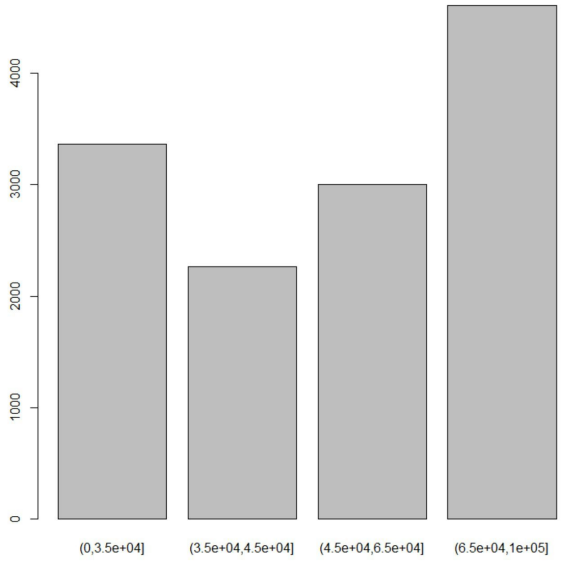
- **A)Explore the general structure of the data: How many rows and columns does the data have?**

  Answer: There are 20,000 rows and 12 columns.

  R Code:

  ```
  > nrow(churn)
  [1] 20000
  > ncol(churn)
  [1] 12
  ```

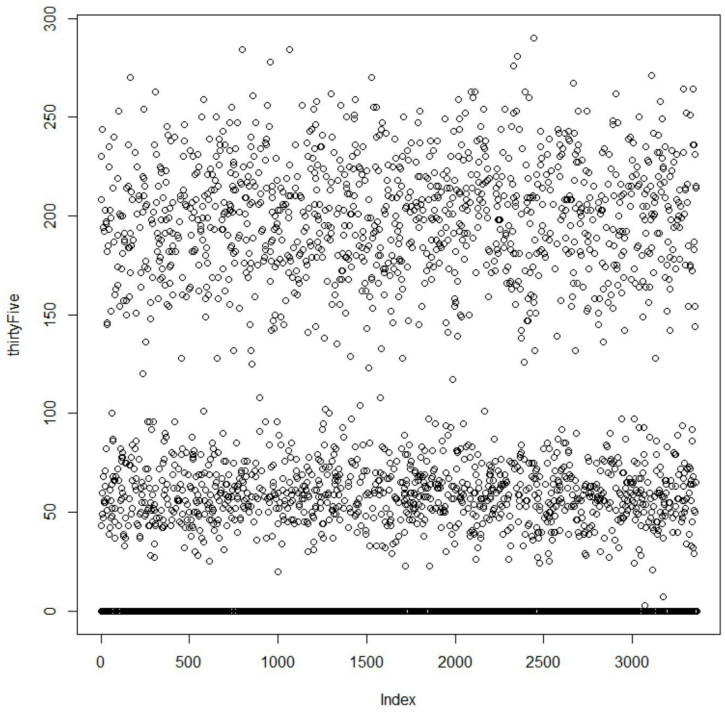
- **B) Create a new variable IncomeGroup that characterizes users with as <$35k; $35k-$45k; $45k-$65k; $65k-$100k; $100k. Use the function cut() to divide the data into intervals.**
  churnIntervals<-cut(churn[,2], breaks=(c(0,35000,45000,65000,100000)))
  plot(churnIntervals)



- **C) Plot the distribution of OVERAGE for each of the income groups.**
  Plot for incomes <35K

  RCode:
  thirtyFive<-subset(churn, INCOME < 35000)$OVERAGE
  plot(thirtyFive)

  Scatterplot:

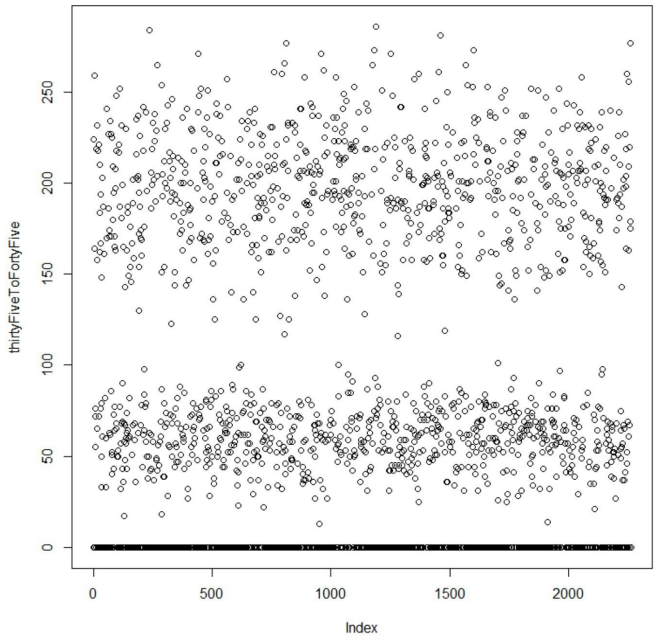


Plot for Incomes >=35K and <45k:

RCode:

```
thirtyFiveToFortyFive<-subset(churn, INCOME >=35000 & INCOME <45000)$OVERAGE
plot(thirtyFiveToFortyFive)
```

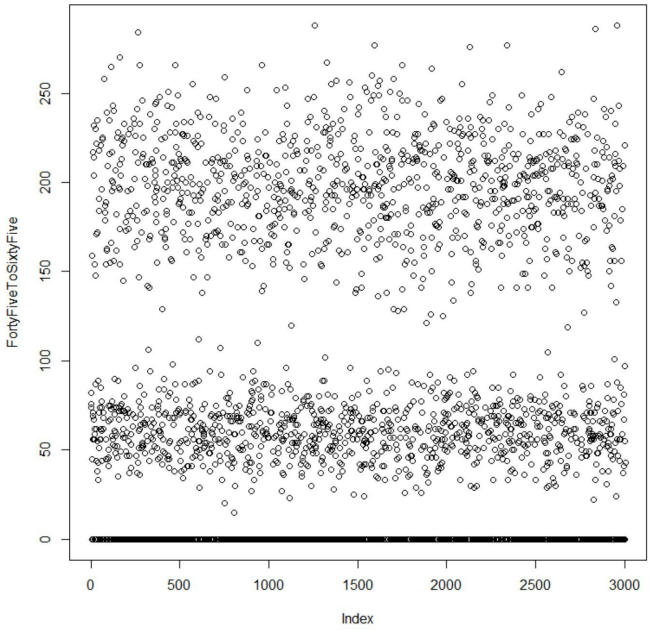
ScatterPlot >=35K and <45k:



Rcode >=45k<65k:

```
FortyFiveToSixtyFive<-subset(churn, INCOME >=45000 & INCOME <65000)$OVERAGE
plot(FortyFiveToSixtyFive)
```

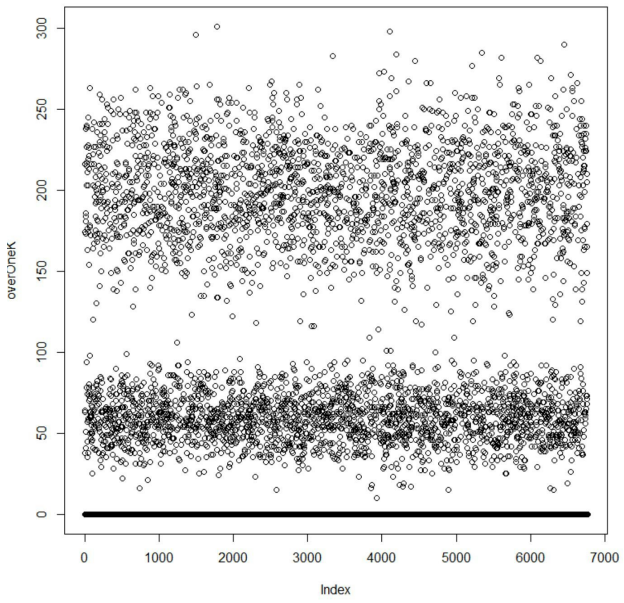
ScatterPlot >=45k<65k:



RCode >100K:

```
overOneK<-subset(churn, INCOME >100000)$OVERAGE
plot(overOneK)
```

ScatterPlot>100k:

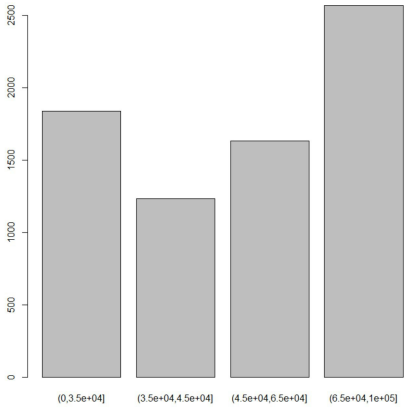


d) **The dataset has a variable that captures whether a certain customer canceled his/her cellphone contract (variable LEAV E). Explore the data with regard to LEAV E decisions and make visual and quantitavie comparisons across income groups and other variables to find out who is most likely to leave.**

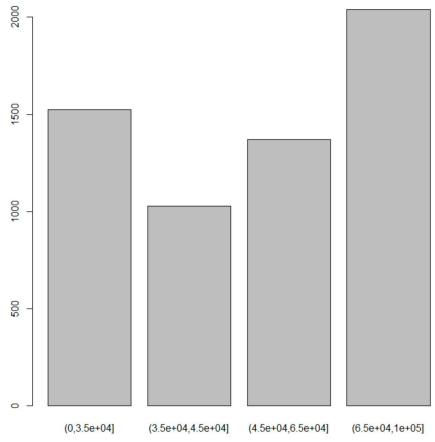
People who stay or leave compared to their income:
RCode stay or leave/income:

```
stay<-subset(churn, LEAVE=="STAY")
stayIntervals<-cut(stay[,2],breaks=(c(0,35000,45000,65000,100000)))
plot(stayIntervals)

leave<-subset(churn, LEAVE=="LEAVE")
leaveIntervals<-cut(leave[,2],breaks=(c(0,35000,45000,65000,100000)))
plot(leaveIntervals)
```



Stay by Income Level



Leave by Income Level

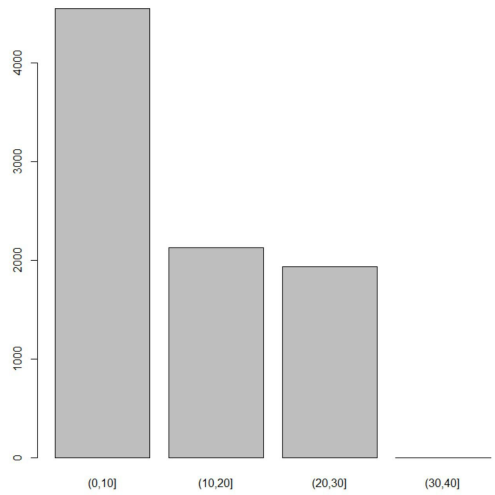
The comparison of these graphs shows a similar distribution for customers who stay and customers who leave. This suggests that income is not a strong factor in whether or not customers choose to leave.

People who leave compared to the Number of Calls they Make:

Rcode leave/number of calls:

```
leave<-subset(churn, LEAVE=="LEAVE")
leaveCallPerMonth<-cut(leave[,7],breaks=(c(0,10,20,30,40)))
plot(leaveCallPerMonth)
```

Graph leave/number of calls:

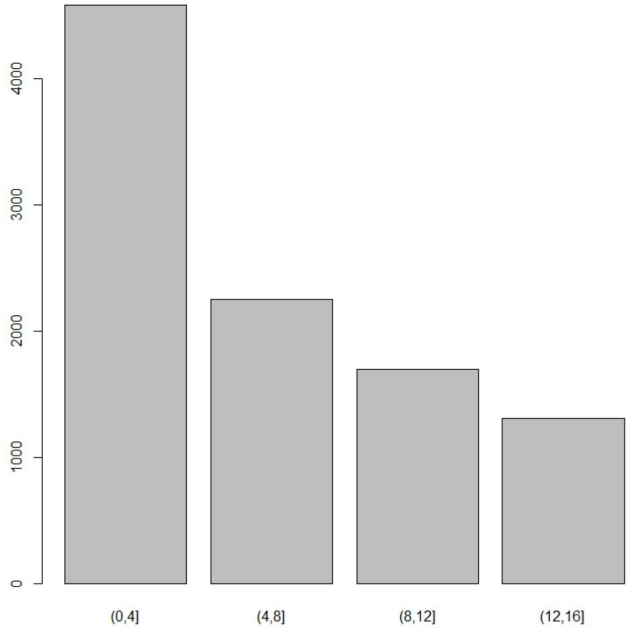


This graph shows a marked drop off in the customers who leave among those who make 30-40 calls per month. Those who make 0-10 calls per month are the most likely to leave, 10-20 calls per months the second most likely, 20-30 the third most likely. This direct relationship suggest that the number of calls a customer makes per month could potentially serve as a good marker of whether or not they will leave.

People who leave compared to the length of their phone calls:
Rcode leave/length of call:

```
leave<-subset(churn, LEAVE=="LEAVE")
leaveDuration<-cut(leave[,8],breaks=(c(0,4,8,12,16)))
plot(leaveDuration)
```

Graph leave/length of call:



Again, in this graph, a direct relationship can be observed between the customers who leave and the length of their phone calls. Those who make the shortest phone calls (0-4 minutes) are much more likely to leave than those who talk longer. This too could serve as a potential marker for identifying customers who might choose to leave.

D)Create metrics/measures/statistics that summarize the data. Examples of potentials metrics include min, max, mean, variance (standard deviation) and these can be calculated across various user segments. Be selective. Think about what might be most important to track.

Mean income of people who leave compared to the mean income of people who stay:

```
leave<-subset(churn, LEAVE=="LEAVE")
stay<-subset(churn, LEAVE=="STAY")
mean(leave[,2])
[1] 84355.88
mean(stay[,2])
[1] 76325.86
```

The mean income of people who leave is higher than the income of those who stay by ~$8,000/year. This is a notable difference that suggest that income may have a greater role to play than previously predicted in the bar graphs above, however it is most likely a minor correlation.

The number of calls people who leave make compared to the number of calls people who stay make:

```
leave<-subset(churn, LEAVE=="LEAVE")
stay<-subset(churn, LEAVE=="STAY")
number of calls
mean(leave[,7])
[1] 9.832217
mean(stay[,7])
[1] 6.222605
```

Keeping in line with the bar graphs above, one can see that the customers who make an average of 9.8 calls per month are much more likely to stay than those who make an average only 6.2 calls per month.