Problem Set 3: Graphics

Prateek Kumar

1. Dotchart Function

```
# I have set the figure dimension as a global option
```

```
---
title: "Problem Set 3"
author: "Prateek Kumar"
date: "23 September 2018"
output:
  word_document:
    fig_width: 4
    fig_height: 3
---
```

Figure 1: I am setting 4X3 because I have interchanged my plot's x and y axis for Q1

```
set.seed(100)
data2 <- data.frame(q1=sample(letters[1:10],100,replace=T), #The given dataset to test
                    q2=sample(letters[1:10],100,replace=T),
                    q3=sample(letters[1:10],100,replace=T),
                    q4=sample(letters[1:10],100,replace=T),
                    q5=sample(letters[1:10],100,replace=T))

datatable2<-apply(data2,2,table)

# A new dotchart function mydotchart()
mydotchart <- function(data,labels=NULL, colors = 1:5, main = "Displaying the Dot chart", xla
b = "Letters", ylab = "Number of Letters", xlim = c(0, 13), ylim = c(0, 20), lty = 1, normali
ze=F,col, pch = 15, cex = 1,subsets)
  {
  # Checking if we want to normalize the data
  if (normalize)
    {
    data <- (data - min(data))/(max(data)-min(data)) # Data normalized
    # Plotting the dotchart
    matplot(1:nrow(data), data , pch = pch, xlim = xlim, ylim = c(0,1), col = colors, xaxt="n
", main = main, xlab = xlab, ylab = ylab, lty = lty, cex = cex, type='b')

    axis(1,1:nrow(data),letters[1:10],las=1) # Setting the x-axis as the parameters

    # Drawing the segments
    segments( 1:10, 0, 1:10, 10, lty = 3)
    segments( x0=1, y0=c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1), x1=10, y1=c(0,0.1,0.2,0.3,
0.4,0.5,0.6,0.7,0.8,0.9,1), lty = 3)

    # Drawing the legend
    legend(11,1, legend = colnames(data), col = 1:5, fill = 1:5, cex = 0.6, title="Line types
", bg='grey')
  }
  else
```

```
  {
    # If we do not want to normalize, plotting the dotchart as per the data
    matplot(1:nrow(data), data , pch = pch, xlim = xlim, ylim = ylim, col = colors, xaxt="n",
main = main, xlab = xlab, ylab = ylab, lty = lty, cex = cex, type='b')

    axis(1,1:nrow(data),letters[1:10],las=1) # Setting the x-axis as the parameters

    # Drawing the segments
    segments( 1:10, 0, 1:10, 20, lty = 3)
    segments( x0=1, y0=0:20, x1=10, y1=0:20, lty = 3)

    # Drawing the legend
    legend(11,20, legend = colnames(data), col = 1:5, fill = 1:5, cex = 0.6, title="Line type
s", bg='grey')
  }

}
```

- The above is the dotchart function based on the matplot version we did in class. The function name is mydotchart(). The function has arguments like data, colors, main etc. which helps to customize the dotchart as per the user. Here I have set the default parameters for some of the values incase the user does not pass those arguments the dotchart will be plotted based upon those values.
- Here we have tested the dotchart plotting based upon the data frame “datatable2”.
- When we see the set of arguments in the function, there is an argument “normalize”. We are setting the value to False as default and when the user wants to normalize the data he/she can just call the mydotchart() function passing the normalize argument as True. Now what normalize does? Normalize adjusts the value on different scale to a common scale, here it is setting the values between 0 to 1.
- Now the meaning of the arguments passed:
    1. data: The dataset on which we want to plot the dotchart
    2. labels: Inorder to create our own value labels
    3. colors: Setting the colors in the plot
    4. main: Display the plot title
    5. xlab: Names of the x-axis
    6. ylab: Names of the y-axis
    7. xlim: Setting the limit of x-axis
    8. ylim: Setting the limit of y-axis
    9. lty: Setting the line type in the plot
    10. normalize: Contains the boolean value T/F if we want to normalize the data or not
    11. col: Setting the plotting color
    12. pch: Setting the point shape in the plot
    13. cex: Scaling the plot
    14. subsets: Inorder to subset the data

```
mydotchart(datatable2)
```

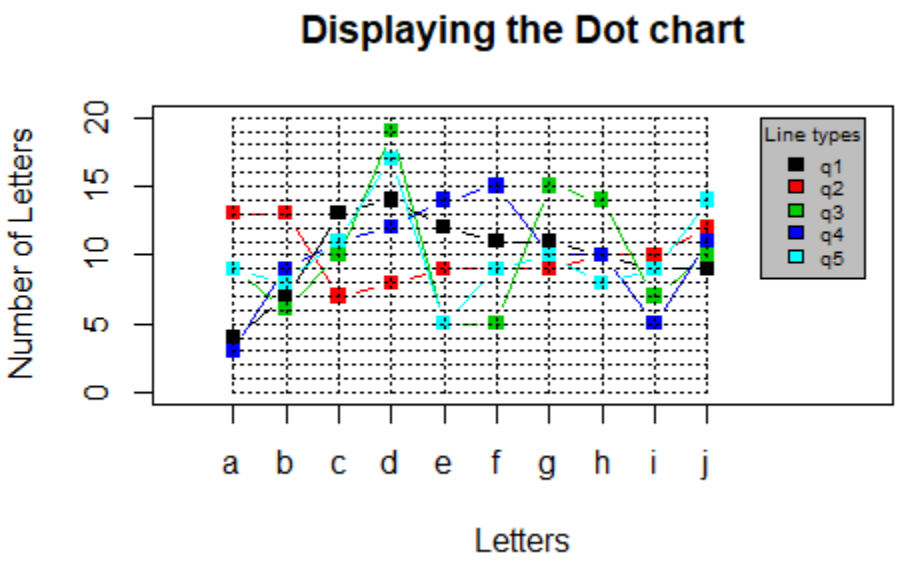

Figure 2: Here we just plot the dotchart on the dataset 'datatable2'

```
mydotchart(datatable2[,1:2])
```

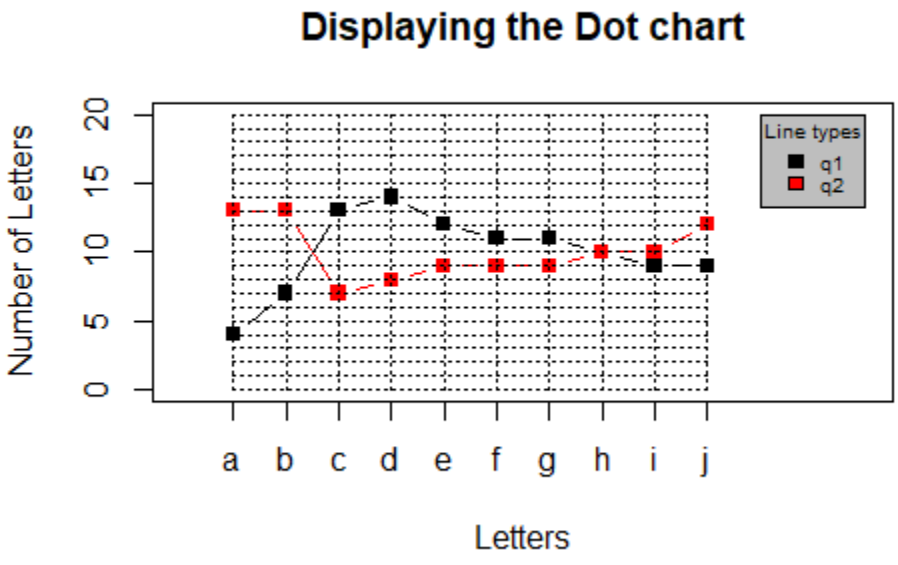

Figure 3: Here we plot the dotchart on the first 2 columns of the dataset 'datatable2'

```
mydotchart(datatable2[,1])
```

```
Error in 1:nrow(data) : argument of length 0
```

Figure 4: Here we get the error because data is interpreted as a vector

```
mydotchart(as.matrix(datatable2[,1]))
```

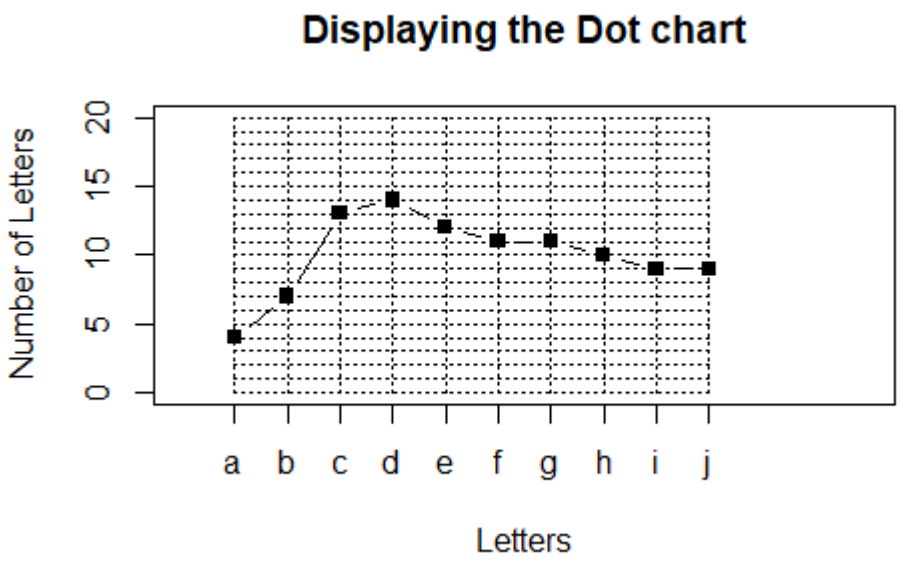

Figure 5: Here we get the plot without the legend

```
Error in legend(11, 20, legend = colnames(data), col = 1:5, fill = 1:5,  :
  'legend' is of length 0
```

Figure 6: The error because the length of legend is zero

```
mydotchart(datatable2[,1:3])
```

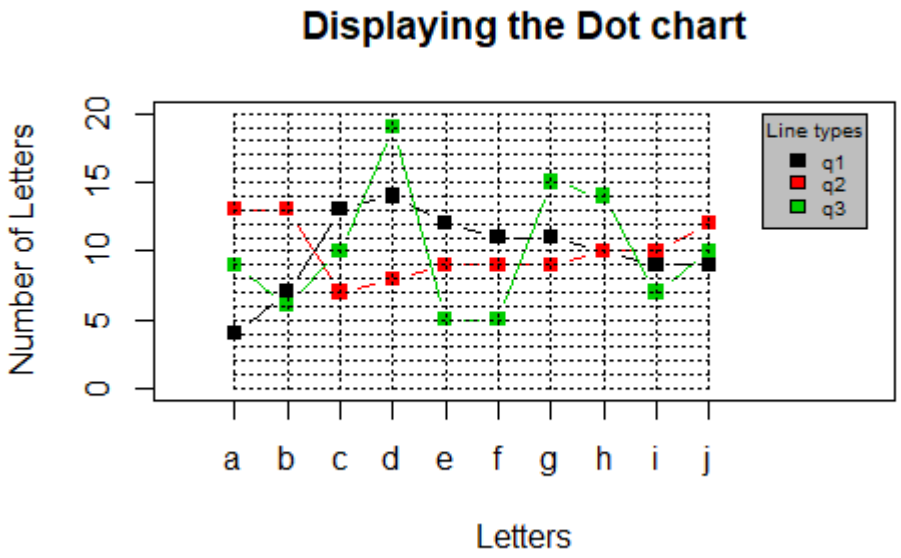

Figure 7: Here we plot the dotchart on the first 3 columns of the dataset 'datatable2'

```
mydotchart(datatable2,col=1:5)
```

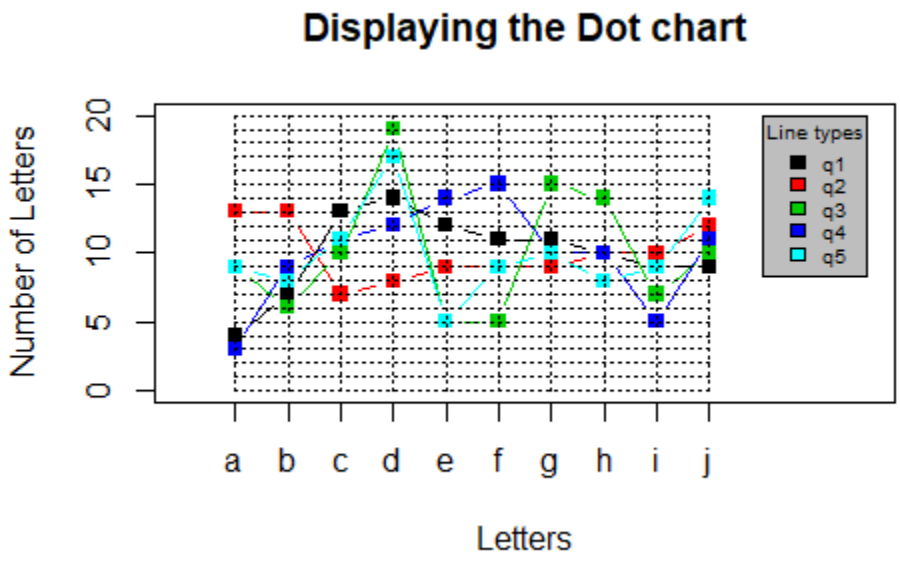

Figure 8: Here we plot the dotchart on the first 5 columns of the dataset 'datatable2'

```
mydotchart(datatable2,col=1:5,pch=16)
```

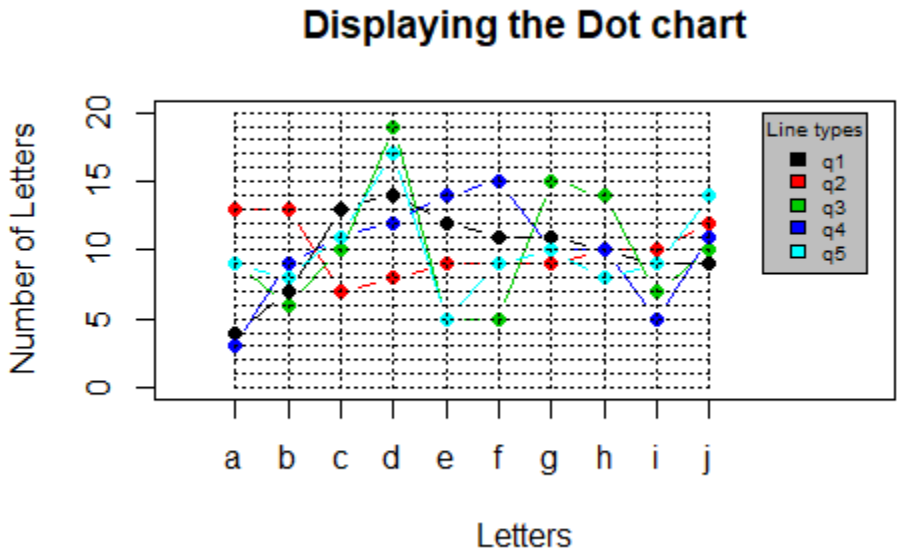

Figure 9: Here we just plot the dotchart on the dataset 'datatable2' with color and pch value

```
mydotchart(datatable2,col=1:5,pch=16,cex=2.5,main="Everything",xlab="Value", ylab="Category")
```

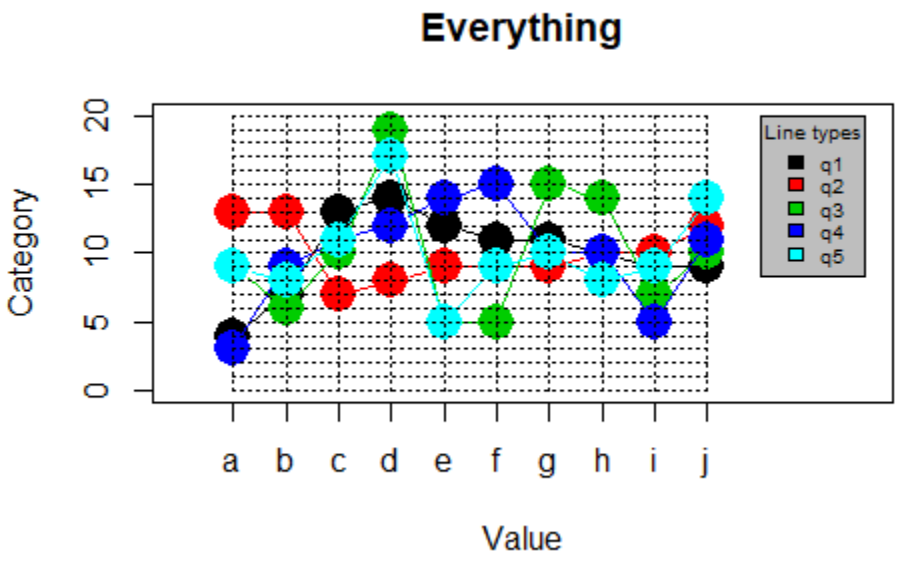

Figure 10: This is same as fig 9 with cex, main, xlab and ylab values

```
mydotchart(datatable2,col=1:5,pch=16,cex=2.5,main="Everything normalized",xlab="Value", ylab=
"Category", normalize=T)
```

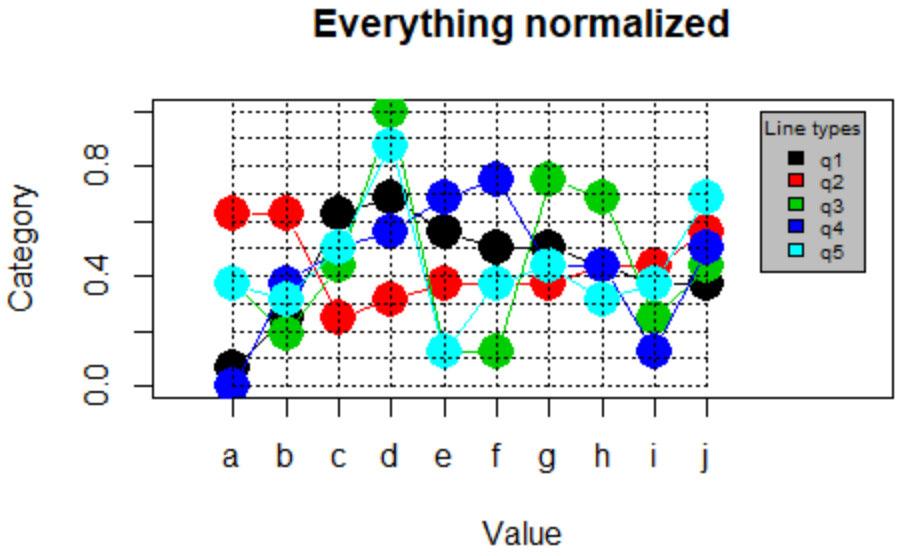

Figure 11: This is same as fig 10 but the data is normalized

2. Correlating word frequency with SCRABBLE scores

```
# Frequency of each letter
lf <- c(8.167,1.492,2.782,4.253,12.702,2.228,2.015,6.094,
        6.966,0.153,0.772,4.025,2.406,6.749,7.507,1.929,
        0.095,5.987,6.327,9.056,2.758,0.978,2.36,0.15,1.974,0.074)/100

# Points earned in Scrabble
pts <- c(1,3,3,2,1,4,2,4,1,8,5,1,3,1,1,3,10,1,1,1,1,4,4,8,4,10)

# Number of Scrabble tiles
tiles <- c(9,2,2,4,12,2,3,2,9,1,1,4,2,6,8,2,1,6,4,6,4,2,2,1,2,1)

#Creating the data frame of the four values
lf.table <- data.frame(LETTERS, freq=lf, points=pts, ntiles=tiles)
```

This function computes the sum of the inverse letter frequency of the letters, the total scrabble points, the mean numbers of tiles of the letters in the word, and the length of the word

```
scoreme <- function(word)
{

  lets <- strsplit(splus2R::upperCase(word),"")[[1]]
  data <- matrix(0,ncol=4,nrow=length(lets))

  for(i in 1:length(lets))
  {
    index <- which(lets[i]==LETTERS)
    data[i,1] <- lf.table$freq[index]
    data[i,2] <- lf.table$points[index]
    data[i,3] <- lf.table$ntiles[index]

  }
  list(suminvfreq= sum(1/data[,1]),
       points=sum(data[,2]),
       meantiles=mean(data[,3]),
       length=length(lets))
}
```

The following lists a set of words, along with their rank frequency (lower meaning more frequent), and their total frequency (number of occurrences in a large corpus)

```
test <- read.table(text='rank word frequency
                   1081    CUP             1441306
                   2310    FOUND           573305
                   5285    BUTTERFLY       171410
                   7371    brew            94904
                   11821   CUMBERSOME      39698
                   17331   useable         17790
                   18526   WHITTLE         15315
                   25416   SPINY           7207
                   27381   uppercase       5959
                   37281   halfnaked       2459
                   47381   bellhop         1106
                   57351   tetherball      425
```

```
                    7309    attic           2711
                    17311   tearful         542
                    27303   tailgate        198
                    37310   hydraulically   78
                    47309   unsparing       35
                    57309   embryogenesis   22 ', header=T, stringsAsFactors=FALSE)[,c(2,1,3)]
```

We add four columns into the data frame for the four statistics value: sum of the inverse letter frequency of the letters, the total scrabble points, the mean numbers of tiles of the letters in the word, and the length of the word

```
test$meantiles <- NA
test$suminvfreq <- NA
test$points <- NA
test$length <- NA
```

We now populate the four statistics value into the table

```
for(i in 1:nrow(test))
  {
  temp<-scoreme(test[i,1])
  test[i,5] <- temp[1]
  test[i,6] <- temp[2]
  test[i,4] <- temp[3]
  test[i,7] <- temp[4]
}
```

We now plot the values

```
par(mfrow=c(1,2)) # we are showing plots with one statistic value for each rank and frequency

plot(test$rank,test$meantiles,xlab = 'Rank', ylab = 'Meantiles',pch=16, main = paste('Rank vs
Meantiles\nCor =',round(cor(test$rank,test$meantiles),3)))


plot(test$frequency,test$meantiles, xlab = 'Frequency', ylab = 'Meantiles', xlim = c(0,10000)
, pch=16, main = paste('Frequency vs Meantiles\nCor =',round(cor(test$frequency,test$meantile
s),3)))
```

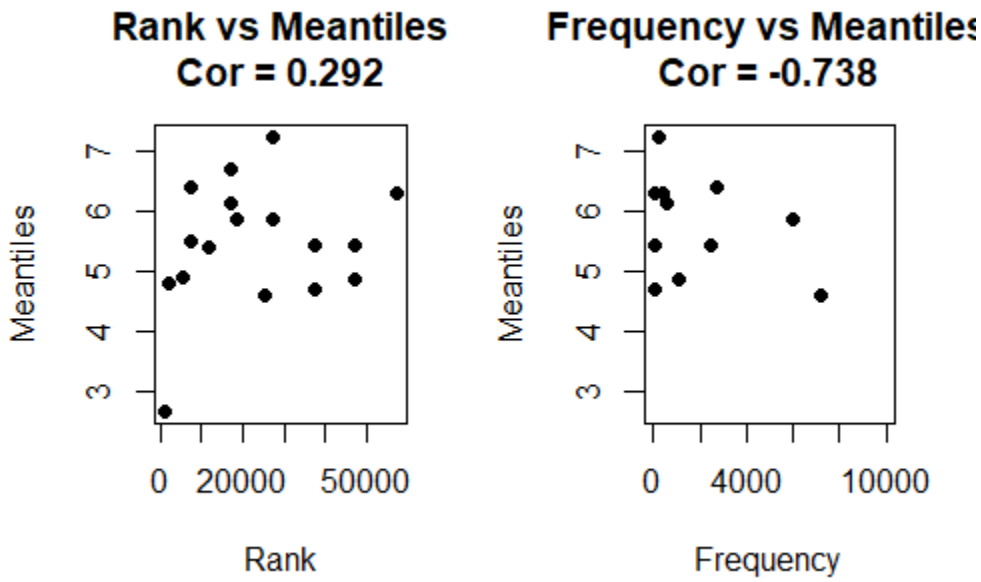

Figure 12: Rank and frequency vs the mean numbers of tiles of the letters in the word

```
plot(test$rank,test$suminvfreq, xlab = 'Rank', ylab = 'Suminvfreq',pch=16, main = paste('Rank
vs Suminvfreq\nCor =',round(cor(test$rank,test$suminvfreq),3)))

plot(test$frequency,test$suminvfreq, xlab = 'Frequency', ylab = 'Suminvfreq',  xlim = c(0,100
00), pch=16, main = paste('Frequency vs Suminvfreq\nCor =',round(cor(test$frequency,test$sumi
nvfreq),3)))
```

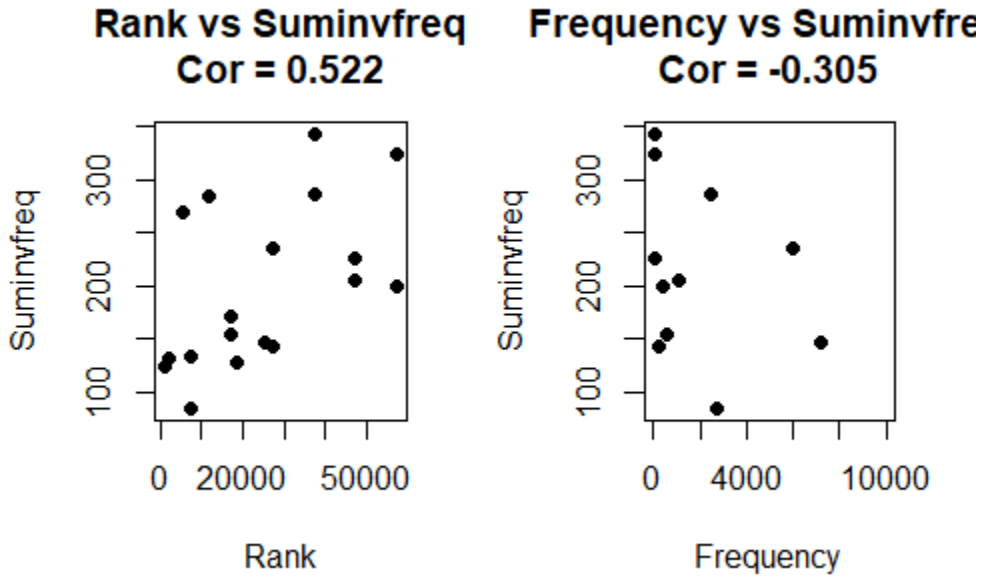

Figure 13: Rank and frequency vs the sum of the inverse letter frequency of the letters

```
plot(test$rank,test$points, xlab = 'Rank', ylab = 'Points',pch=16, main = paste('Rank vs Poin
ts\nCor =',round(cor(test$rank,test$points),3)))

plot(test$frequency,test$points, xlab = 'Frequency', ylab = 'Points',  xlim = c(0,10000),pch=
16, main = paste('Frequency vs Points\nCor =',round(cor(test$frequency,test$points),3)))
```

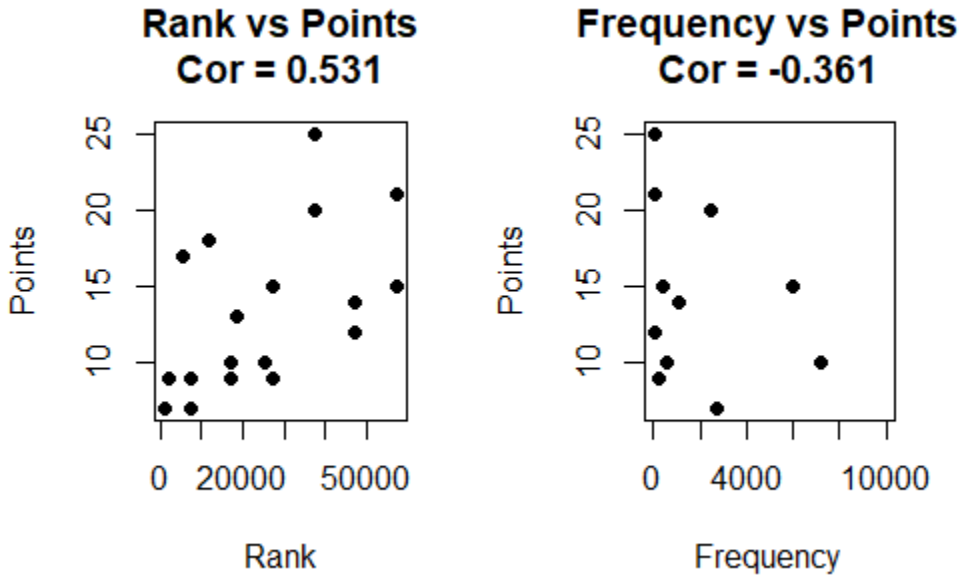

Figure 14: Rank and frequency vs the total scrabble points

```
plot(test$rank,test$length, xlab = 'Rank', ylab = 'Length',pch=16, main = paste('Rank vs Leng
th\nCor =',round(cor(test$rank,test$length),3)))

plot(test$frequency,test$length, xlab = 'Frequency', ylab = 'Length',  xlim = c(0,10000),pch=
16, main = paste('Frequency vs Length\nCor =',round(cor(test$frequency,test$length),3)))
```

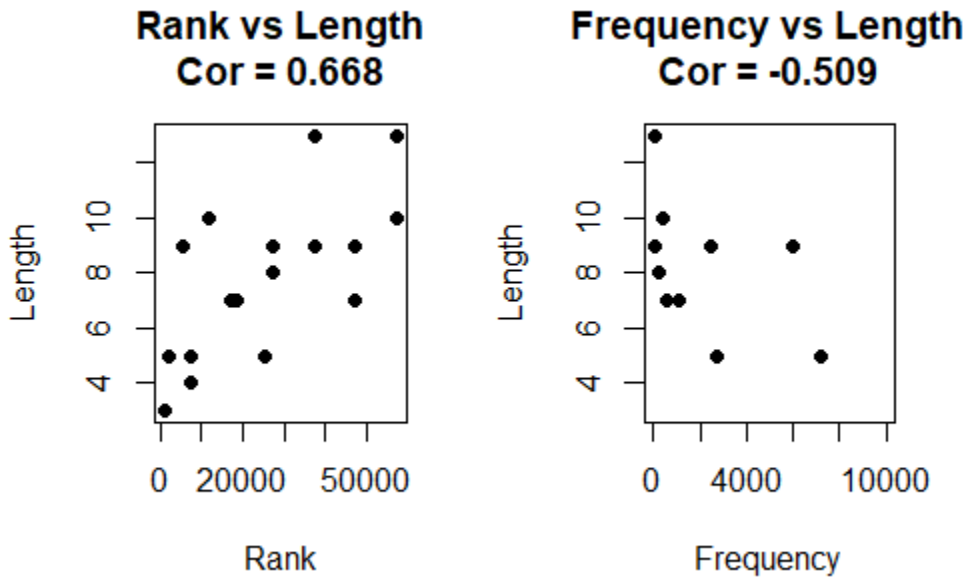

Figure 15: Rank and frequency vs length of the word

- Negative correlation is a relationship between two variables in which one variable increases as the other decreases, and vice versa and positive correlation exists when one variable decreases as the other variable decreases, or one variable increases while the other increases.
- In the above plots we see that all the four statistics values are positively correlated with rank and are in negative correlation with frequency.

Problem Set 3.Rmd* × test ×

Filter

	word	rank	frequency	meantiles	suminvfreq	points	length
1	CUP	1081	1441306	2.666667	124.04385	7	3
2	FOUND	2310	573305	4.800000	132.79219	9	5
3	BUTTERFLY	5285	171410	4.888889	270.32931	17	9
13	attic	7309	2711	6.400000	84.63001	7	5
4	brew	7371	94904	5.500000	133.97264	9	4
5	CUMBERSOME	11821	39698	5.400000	283.92776	18	10
14	tearful	17311	542	6.142857	153.84862	10	7

Figure 16: Looking at the table we can say that rank is positively correlated with all the four statistics value i.e. when rank increases the values tend to increase.

Problem Set 3.Rmd* × test ×

Filter

	word	rank	frequency	meantiles	suminvfreq	points	length
18	embryogenesis	57309	22	6.307692	323.29833	21	13
17	unsparing	47309	35	5.444444	226.46828	12	9
16	hydraulically	37310	78	4.692308	343.52430	25	13
15	tailgate	27303	198	7.250000	143.27433	9	8
12	tetherball	57351	425	6.300000	199.90076	15	10
14	tearful	17311	542	6.142857	153.84862	10	7
11	bellhop	47381	1106	4.857143	206.15716	14	7
10	halfnaked	37281	2459	5.444444	286.36268	20	9

Figure 17: When we check the values with frequency we see that they are negatively correlated i.e. when the frequency increase the values tend to decrease.

- Rank frequency gives an idea that how much is the word frequent in general, the lower the value the more frequent the word, and total frequency is the number of occurrences of the word in a large corpus. By looking at rank frequency we can identify that how frequent is the word and the total frequency is relevant to a particular collection of written or spoken material, so it specifies the occurrence of a word in that corpus.
- This does not signify that if rank frequency is lower than the raw frequency will be high. As said earlier that raw frequency depends on a particular corpus, so there might be a possibility that a particular word has less rank frequency and also has less raw frequency. E.g. In our table test the word ‘attic’ signifies this. So we can say that a higher correlation is not always meaningful.
- Note that correlation does not suggest causality. E.g. Let’s consider the correlation between sales of ice-cream in summer and increment in petroleum costs. Obviously, both are not related but rather we will

get a positive correlation here because both the values are increasing. So it is additionally critical to note that the correlation coefficient just estimates linear relationships. A meaningful *nonlinear* relationship may exist regardless of whether the correlation coefficient is 0.