kakumanuLab1Part1

February 13, 2019

Name : Asish Kakumanu UB Person No. : 50288695 UBIT Name : asishkak

---

Teammate
Name : Swapnika P UB Person No. : 50289464 UBIT Name : swapnika

0.1 Basic R Commands

```
In [2]: # Creating Variables foo and bar with values 2 and 4 respectively.

        foo <- 2
        bar <- 4
        foo + bar
```

```
6
```

```
In [3]: # Assigning the resultant value of foo + bar to variable (result).

        result <- foo + bar
        result
```

```
6
```

```
In [18]: ## Vectors (Lists)
         ## Combine all numbers into a vector and assign them to a variable called list.

         list <- c(2,4,6,8)

         # Returns item in list with index 2.
         list[2]
```

```
4
```

```
In [19]: ## Returns item in list with index 1.
         list[1]
```

```
2
```

```
In [20]: list[0]
In [21]: list[5]
```

<NA>

```
In [22]: # Adding a value to the list at index 5.

         list[5] <- 10
```

```
In [24]: # Return list.

         list
```

1. 2 2. 4 3. 6 4. 8 5. 10

0.2 Arithmetic Operations

```
In [25]: 10 / 2
```

5

```
In [26]: 2 == 0
```

FALSE

```
In [27]: 10 ^ 2
```

100

```
In [28]: 4 * 5
```

20

```
In [29]: 1 + 6
```

7

```
In [30]: (2+2) == 4
```

TRUE

```
In [31]: T == TRUE
```

TRUE

```
In [32]: F && T
```

FALSE

```
In [33]: F || TRUE
```

TRUE

```
In [34]: vect = c(2,4,6,8)
```

```
In [35]: vect * 2
```

1. 4 2. 8 3. 12 4. 16

```
In [36]: names(vect) = c("1st","2nd","3rd","4th")
```

```
In [37]: vect
```

1st 2 **2nd** 4 **3rd** 6 **4th** 8

```
In [39]: vect["2nd"] <- 20
```

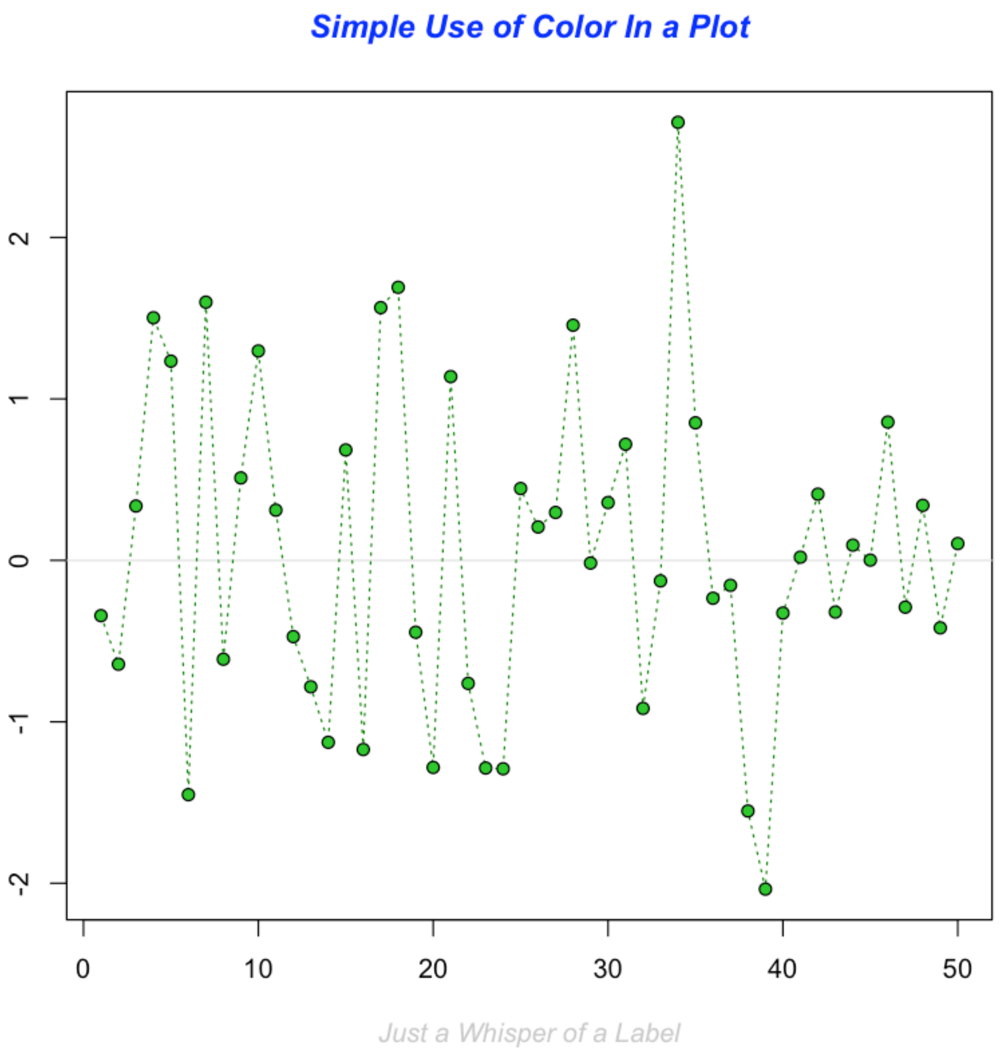
```
In [40]: vect
```

1st 2 **2nd** 20 **3rd** 6 **4th** 8

```
In [41]: demo(graphics)
```

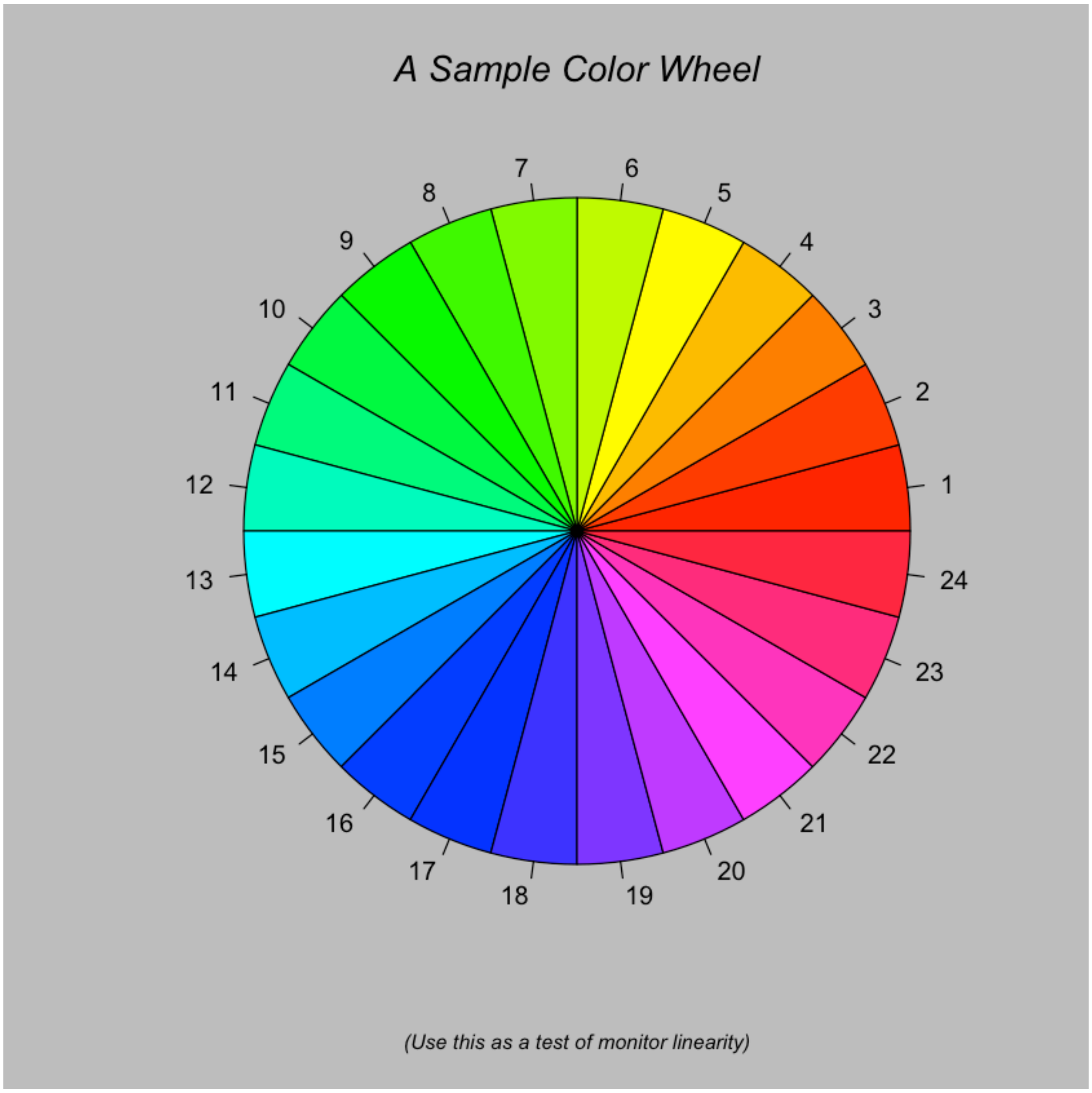
```


        demo(graphics)
        ---- ~~~~~~~~~

> #  Copyright (C) 1997-2009 The R Core Team
>
> require(datasets)

> require(grDevices); require(graphics)

> ## Here is some code which illustrates some of the differences between
> ## R and S graphics capabilities.  Note that colors are generally specified
> ## by a character string name (taken from the X11 rgb.txt file) and that line
> ## textures are given similarly.  The parameter "bg" sets the background
> ## parameter for the plot and there is also an "fg" parameter which sets
> ## the foreground color.
>
>
> x <- stats::rnorm(50)

> opar <- par(bg = "white")

> plot(x, ann = FALSE, type = "n")

> abline(h = 0, col = gray(.90))

> lines(x, col = "green4", lty = "dotted")
```

```
> points(x, bg = "limegreen", pch = 21)

> title(main = "Simple Use of Color In a Plot",
+       xlab = "Just a Whisper of a Label",
+       col.main = "blue", col.lab = gray(.8),
+       cex.main = 1.2, cex.lab = 1.0, font.main = 4, font.lab = 3)

> ## A little color wheel.         This code just plots equally spaced hues in
> ## a pie chart.        If you have a cheap SVGA monitor (like me) you will
> ## probably find that numerically equispaced does not mean visually
> ## equispaced.  On my display at home, these colors tend to cluster at
> ## the RGB primaries.  On the other hand on the SGI Indy at work the
> ## effect is near perfect.
>
> par(bg = "gray")

> pie(rep(1,24), col = rainbow(24), radius = 0.9)
```



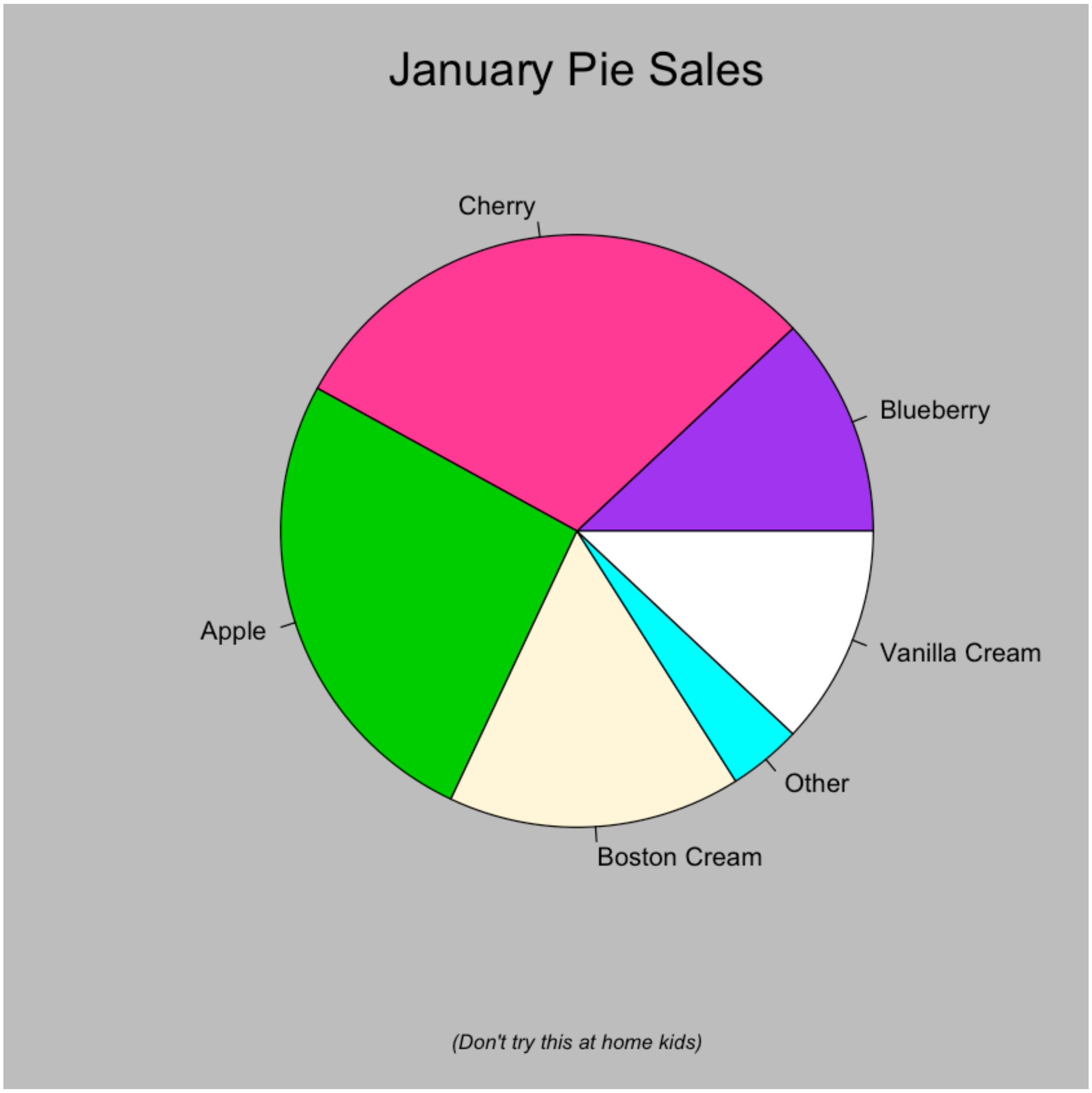
```
> title(main = "A Sample Color Wheel", cex.main = 1.4, font.main = 3)

> title(xlab = "(Use this as a test of monitor linearity)",
+       cex.lab = 0.8, font.lab = 3)

> ## We have already confessed to having these.  This is just showing off X11
> ## color names (and the example (from the postscript manual) is pretty "cute".
>
> pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)

> names(pie.sales) <- c("Blueberry", "Cherry",
+                       "Apple", "Boston Cream", "Other", "Vanilla Cream")
```

```
> pie(pie.sales,
+     col = c("purple","violetred1","green3","cornsilk","cyan","white"))
```

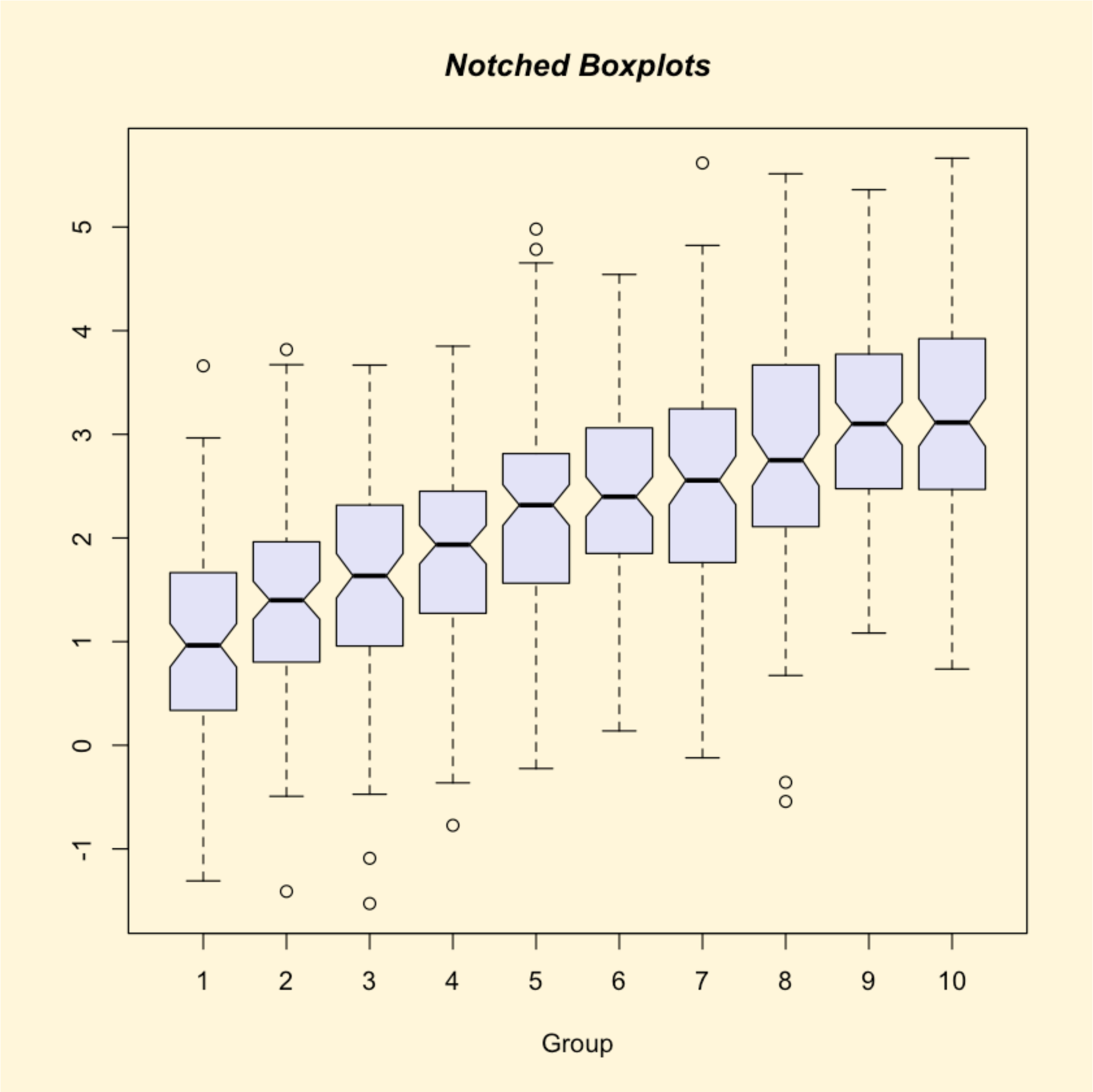


```
> title(main = "January Pie Sales", cex.main = 1.8, font.main = 1)

> title(xlab = "(Don't try this at home kids)", cex.lab = 0.8, font.lab = 3)

> ## Boxplots:  I couldn't resist the capability for filling the "box".
> ## The use of color seems like a useful addition, it focuses attention
> ## on the central bulk of the data.
```

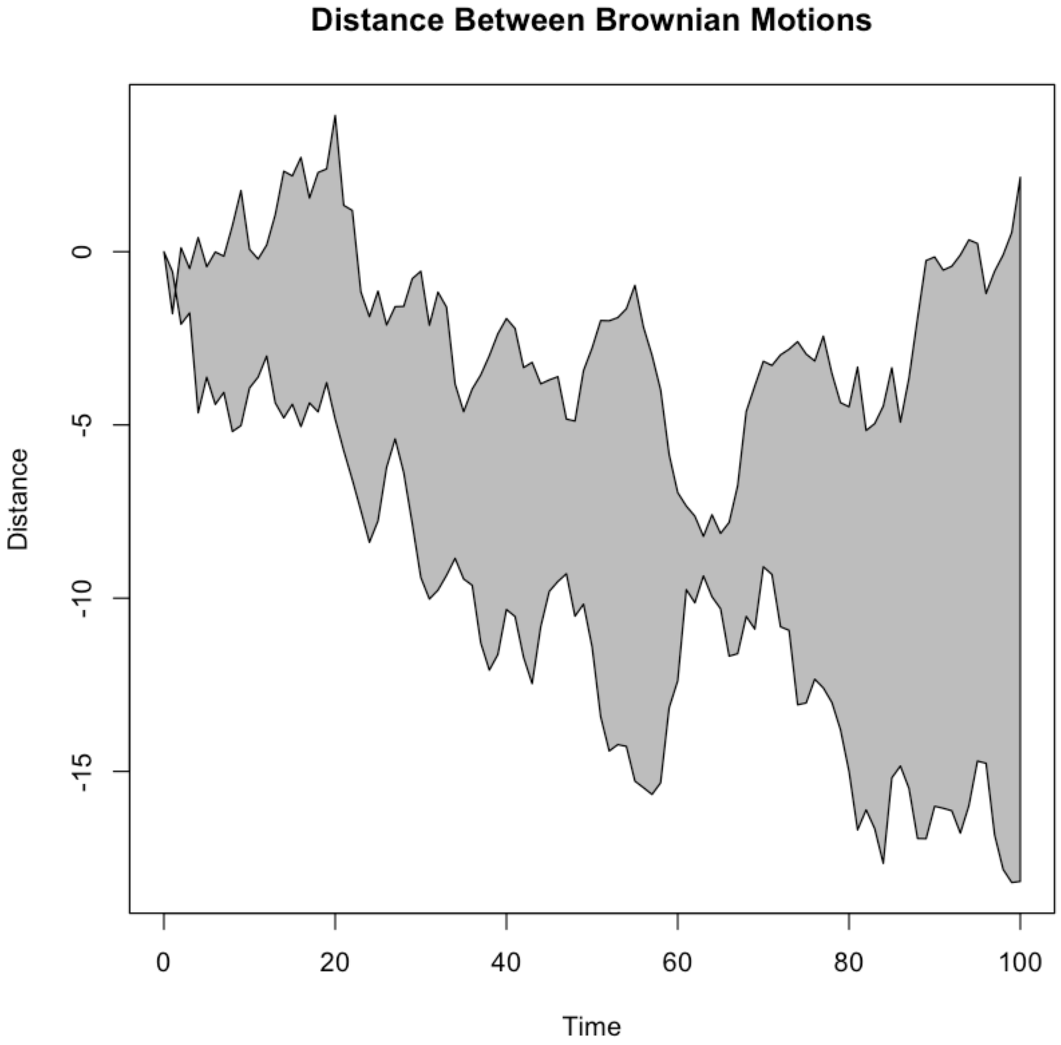
```
>
> par(bg="cornsilk")

> n <- 10

> g <- gl(n, 100, n*100)

> x <- rnorm(n*100) + sqrt(as.numeric(g))

> boxplot(split(x,g), col="lavender", notch=TRUE)
```



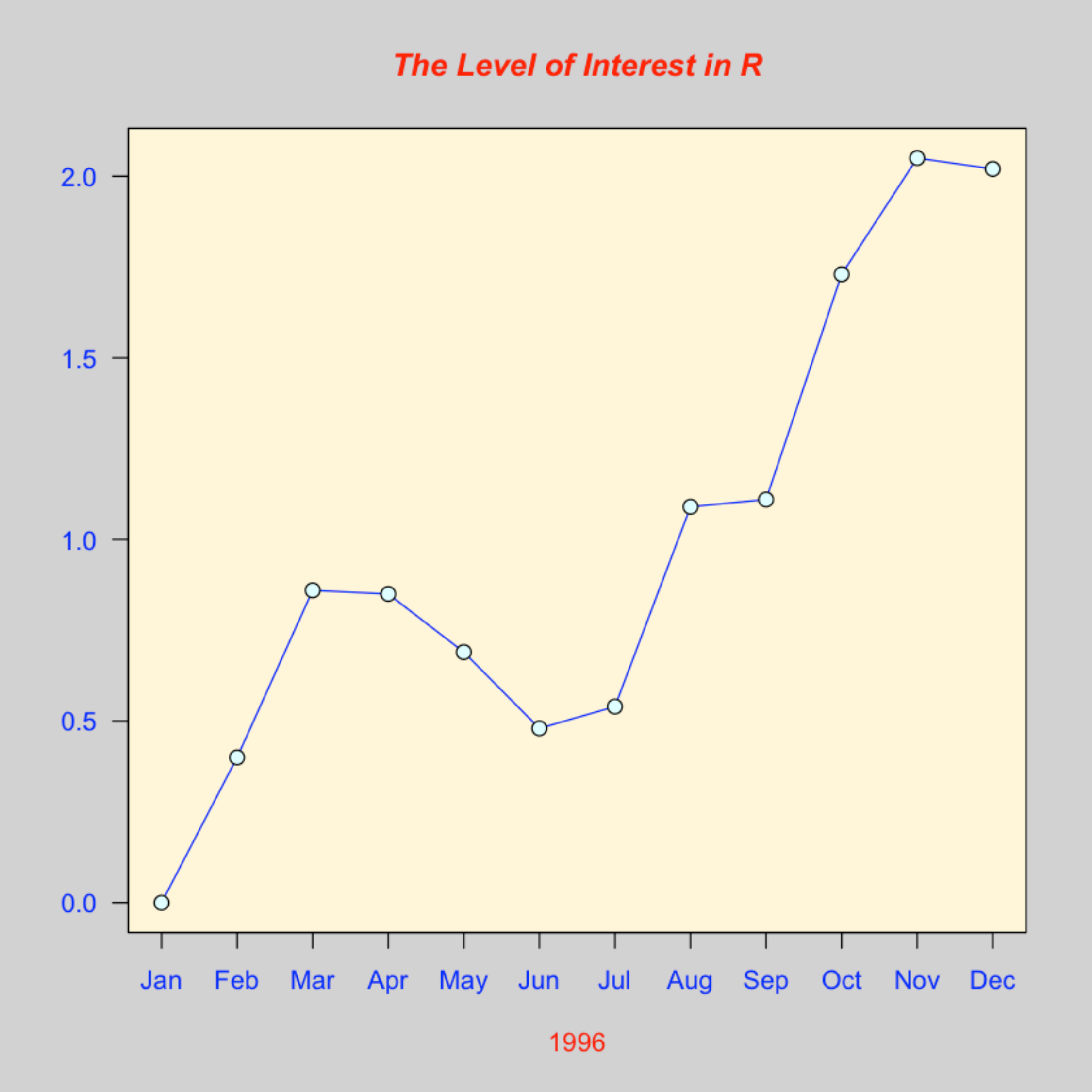
```
> title(main="Notched Boxplots", xlab="Group", font.main=4, font.lab=1)
> ## An example showing how to fill between curves.
>
> par(bg="white")

> n <- 100

> x <- c(0,cumsum(rnorm(n)))

> y <- c(0,cumsum(rnorm(n)))

> xx <- c(0:n, n:0)

> yy <- c(x, rev(y))

> plot(xx, yy, type="n", xlab="Time", ylab="Distance")
```



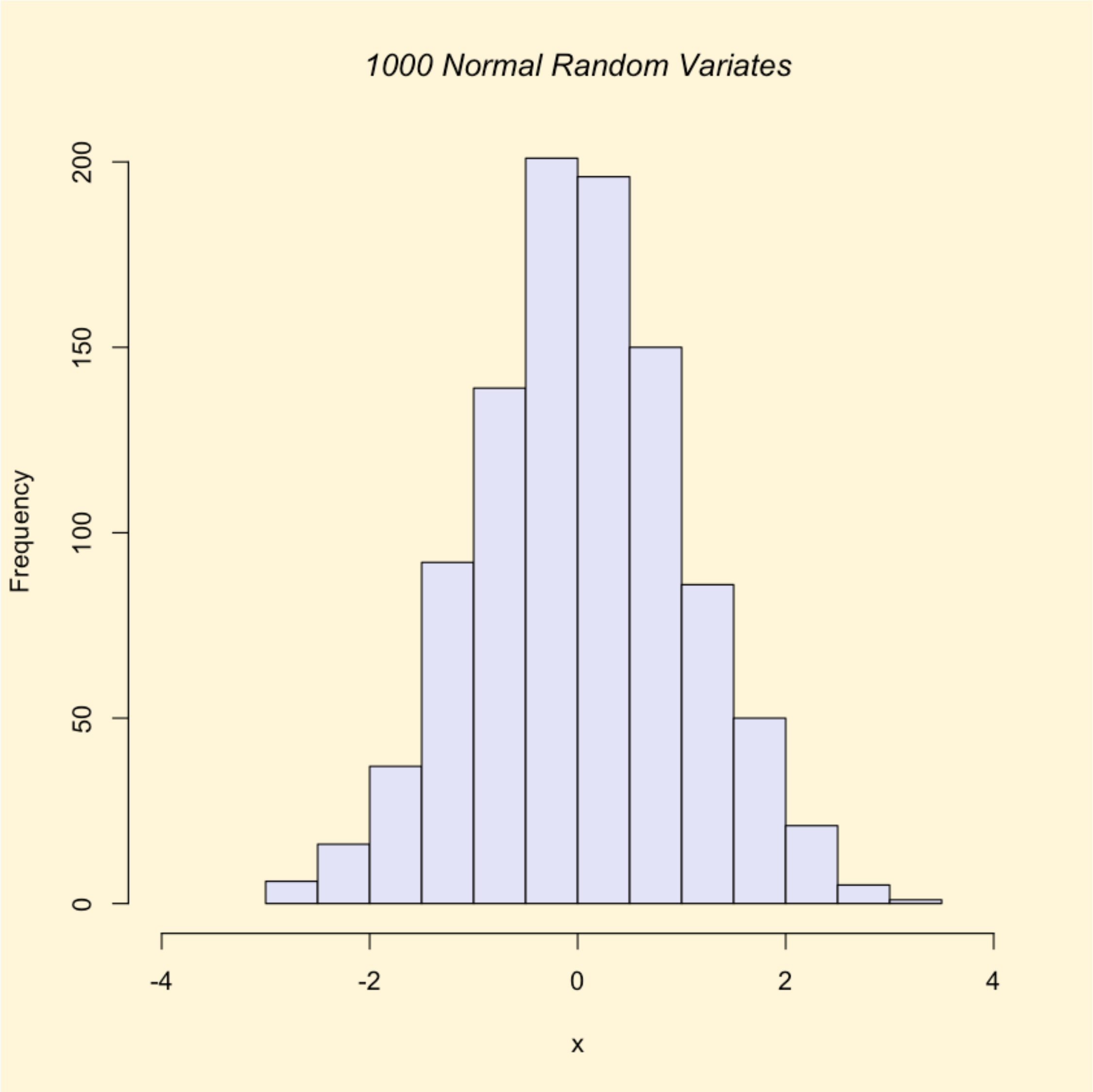
```
> polygon(xx, yy, col="gray")

> title("Distance Between Brownian Motions")

> ## Colored plot margins, axis labels and titles.         You do need to be
> ## careful with these kinds of effects.        It's easy to go completely
> ## over the top and you can end up with your lunch all over the keyboard.
> ## On the other hand, my market research clients love it.
>
> x <- c(0.00, 0.40, 0.86, 0.85, 0.69, 0.48, 0.54, 1.09, 1.11, 1.73, 2.05, 2.02)

> par(bg="lightgray")
> plot(x, type="n", axes=FALSE, ann=FALSE)
```



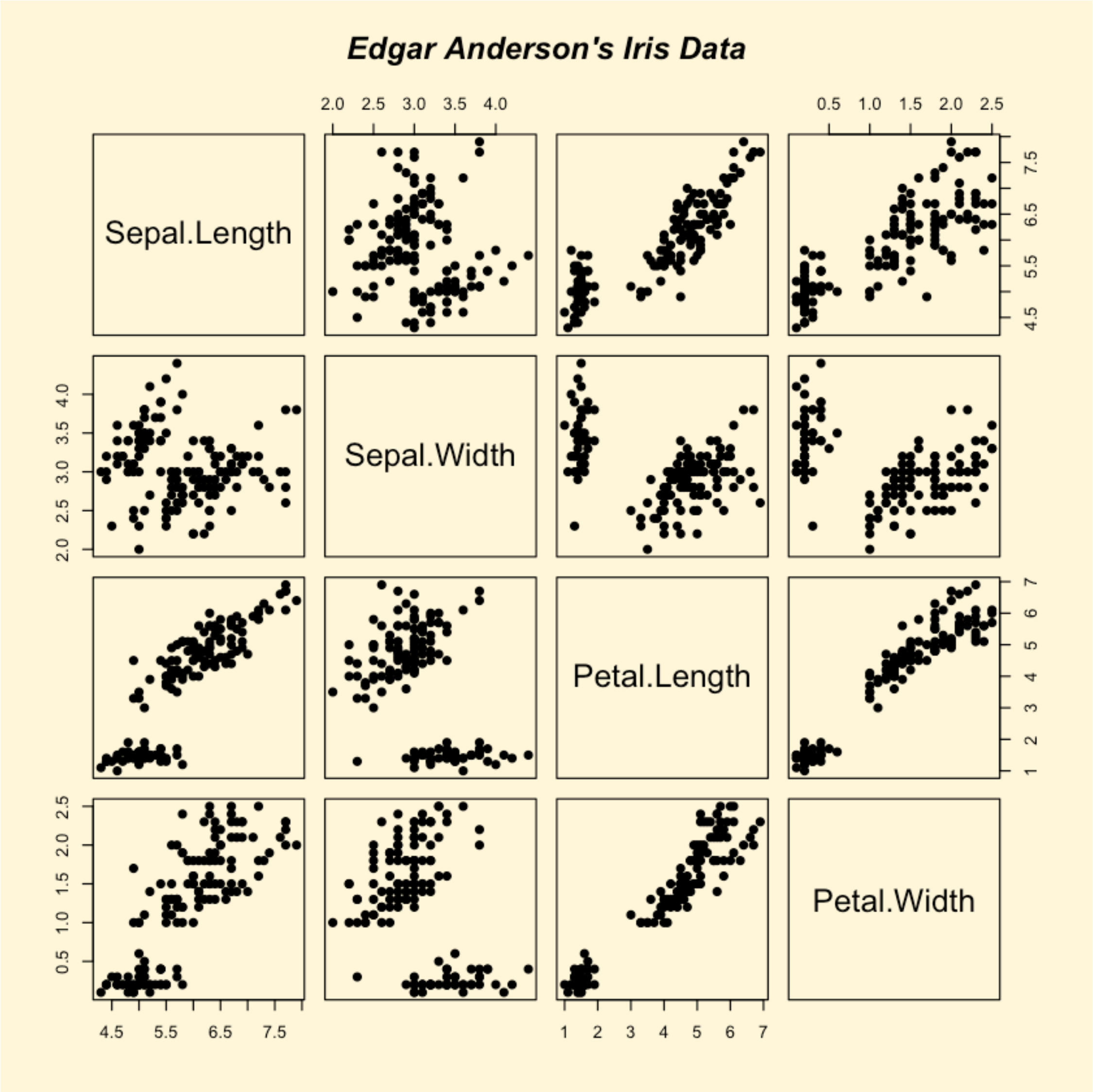
```
> usr <- par("usr")

> rect(usr[1], usr[3], usr[2], usr[4], col="cornsilk", border="black")

> lines(x, col="blue")

> points(x, pch=21, bg="lightcyan", cex=1.25)
> axis(2, col.axis="blue", las=1)

> axis(1, at=1:12, lab=month.abb, col.axis="blue")

> box()

> title(main= "The Level of Interest in R", font.main=4, col.main="red")

> title(xlab= "1996", col.lab="red")

> ## A filled histogram, showing how to change the font used for the
> ## main title without changing the other annotation.
>
> par(bg="cornsilk")

> x <- rnorm(1000)

> hist(x, xlim=range(-4, 4, x), col="lavender", main="")
```



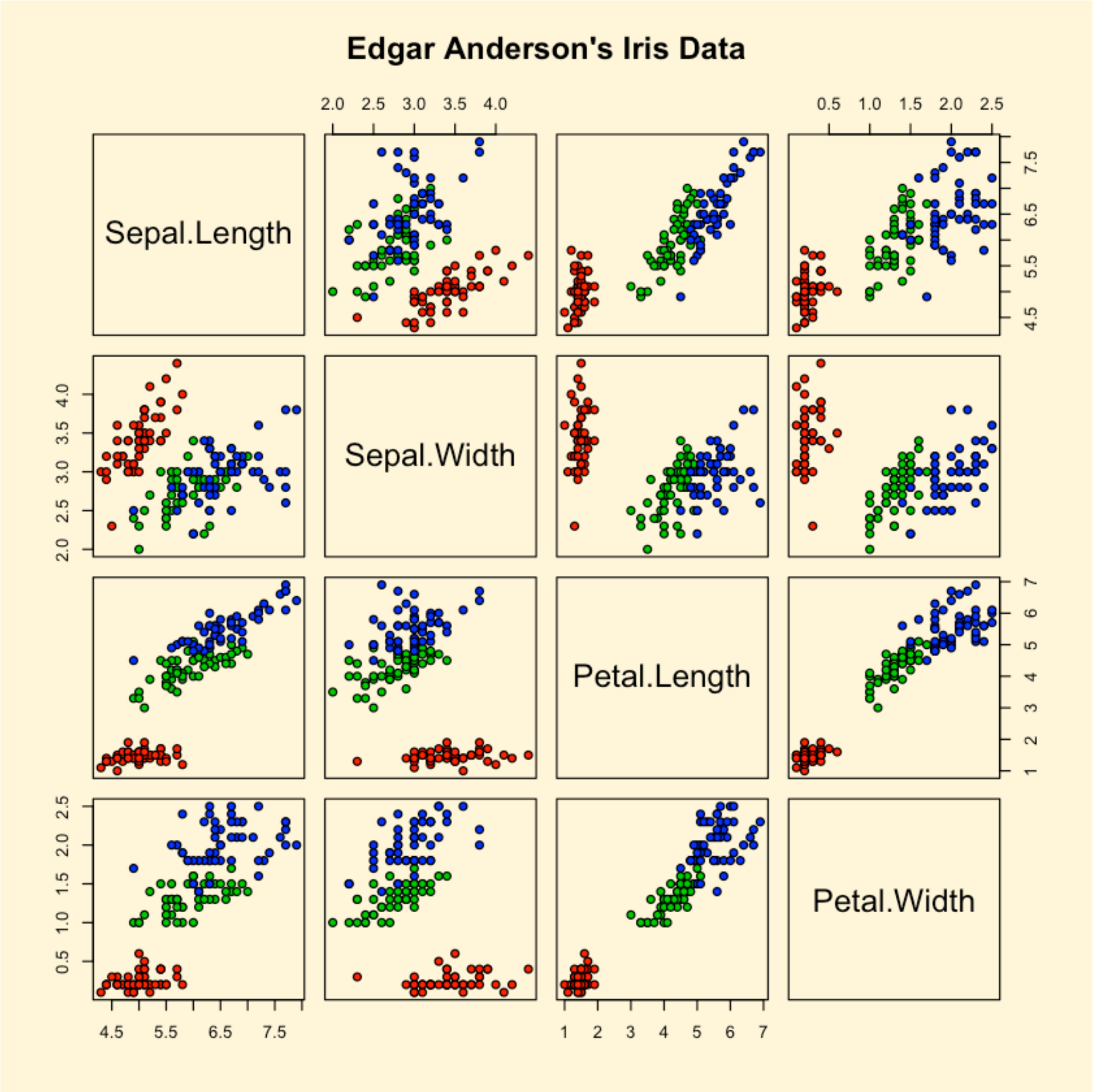
```
> title(main="1000 Normal Random Variates", font.main=3)

> ## A scatterplot matrix
> ## The good old Iris data (yet again)
>
> pairs(iris[1:4], main="Edgar Anderson's Iris Data", font.main=4, pch=19)
```



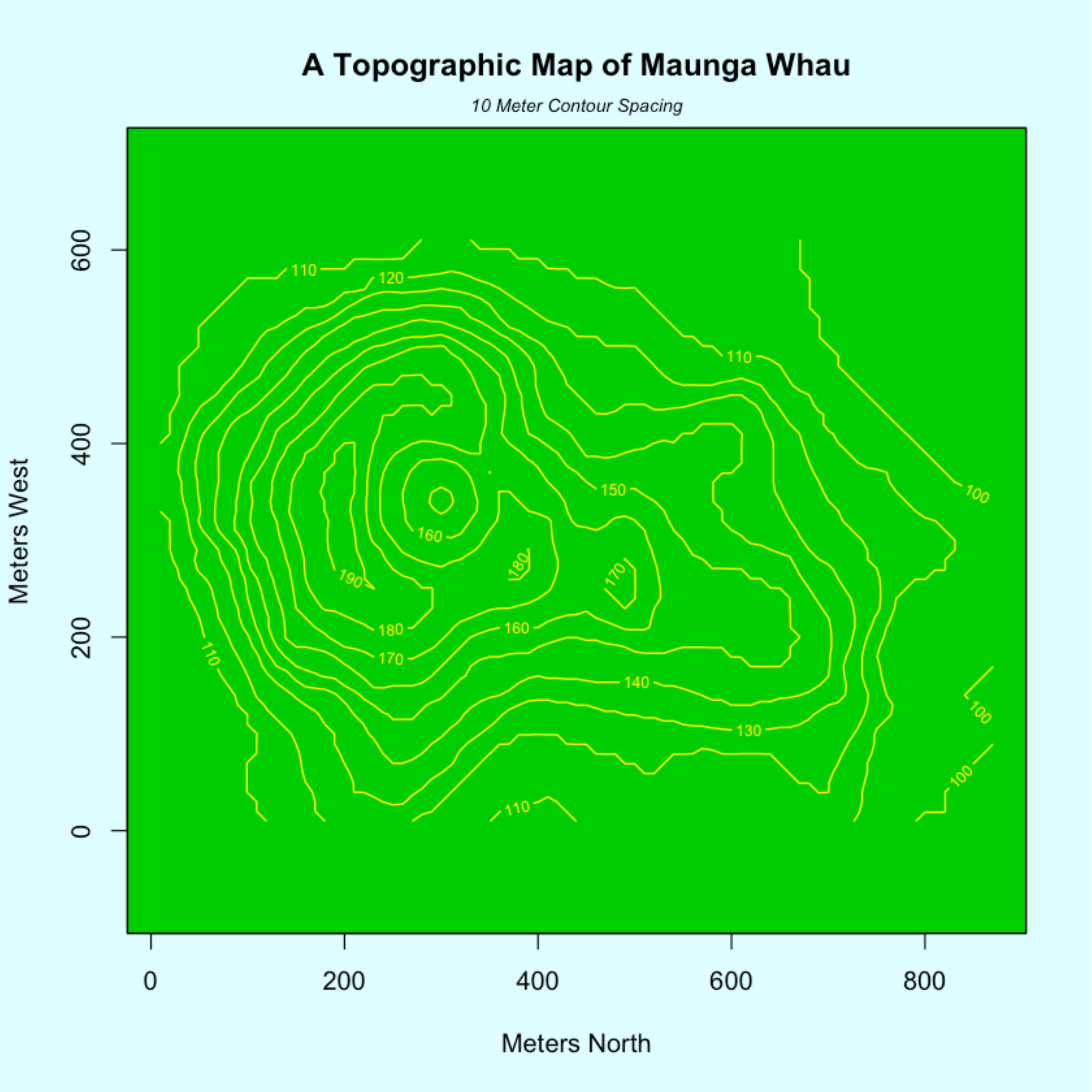
```
> pairs(iris[1:4], main="Edgar Anderson's Iris Data", pch=21,
+       bg = c("red", "green3", "blue")[unclass(iris$Species)])
```



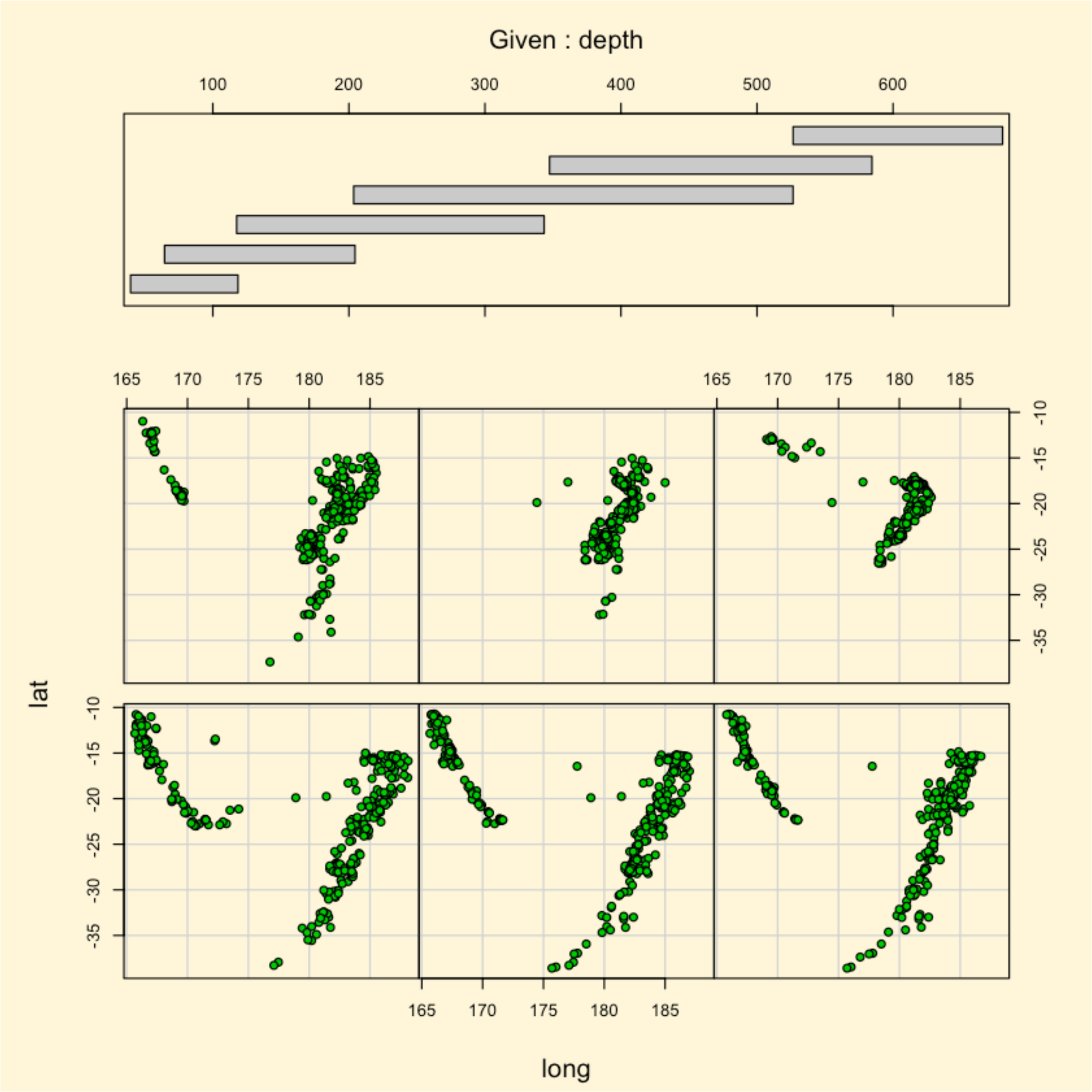
```
> ## Contour plotting
> ## This produces a topographic map of one of Auckland's many volcanic "peaks".
>
> x <- 10*1:nrow(volcano)

> y <- 10*1:ncol(volcano)

> lev <- pretty(range(volcano), 10)

> par(bg = "lightcyan")

> pin <- par("pin")
> xdelta <- diff(range(x))

> ydelta <- diff(range(y))

> xscale <- pin[1]/xdelta

> yscale <- pin[2]/ydelta

> scale <- min(xscale, yscale)

> xadd <- 0.5*(pin[1]/scale - xdelta)

> yadd <- 0.5*(pin[2]/scale - ydelta)

> plot(numeric(0), numeric(0),
+      xlim = range(x)+c(-1,1)*xadd, ylim = range(y)+c(-1,1)*yadd,
+      type = "n", ann = FALSE)
```



```
> usr <- par("usr")

> rect(usr[1], usr[3], usr[2], usr[4], col="green3")

> contour(x, y, volcano, levels = lev, col="yellow", lty="solid", add=TRUE)

> box()

> title("A Topographic Map of Maunga Whau", font= 4)

> title(xlab = "Meters North", ylab = "Meters West", font= 3)
> mtext("10 Meter Contour Spacing", side=3, line=0.35, outer=FALSE,
+       at = mean(par("usr")[1:2]), cex=0.7, font=3)

> ## Conditioning plots
>
> par(bg="cornsilk")

> coplot(lat ~ long | depth, data = quakes, pch = 21, bg = "green3")
```



```
> par(opar)
```



```
In [42]: # A function name generateSquares which prints the square of the given number.

         generateSquares <- function(x){
             return(x^2)
         }

In [43]: generateSquares(4)
```

16

```
In [44]: # A function with three arguments a,b,c.

         func_arguments <- function (a=1,b=2,c=3){
             res <- a + (b * c)
             print(res)
         }
```

```
In [45]: # Pass values to arguments with index.

         func_arguments(4,5,6)
```

```
[1] 34
```

```
In [46]: # Passing values to arguments using name of arguments

         func_arguments(a=4,b=5,c=2)
```

```
[1] 14
```

```
In [47]: mydataframe <- data.frame(

         # Creates a vector (list) starting from 1 to 5.
         stu_id = c(1:5),
         # Creates a vector (list) using the below names.
         stu_name = c("Bob","Pat","Jane","Peter","Han"),
         # To avoid problems when reassigning values within a dataframe. We use stringsAsFacto
         stringsAsFactors = FALSE
         )
```

```
In [48]: res  <- data.frame(mydataframe$stu_id,mydataframe$stu_name)
```

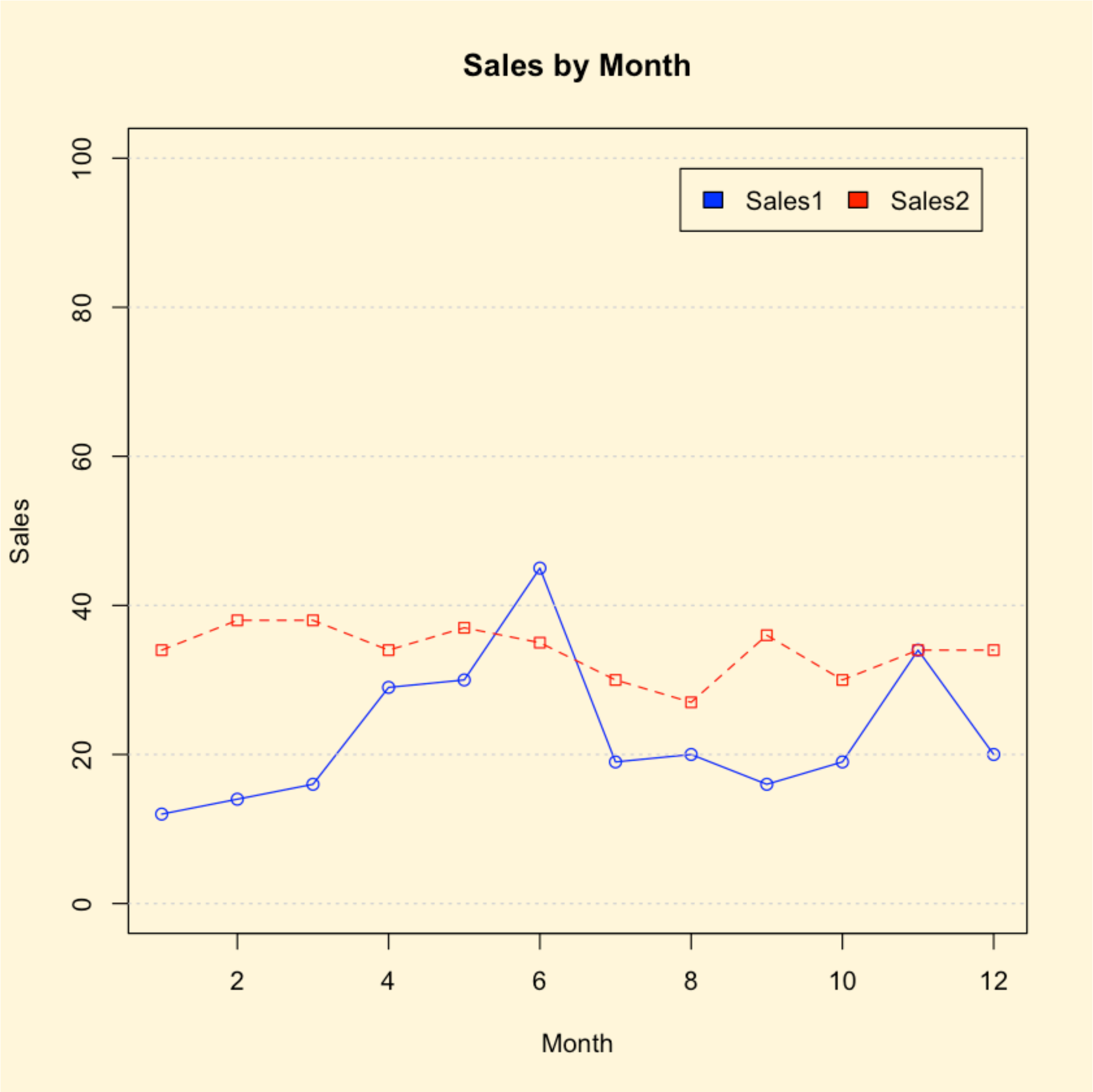
```
In [49]: res
```

mydataframe.stu_id	mydataframe.stu_name
1	Bob
2	Pat
3	Jane
4	Peter
5	Han

0.3 Problem 1

```
In [6]: # pch - Generates a Symbol
        # lty - line type (Solid, Dashled line)
        # col - Color
        # Inset - Place to display on the grid
        # rpois - Generates multinomial or multi-Poission random variates based on an Aitchiso
        # nx, ny - Horizontal and Vertical lines.

        # A vector of values assigned to variable 'sales1'.
```

```r
sales1 <- c(12,14,16,29,30,45,19,20,16,19,34,20)

# A vector of random values between 12 to 34 are assigned to 'sales2'.
sales2 <- rpois(12,34)

# Background Color
par(bg="cornsilk")

# Plot sales1 with color blue, with each point using 'o'.
# y-axis extends from 0 to 100.
# xlab & ylab -> x label and y label
# title - main -> Main title
# title - sub -> Sub title
plot(sales1, col="blue", type="o", ylim=c(0,100), xlab="Month", ylab="Sales" )
title(main="Sales by Month")

# Plot a line using randomly generated data sales2.
# pch code 22 generates a square.
# lty - Line type (Solid or Dashed Line)
lines(sales2, type="o", pch=22, lty=2, col="red")
grid(nx=NA, ny=NULL)

# Fix a legend in topright corner of the grid with a margin of .05.
# Create a vector (sales1, sales2) along with colors set as blue and red respectively.
legend("topright", inset=.05, c("Sales1","Sales2"), fill=c("blue","red"), horiz=TRUE)
```

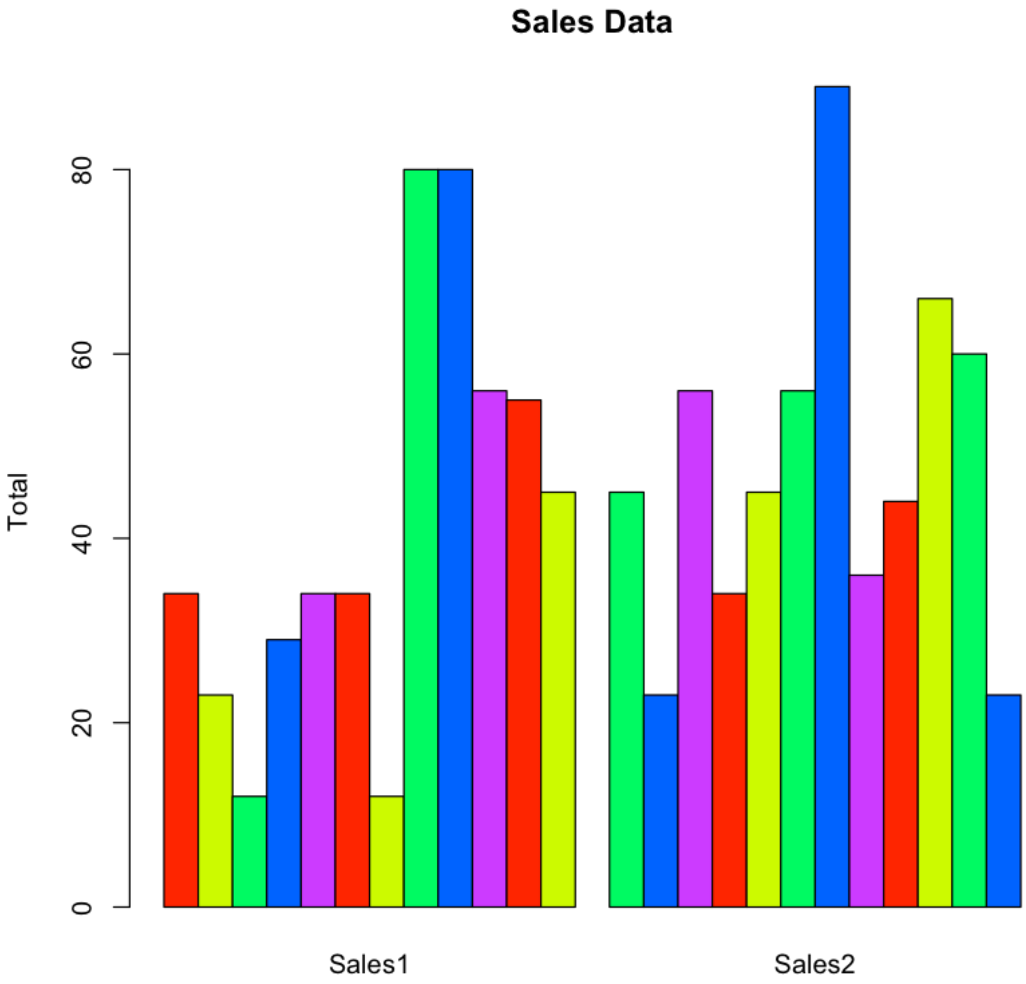


0.4 Problem 2

```
In [10]: # read.delim -> To read a delimited txt file. Here salesdata.txt is delimited with sp
         # sales<-read.table(file.choose(), header=T)
         sales <- read.delim('salesdata.txt')

In [11]: sales
```

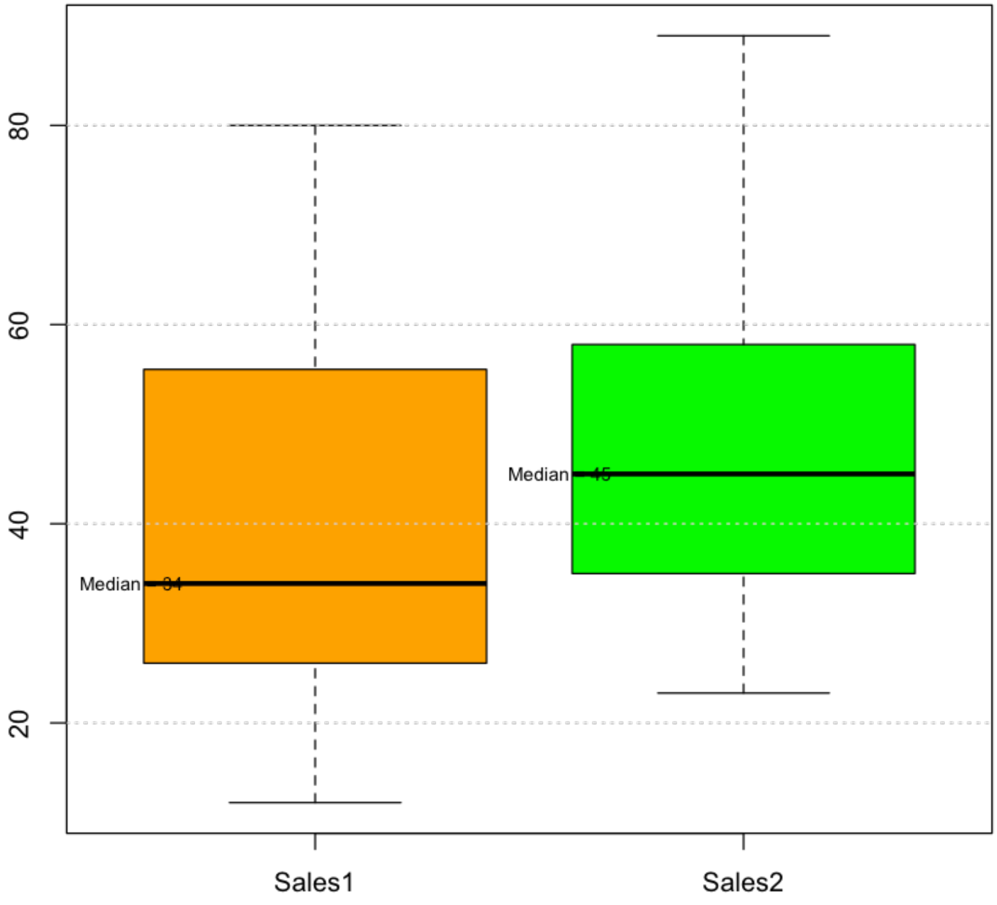
Sales1	Sales2
34	45
23	23
12	56
29	34
34	45
34	56
12	89
80	36
80	44
56	66
55	60
45	23

```
In [13]: # as.matrix - > Returns all values of vector into a matrix
         # barplot -> Plot a bar plot using the data
         # beside -> When False, they stack horizontally. When True, columns are potrayed as s
         barplot(as.matrix(sales), main="Sales Data", ylab= "Total",beside=T, col=rainbow(5))
```



0.5 Problem 3

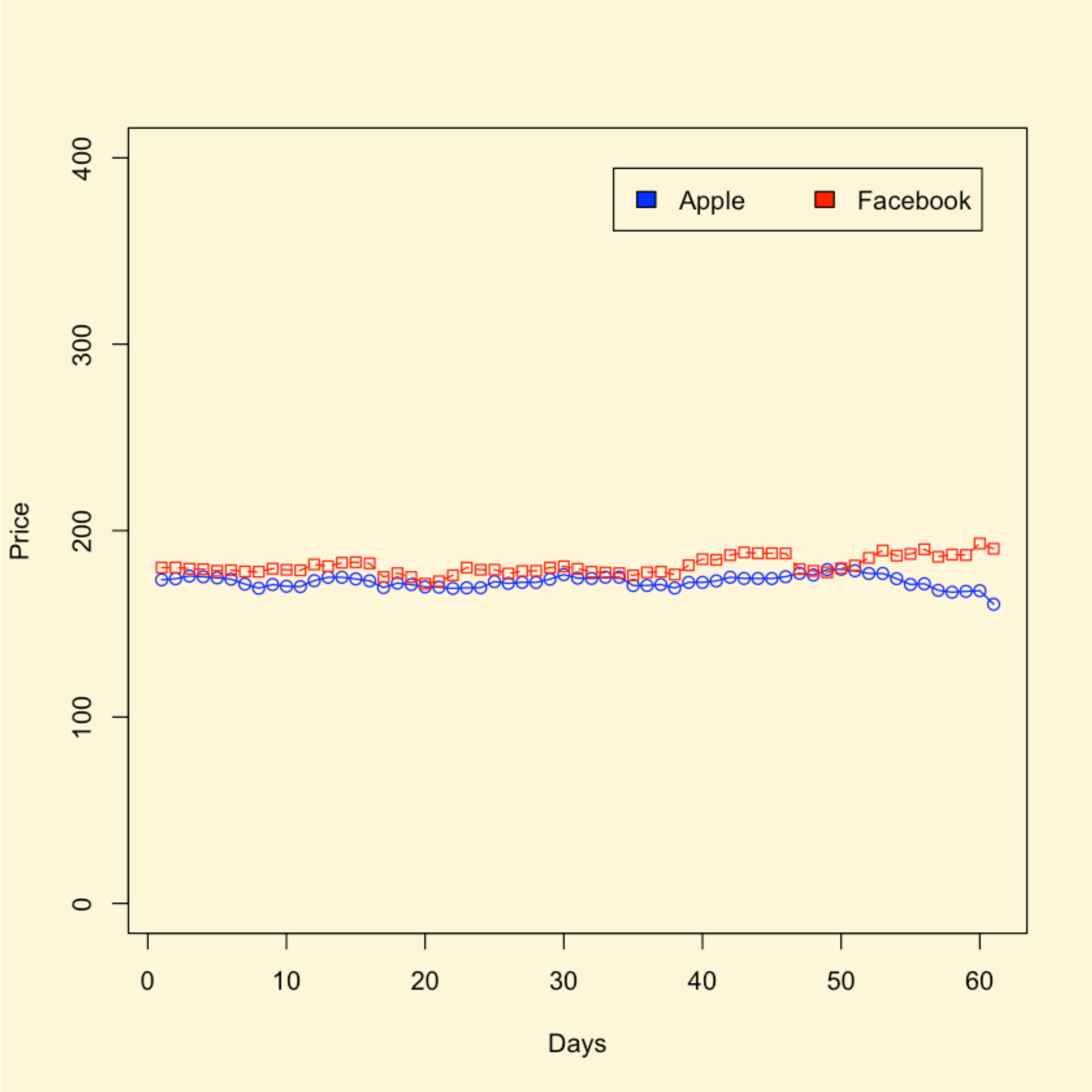
```
In [20]: # boxplot for sales. Two colors for two columns
         fn<-boxplot(sales,col=c("orange","green"))$stats

         # fn converts a argument to function
         # text places a text.
         # cex -> Character size

         text(1.45, fn[3,2], paste("Median =", fn[3,2]), adj=0, cex=.7)
         text(0.45, fn[3,1],paste("Median =", fn[3,1]), adj=0, cex=.7)
         grid(nx=NA, ny=NULL)
```

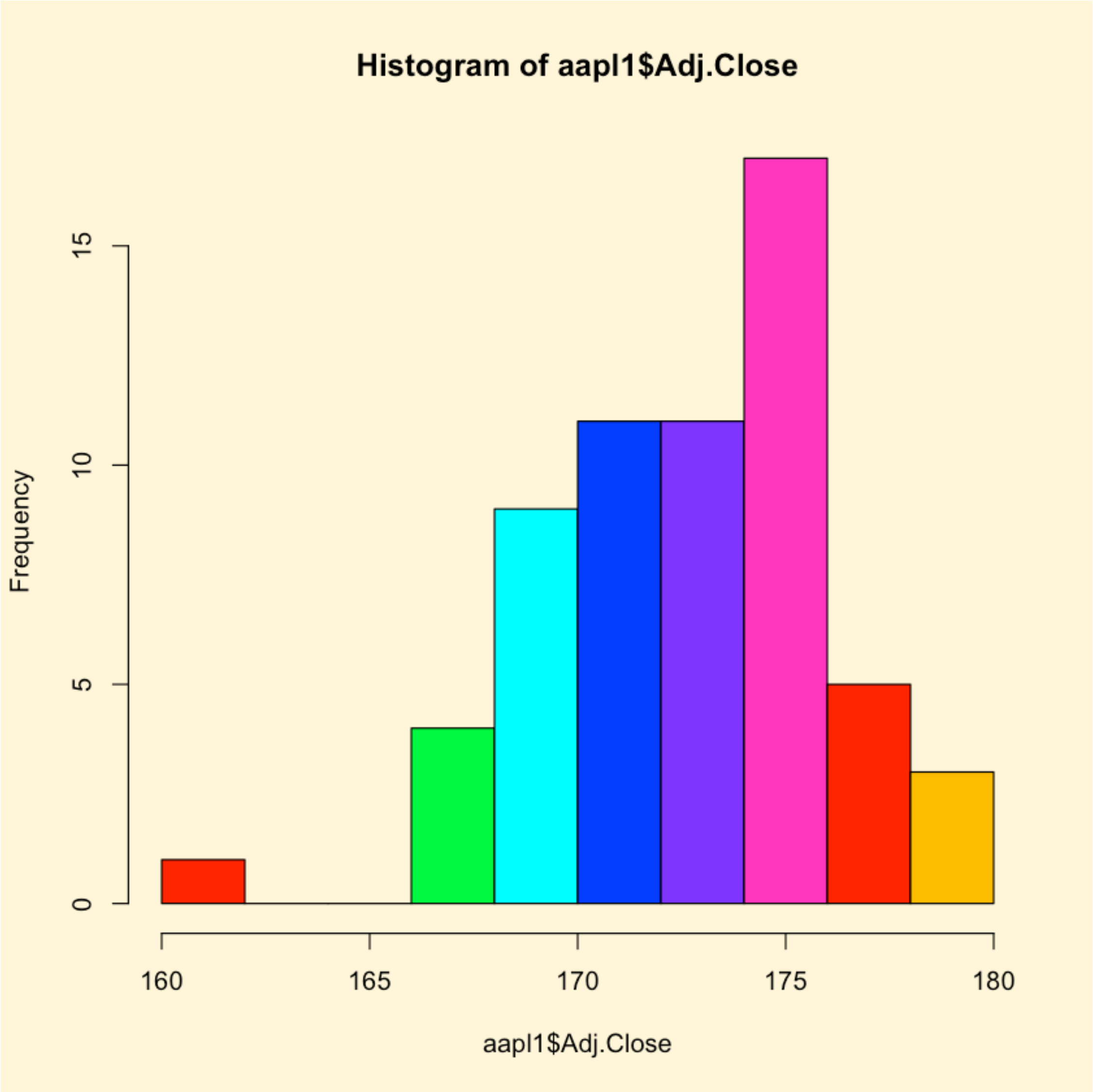


0.6 Problem 4

```
In [28]: # Read FB.csv file to fb1 variable
         # Similarly aapl1
         # plot 'adj close' from appl1 with blue color and with 'o'.
         # plot 'adj close' from fb1 with red color and with a symbol 'pch = 22'.
         # Histogram of column 'Adj Close' from appl1.

         fb1<-read.csv('FB.csv')
         aapl1<-read.csv('AAPL.csv')
         par(bg="cornsilk")
         plot(aapl1$Adj.Close, col="blue", type="o", ylim=c(0,400), xlab="Days", ylab="Price"
         lines(fb1$Adj.Close, type="o", pch=22, lty=2, col="red")
```

```
legend("topright", inset=.05, c("Apple","Facebook"), fill=c("blue","red"), horiz=TRUE
hist(aapl1$Adj.Close, col=rainbow(8))
```





0.7 Problem 5

```
In [29]: # Displays all datasets.
         data()
```

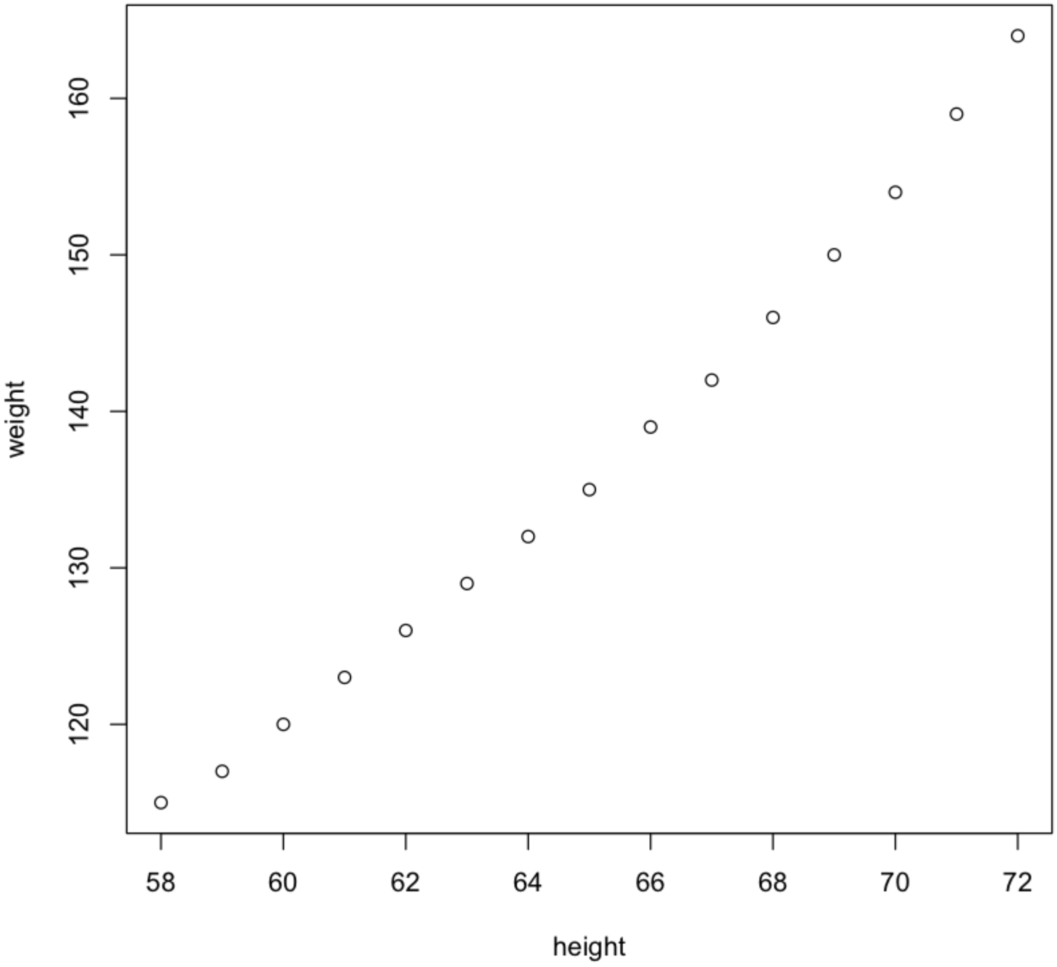
```
In [30]: head(women)
```

height	weight
58	115
59	117
60	120
61	123
62	126
63	129

```
In [31]: summary(women)
```

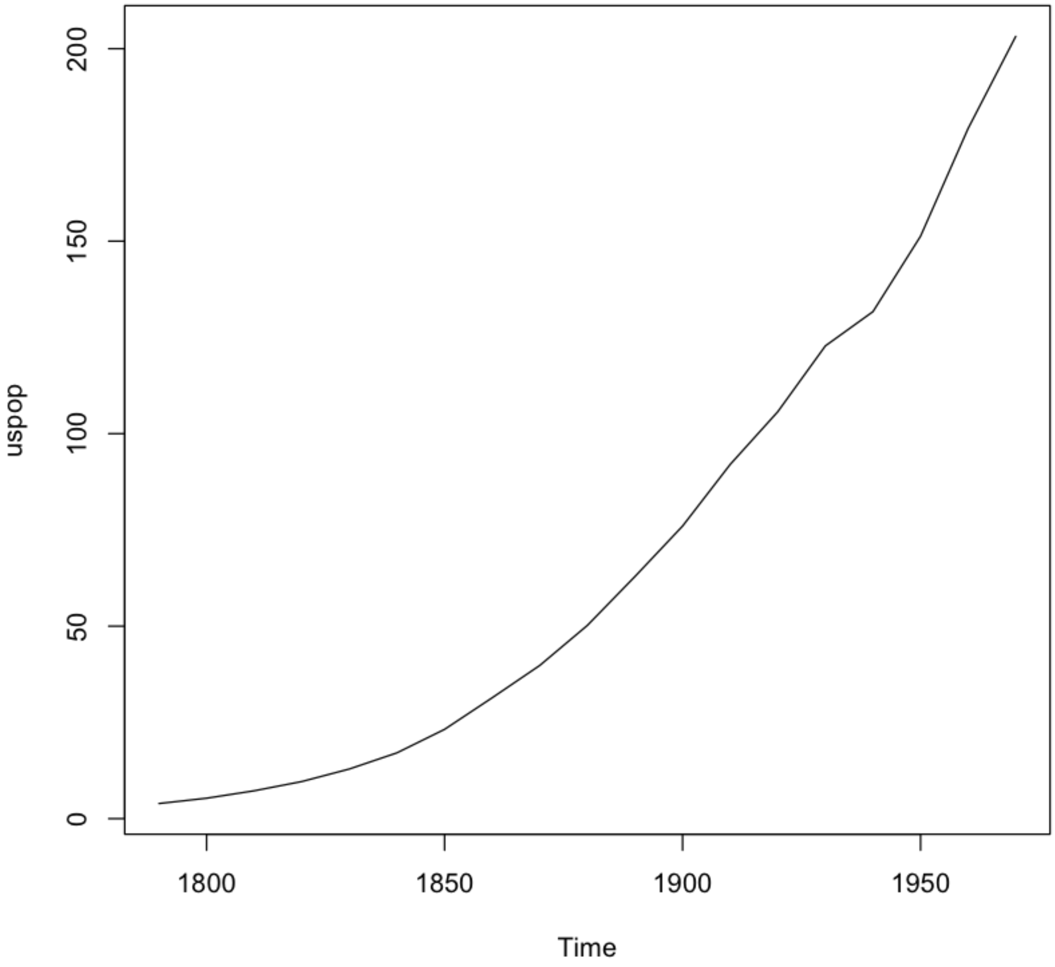
```
     height          weight     
 Min.   :58.0   Min.   :115.0  
 1st Qu.:61.5   1st Qu.:124.5  
 Median :65.0   Median :135.0  
 Mean   :65.0   Mean   :136.7  
 3rd Qu.:68.5   3rd Qu.:148.0  
 Max.   :72.0   Max.   :164.0  
```

```
In [32]: plot(women)
```



```
In [33]: head(uspop)
         plot(uspop)
```

1. 3.93 2. 5.31 3. 7.24 4. 9.64 5. 12.9 6. 17.1

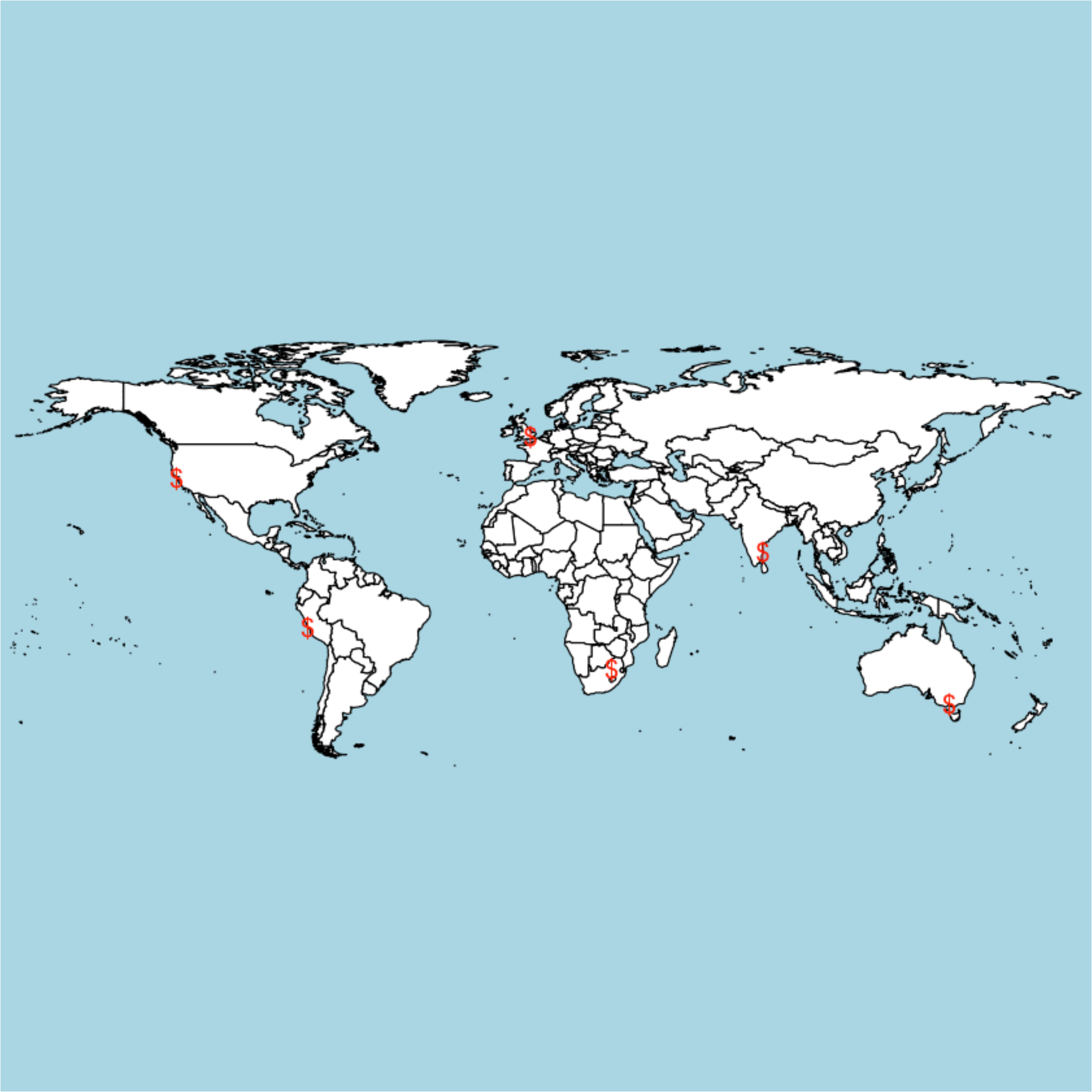


0.8 Problem 6

```
In [34]: # Use libraries ggmap, maptools.
         # register_google(Key = '') -> API Key is given to access API.
         # All the places to point on map are given into a vector named 'visited'.
         # Get Latitude and Longitude of each place and plot it on map.

         library("ggmap")
         library("maptools")
         library(maps)
```

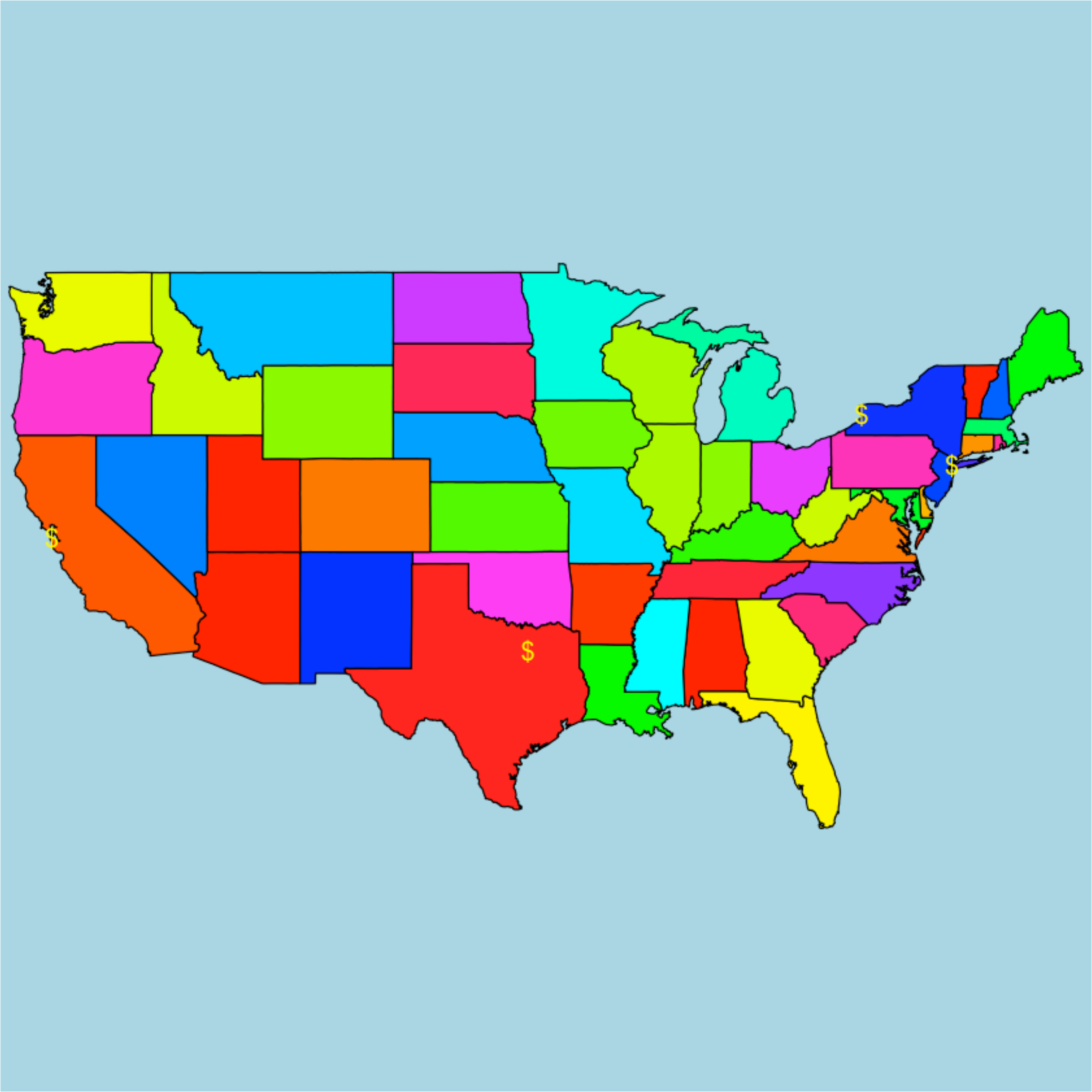
```
# Please provide API.
register_google(key = "")
visited <- c("SFO", "Chennai", "London", "Melbourne","Lima,Peru", "Johannesbury, SA")
ll.visited <- geocode(visited)
visit.x <- ll.visited$lon
visit.y <- ll.visited$lat
map("world", fill=TRUE, col="white", bg="lightblue", ylim=c(-60, 90), mar=c(0,0,0,0))
points(visit.x,visit.y, col="red", pch=36)
```

```
Loading required package: ggplot2
Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
Please cite ggmap if you use it! See citation("ggmap") for details.
Loading required package: sp
Checking rgeos availability: FALSE
         Note: when rgeos is not available, polygon geometry         computations in maptools d
         which has a restricted licence. It is disabled by default;
         to enable gpclib, type gpclibPermit()
Source : https://maps.googleapis.com/maps/api/geocode/json?address=SFO&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Chennai&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=London&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Melbourne&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Lima,Peru&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Johannesbury,+SA&key=xxx-Sm5
```



```
In [39]: # Use libraries ggmap, maptools.
         # All the places to point on map are given into a vector named 'visited'.
         # Get Latitude and Longitude of each place and plot it on map.

         library("ggmap")
         library("maptools")
         library(maps)
         visited <- c("SFO", "New York", "Buffalo", "Dallas, TX")
         ll.visited <- geocode(visited)
         visit.x <- ll.visited$lon
         visit.y <- ll.visited$lat
         map("state", fill=TRUE, col=rainbow(50), bg="lightblue", mar=c(0,0,0,0))
         points(visit.x,visit.y, col="yellow", pch=36)
```

```
Source : https://maps.googleapis.com/maps/api/geocode/json?address=SFO&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=New+York&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Buffalo&key=xxx-Sm541do
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Dallas,+TX&key=xxx-Sm541do
```

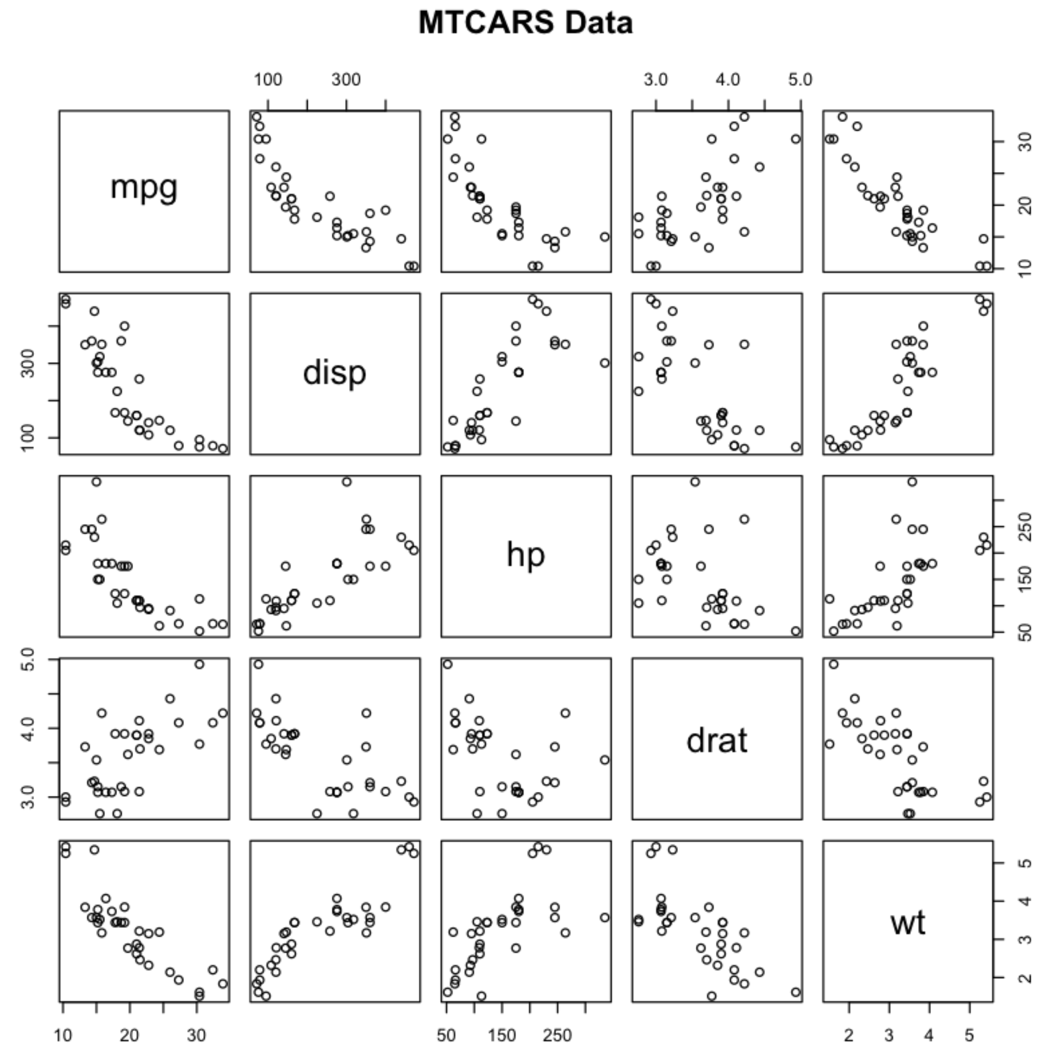


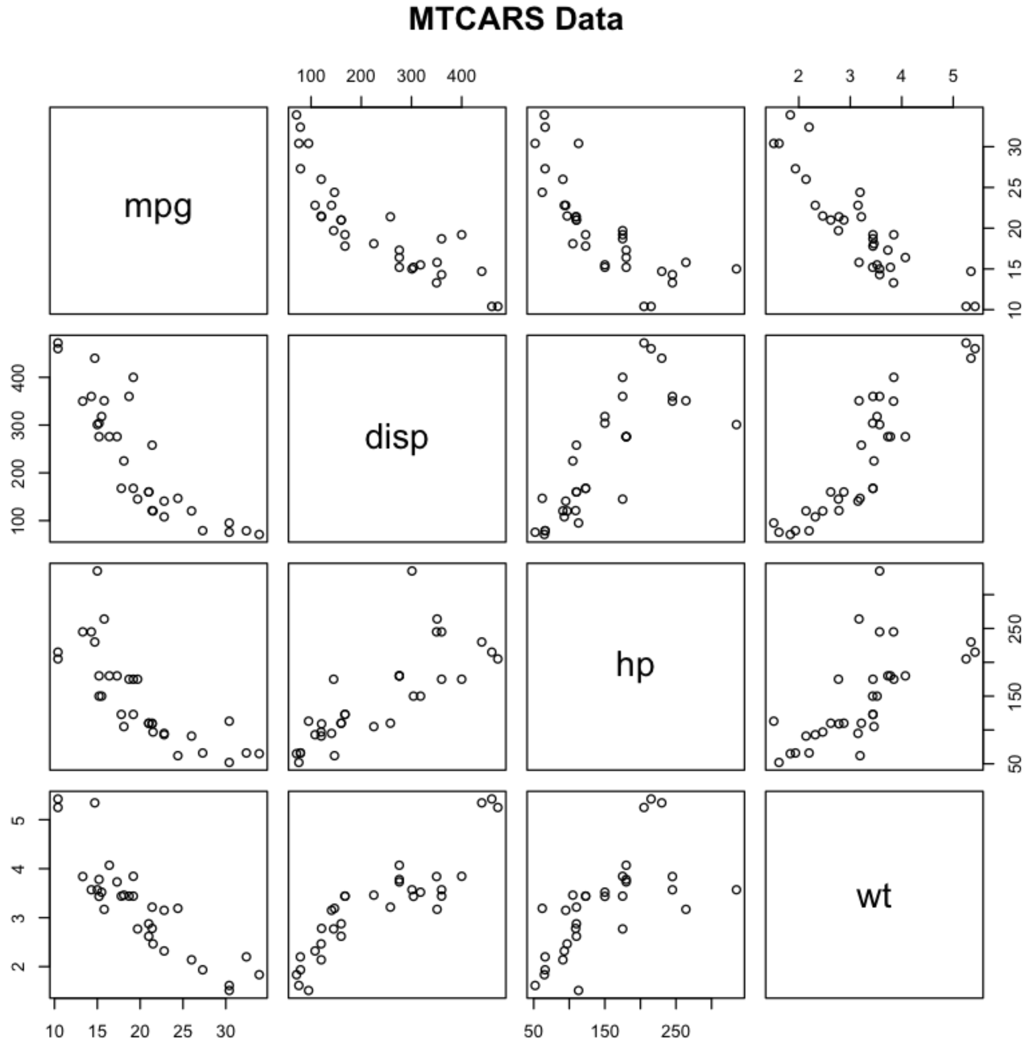
0.9 Problem 7

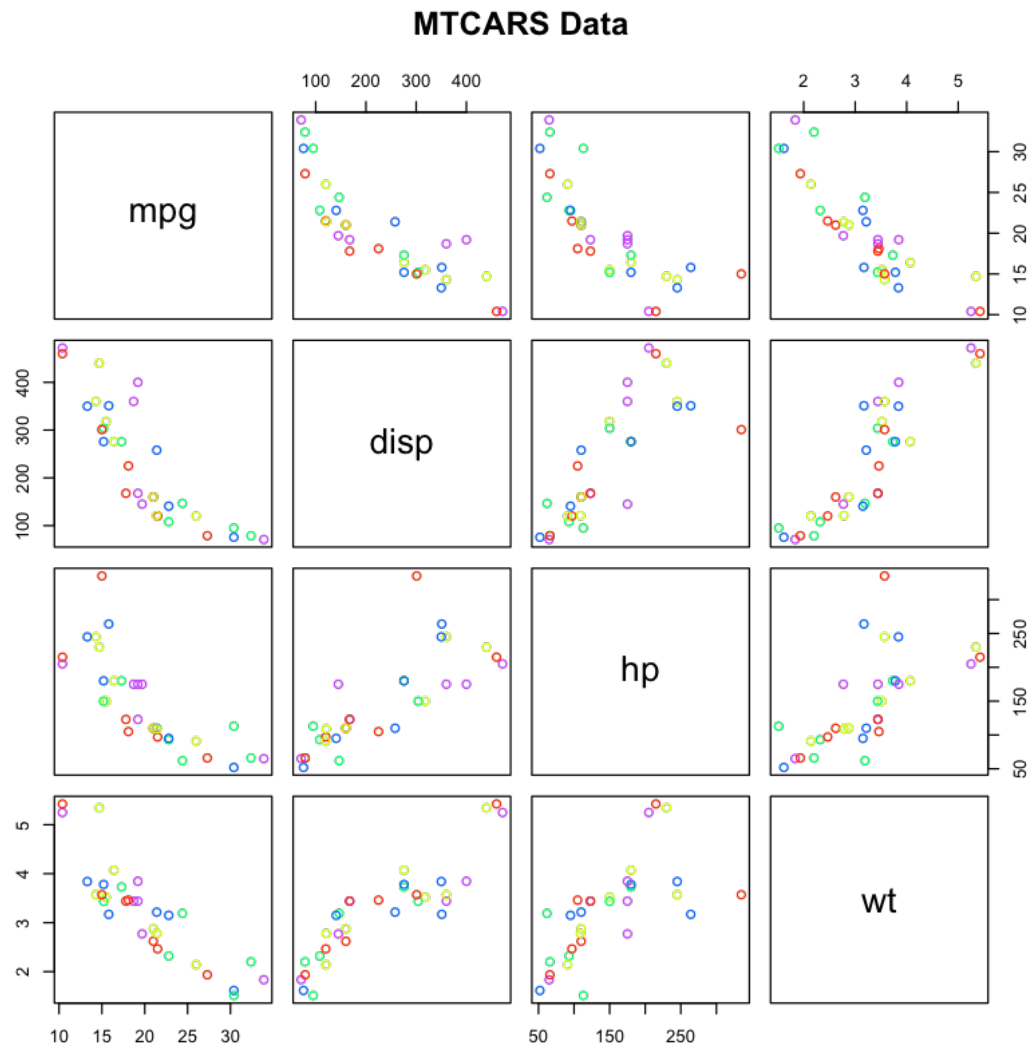
```
In [42]: attach(mtcars)
         head(mtcars)
         plot(mtcars[c(1,3,4,5,6)], main="MTCARS Data")
         plot(mtcars[c(1,3,4,6)], main="MTCARS Data")
         plot(mtcars[c(1,3,4,6)], col=rainbow(5),main="MTCARS Data")
```

```
The following object is masked from package:ggplot2:

    mpg
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

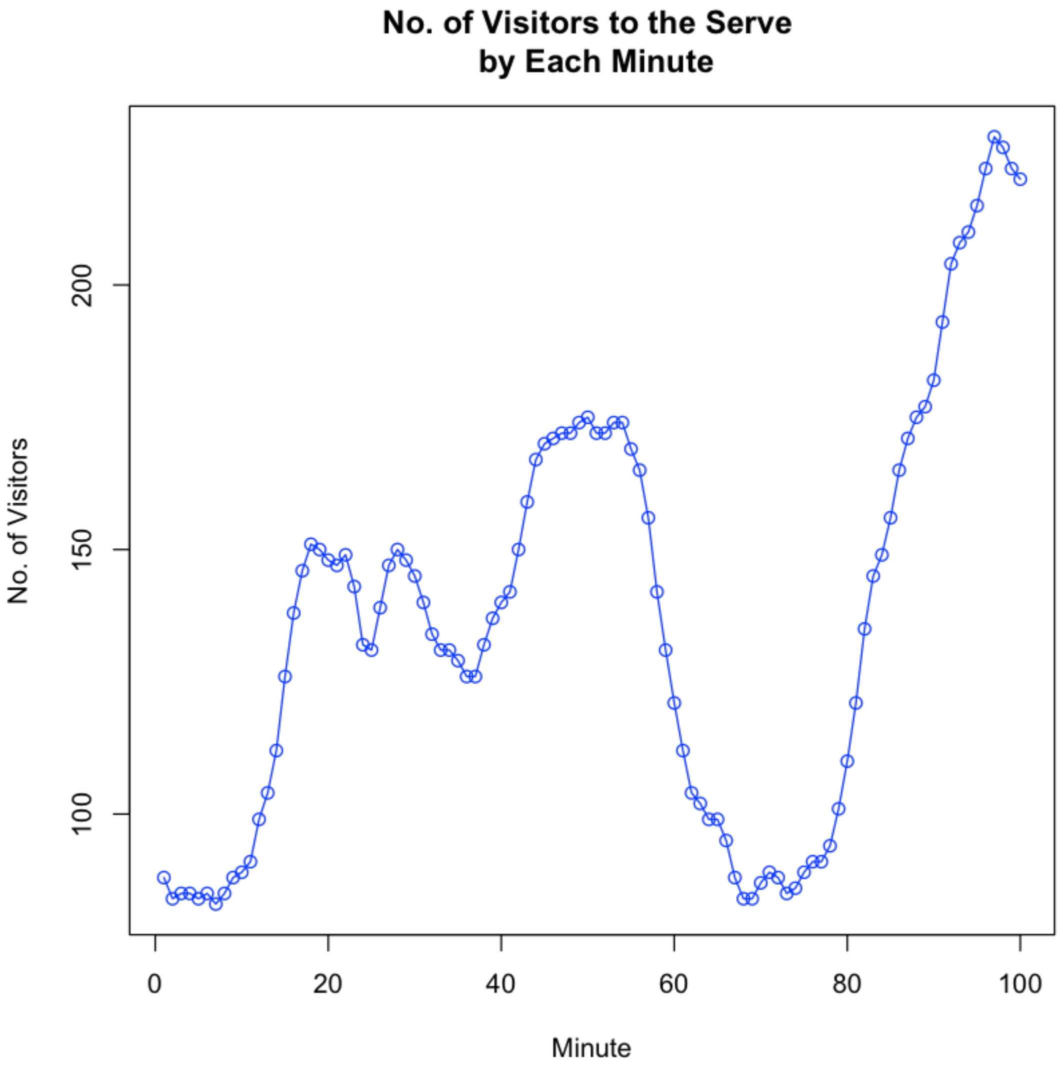



MTCARS Data
mpg
disp
hp
wt
100
200
300
400
2
3
4
5
30
25
20
15
10
400
300
200
100
250
150
50
5
4
3
2
10
15
20
25
30
50
150
250



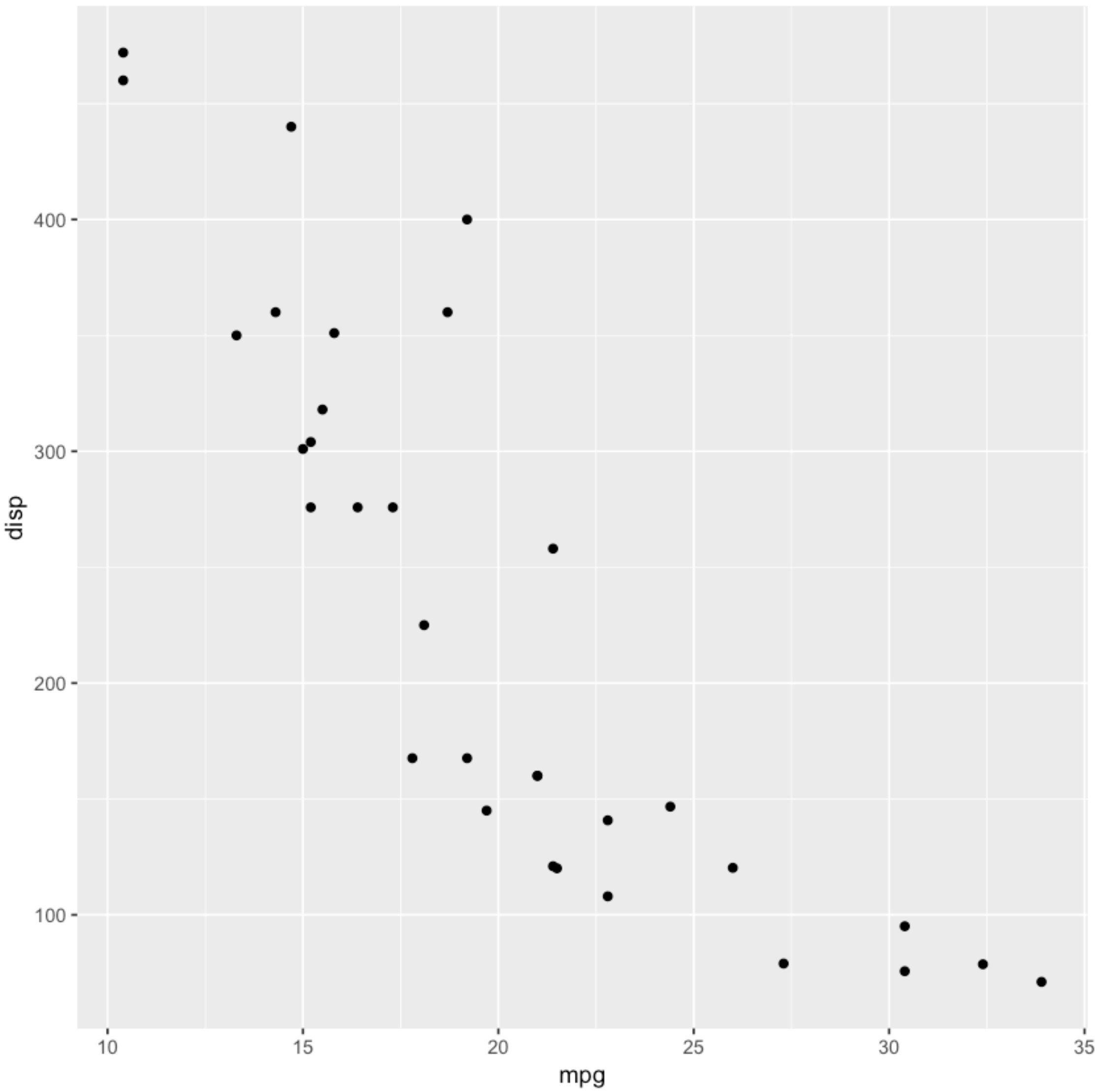
```
In [43]: head(WWWusage)
         plot(WWWusage,type="o",col="blue",xlab="Minute",ylab="No. of Visitors",main="No. of Vi
```

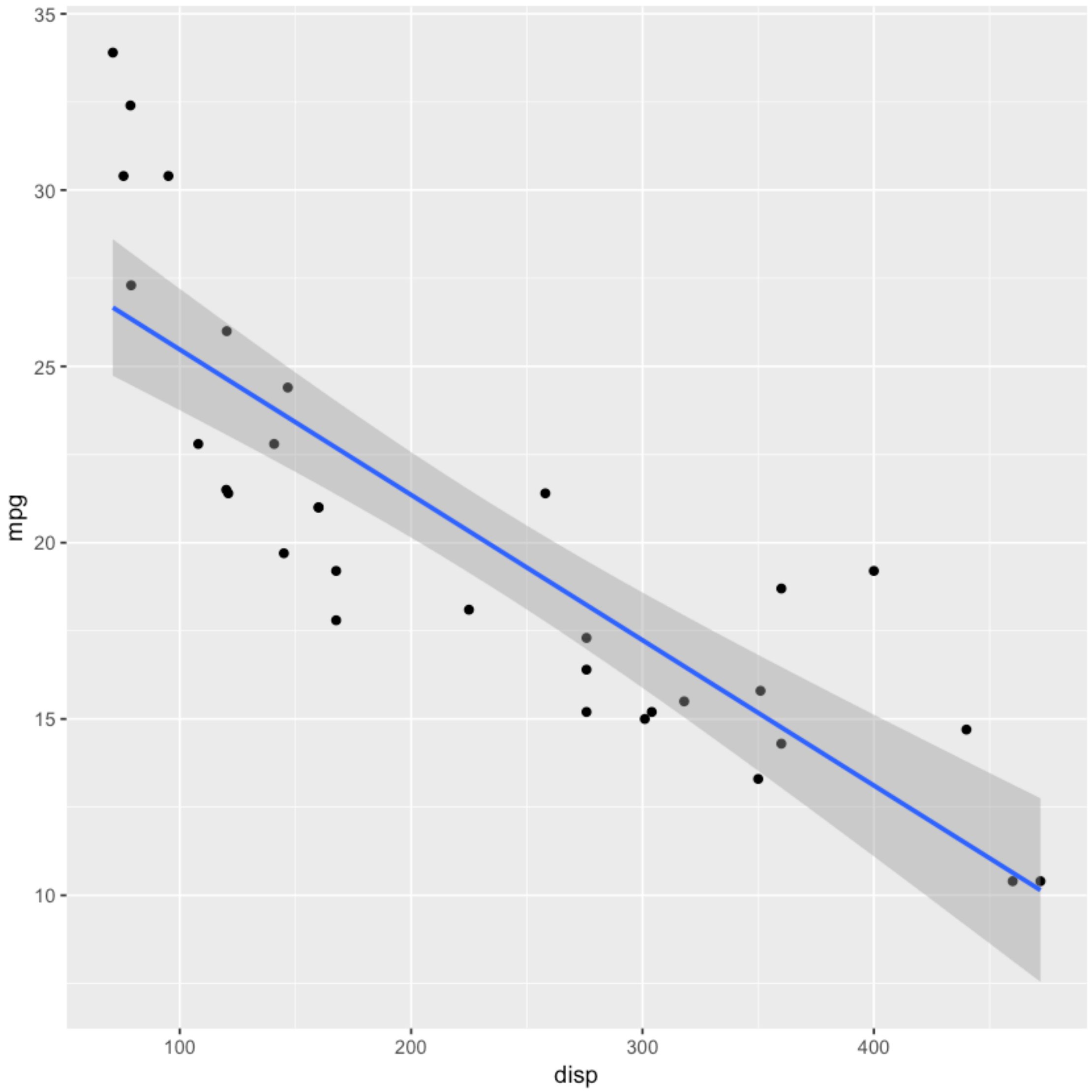
1. 88 2. 84 3. 85 4. 85 5. 84 6. 85

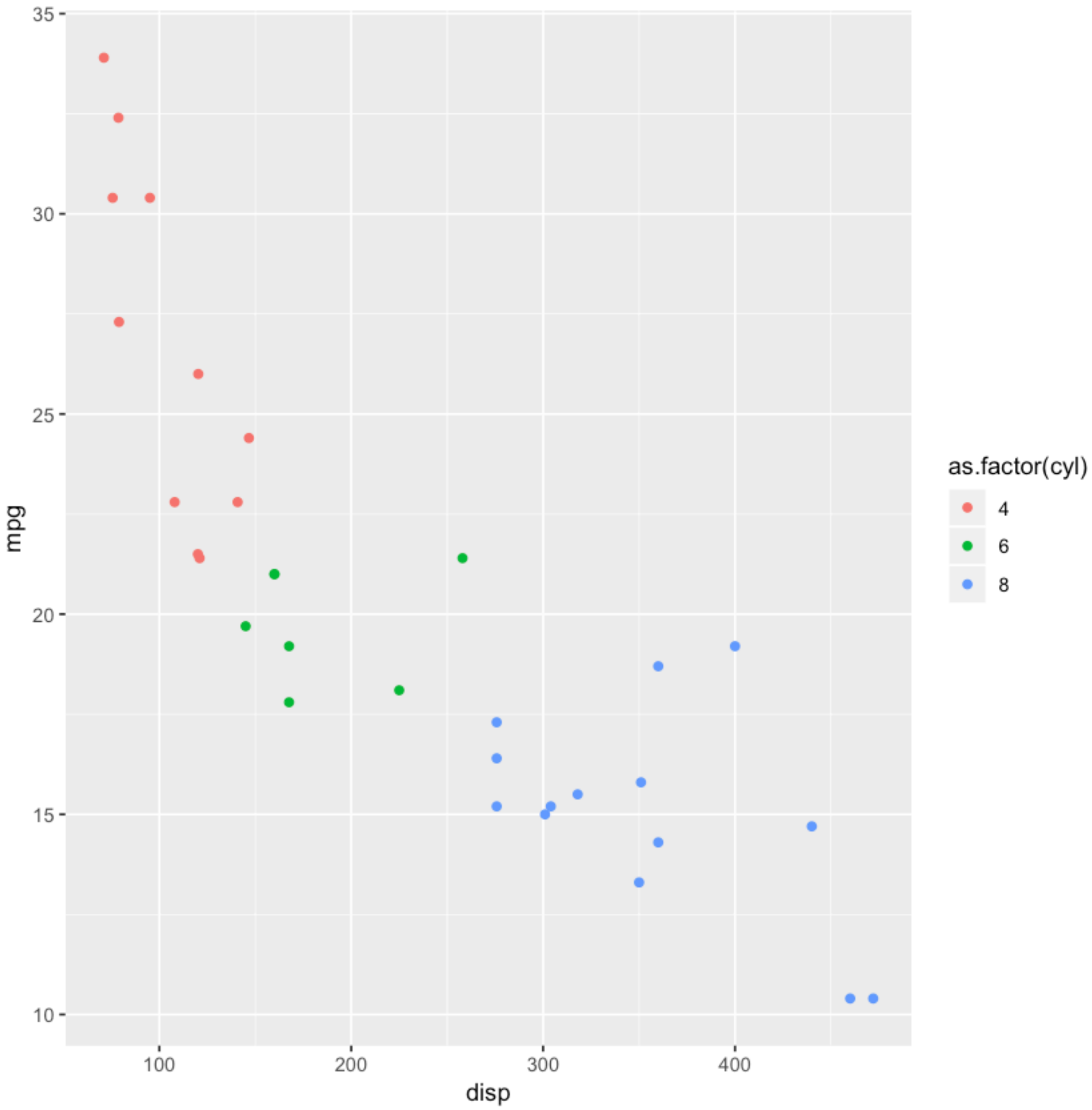


0.10 Problem 8

```
In [44]: library(ggplot2)
         ggplot(mtcars, aes(x=mpg, y=disp)) + geom_point()

         ggplot(mtcars, mapping = aes(x = disp, y = mpg)) + geom_point() +
           stat_smooth(method = 'lm')

         ggplot(mtcars, mapping = aes(x = disp, y = mpg, color = as.factor(cyl))) + geom_point
```


400
300
200
100
disp
10
15
20
25
30
35
mpg


35
30
25
20
15
10
mpg
100
200
300
400
disp


35
30
25
20
15
10
mpg
100
200
300
400
disp
as.factor(cyl)
● 4
● 6
● 8