

Experiment No. 3

Title: Loop functions, debugging tools, Mathematical Functions in R, Exploratory data analysis in R

List of programs:

1. Find the sum of every column in matrix using apply()

Code:

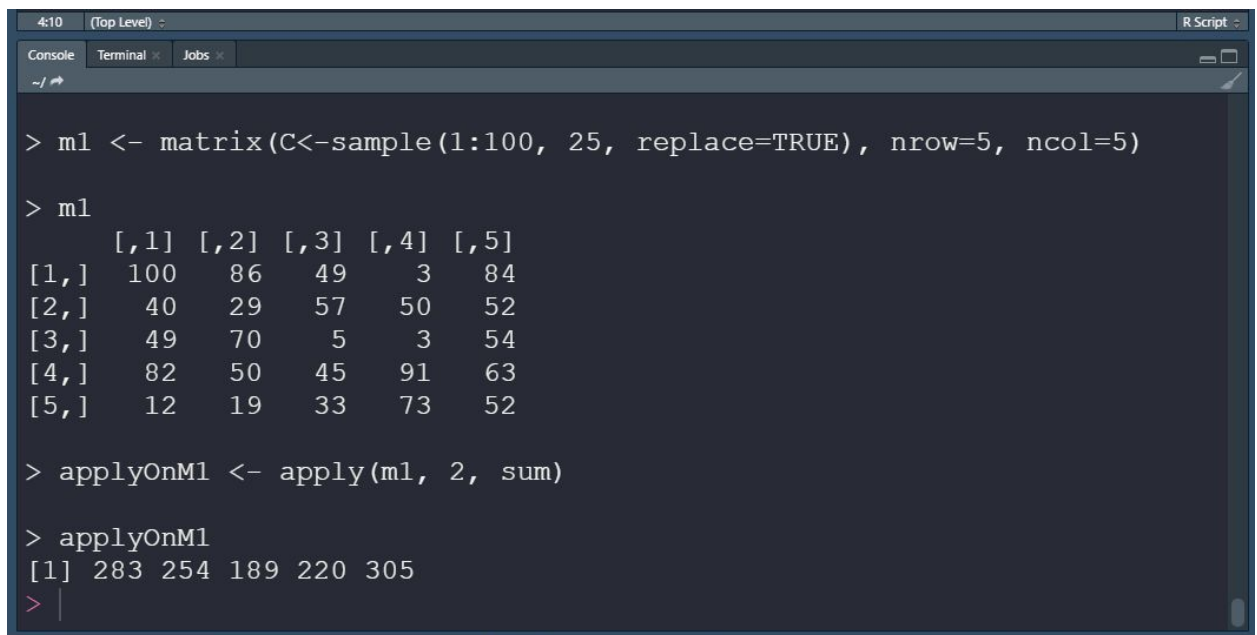
```
m <- matrix(C<-sample(1:100, 25, replace=TRUE), nrow=5, ncol=5)
```

```
m
```

```
applyOnM1 <- apply(m, 2, sum)
```

```
applyOnM1
```

Screenshot:



```
4:10 (Top Level) - R Script
Console Terminal Jobs
~/
> m1 <- matrix(C<-sample(1:100, 25, replace=TRUE), nrow=5, ncol=5)
> m1
      [,1] [,2] [,3] [,4] [,5]
[1,]  100   86   49    3   84
[2,]   40   29   57   50   52
[3,]   49   70    5    3   54
[4,]   82   50   45   91   63
[5,]   12   19   33   73   52
> applyOnM1 <- apply(m1, 2, sum)
> applyOnM1
[1] 283 254 189 220 305
> |
```

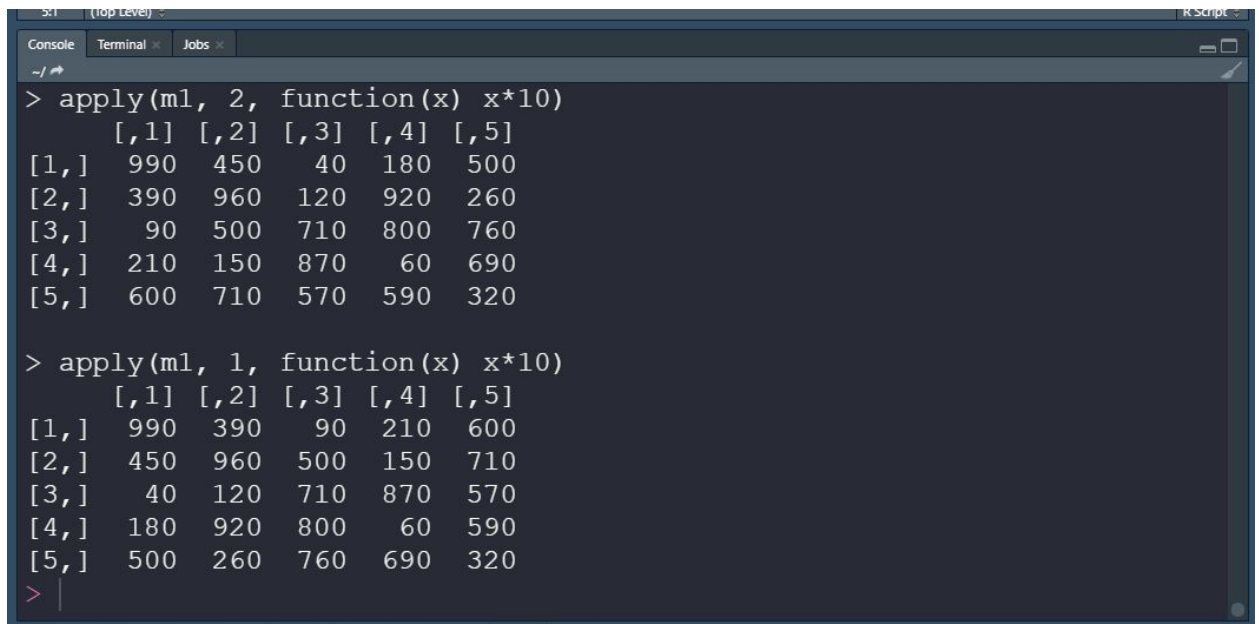
2. Create user-defined function multiply no by 10, and use to multiply each no of a matrix in apply function

Code:

```
apply(m1, 2, function(x) x*10)
```

```
apply(m1, 1, function(x) x*10)
```

Screenshot:



```
> apply(m1, 2, function(x) x*10)
      [,1] [,2] [,3] [,4] [,5]
[1,]  990  450   40  180  500
[2,]  390  960  120  920  260
[3,]   90  500  710  800  760
[4,]  210  150  870   60  690
[5,]  600  710  570  590  320

> apply(m1, 1, function(x) x*10)
      [,1] [,2] [,3] [,4] [,5]
[1,]  990  390   90  210  600
[2,]  450  960  500  150  710
[3,]   40  120  710  870  570
[4,]  180  920  800   60  590
[5,]  500  260  760  690  320
>
```

3. Measure the minimum speed and stopping distances of cars from the cars dataset using `sapply()`. [Use cars dataset]

Code:

```
dt <- cars
```

```
lmn_cars <- lapply(dt, min)
```

```
smn_cars <- sapply(dt, min)
```

```
lmn_cars
```

```
smn_cars
```

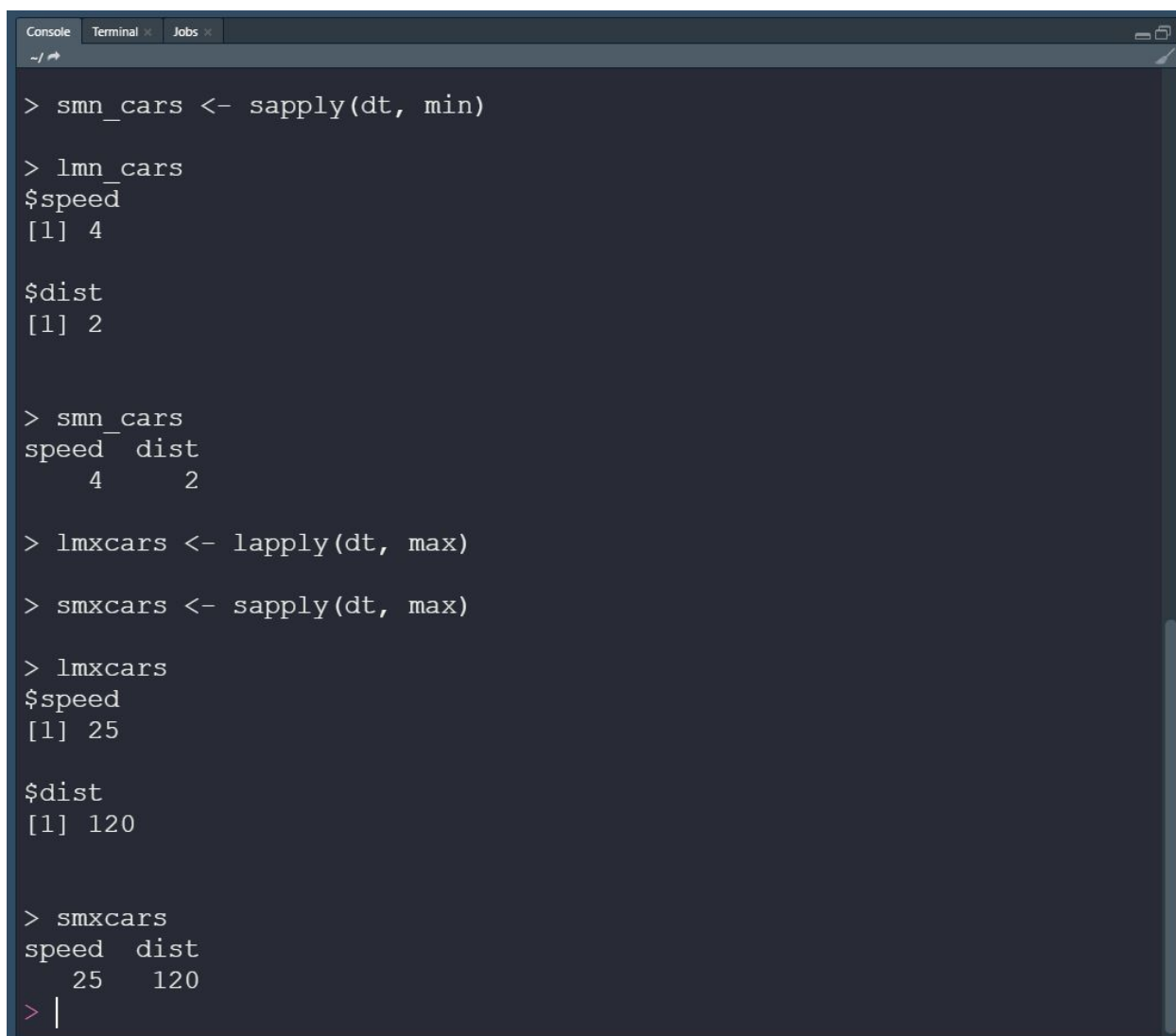
```
lmxcars <- lapply(dt, max)
```

```
smxcars <- sapply(dt, max)
```

```
lmxcars
```

```
smxcars
```

Screenshot:

A screenshot of a R console window with a dark background. The window has tabs for 'Console', 'Terminal', and 'Jobs'. The console shows the following R code and its output:

```
> smn_cars <- sapply(dt, min)

> lmn_cars
$speed
[1] 4

$dists
[1] 2

> smn_cars
speed dists
  4      2

> lmxcars <- lapply(dt, max)

> smxcars <- sapply(dt, max)

> lmxcars
$speed
[1] 25

$dists
[1] 120

> smxcars
speed dists
  25    120

> |
```

4. Generate dataset for 100 patients and find out average no of patients diabetic or with heart disease using `tapply()` function

Code:

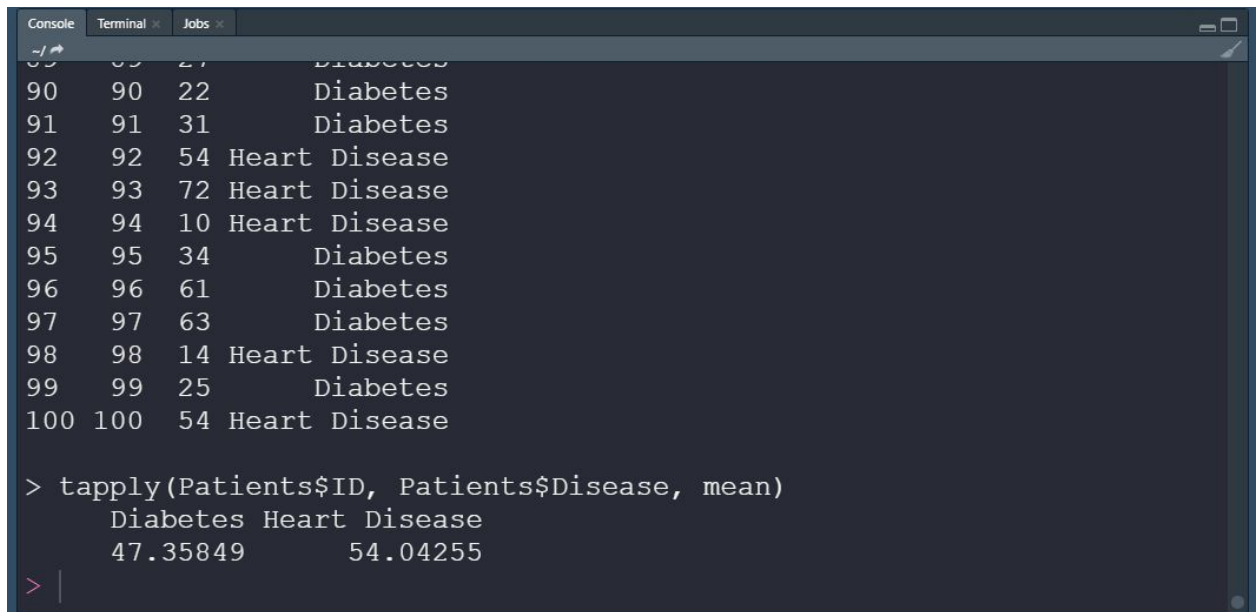
```
Disease <- c("Diabetes", "Heart Disease")
```

```
Patients <- data.frame("ID" = 1:100, "Age" = c(sample(10:80, 100,  
replace=TRUE)), "Disease" = c(sample(Disease, 100, replace=TRUE)))
```

Patients

```
tapply(Patients$ID, Patients$Disease, mean)
```

Screenshot:



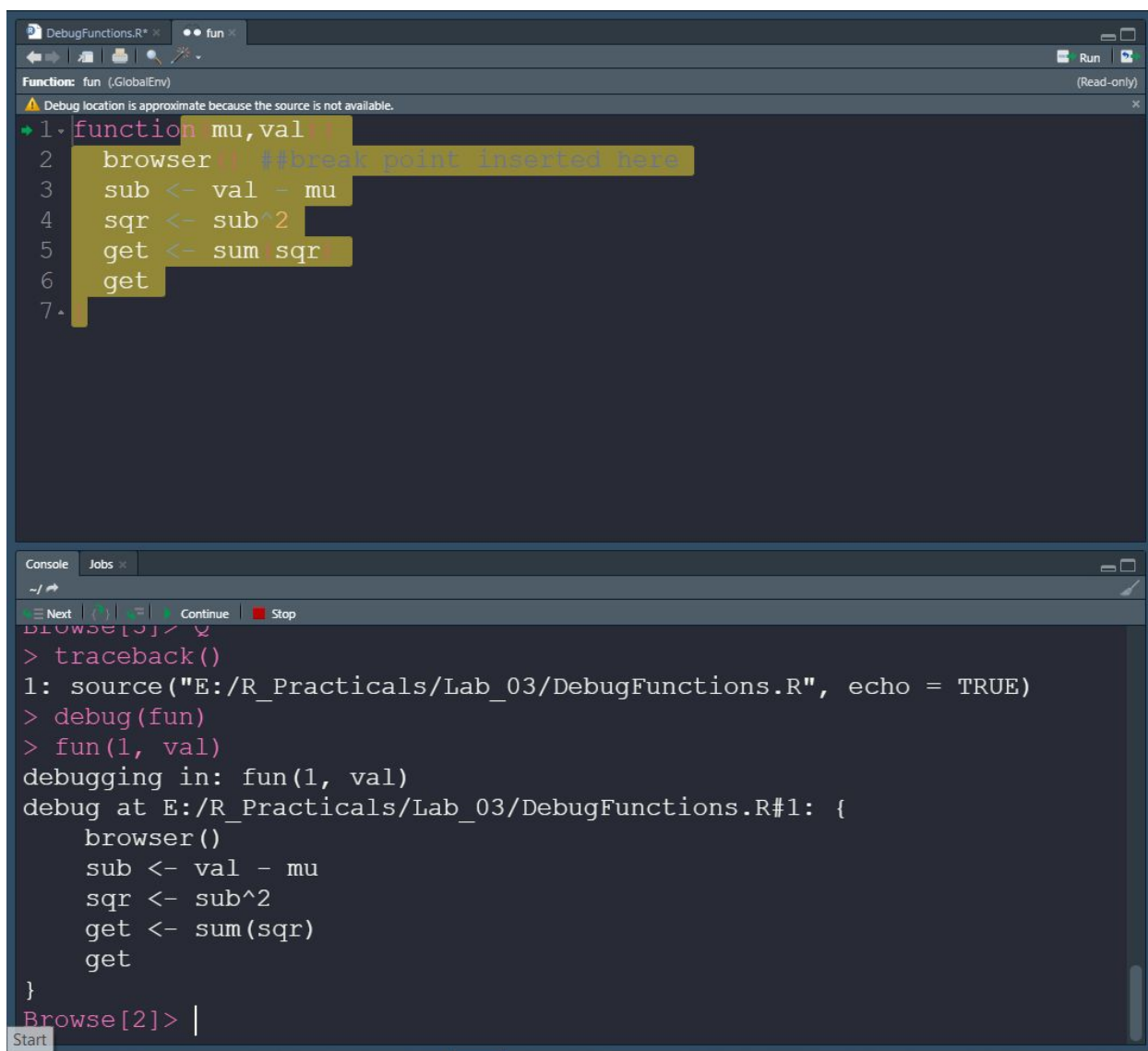
The screenshot shows an R console window with the following content:

```
Console Terminal x Jobs x  
90 90 22 Diabetes  
91 91 31 Diabetes  
92 92 54 Heart Disease  
93 93 72 Heart Disease  
94 94 10 Heart Disease  
95 95 34 Diabetes  
96 96 61 Diabetes  
97 97 63 Diabetes  
98 98 14 Heart Disease  
99 99 25 Diabetes  
100 100 54 Heart Disease  
  
> tapply(Patients$ID, Patients$Disease, mean)  
Diabetes Heart Disease  
47.35849 54.04255  
> |
```

5. Make use of debugging tools in R

Code:

Screenshot:



DebugFunctions.R* x fun x

Function: fun (GlobalEnv) (Read-only)

⚠ Debug location is approximate because the source is not available.

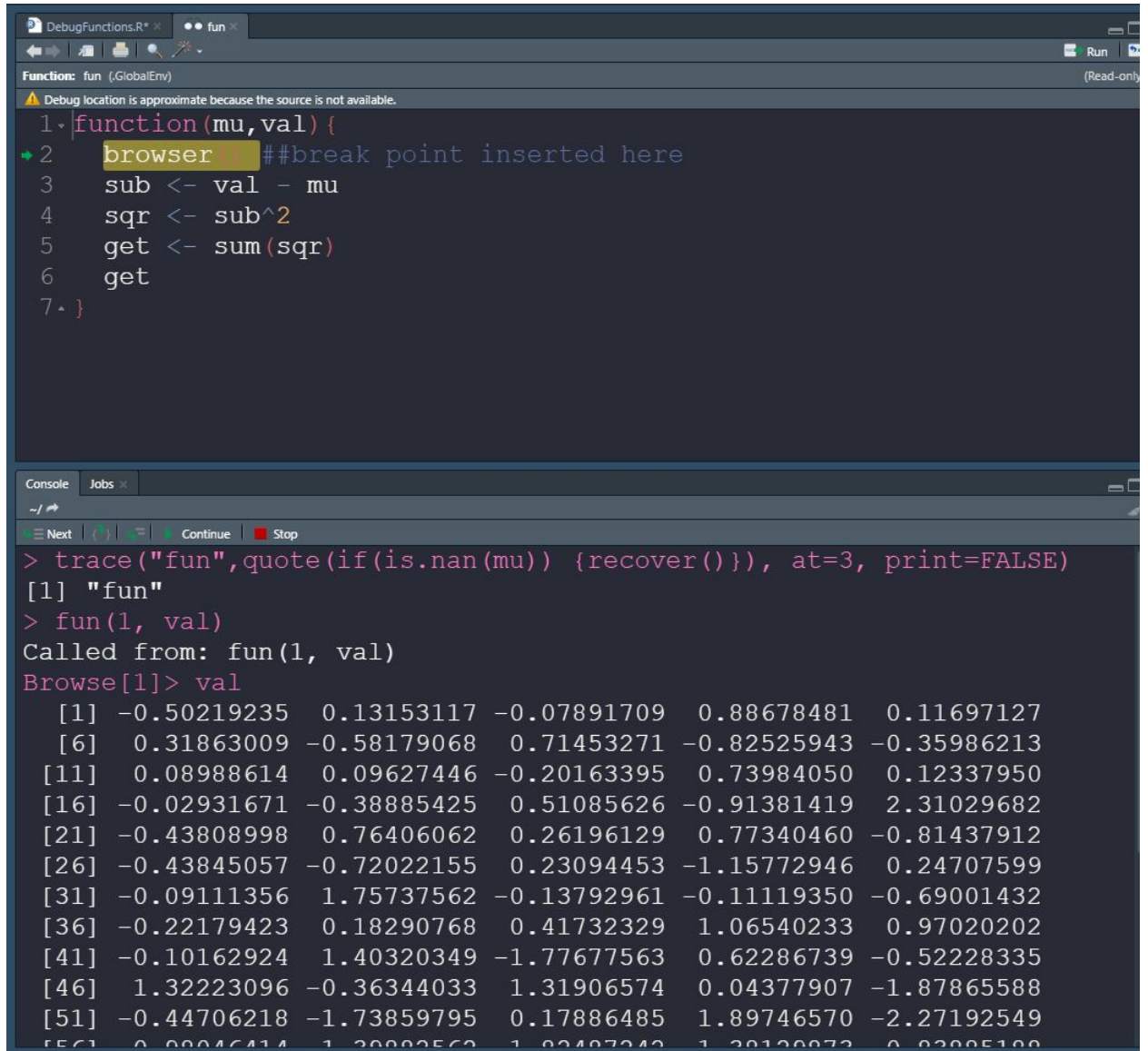
```
1- function mu, val {}  
2-   browser() ##break point inserted here  
3-   sub <- val - mu  
4-   sqr <- sub^2  
5-   get <- sum sqr  
6-   get  
7-
```

Console Jobs x

Next Previous Continue Stop

```
Browse[3]>  
> traceback()  
1: source("E:/R_Practicals/Lab_03/DebugFunctions.R", echo = TRUE)  
> debug(fun)  
> fun(1, val)  
debugging in: fun(1, val)  
debug at E:/R_Practicals/Lab_03/DebugFunctions.R#1: {  
  browser()  
  sub <- val - mu  
  sqr <- sub^2  
  get <- sum(sqr)  
  get  
}  
Browse[2]> |
```

Start



DebugFunctions.R* x fun x

Function: fun (GlobalEnv) (Read-only)

⚠ Debug location is approximate because the source is not available.

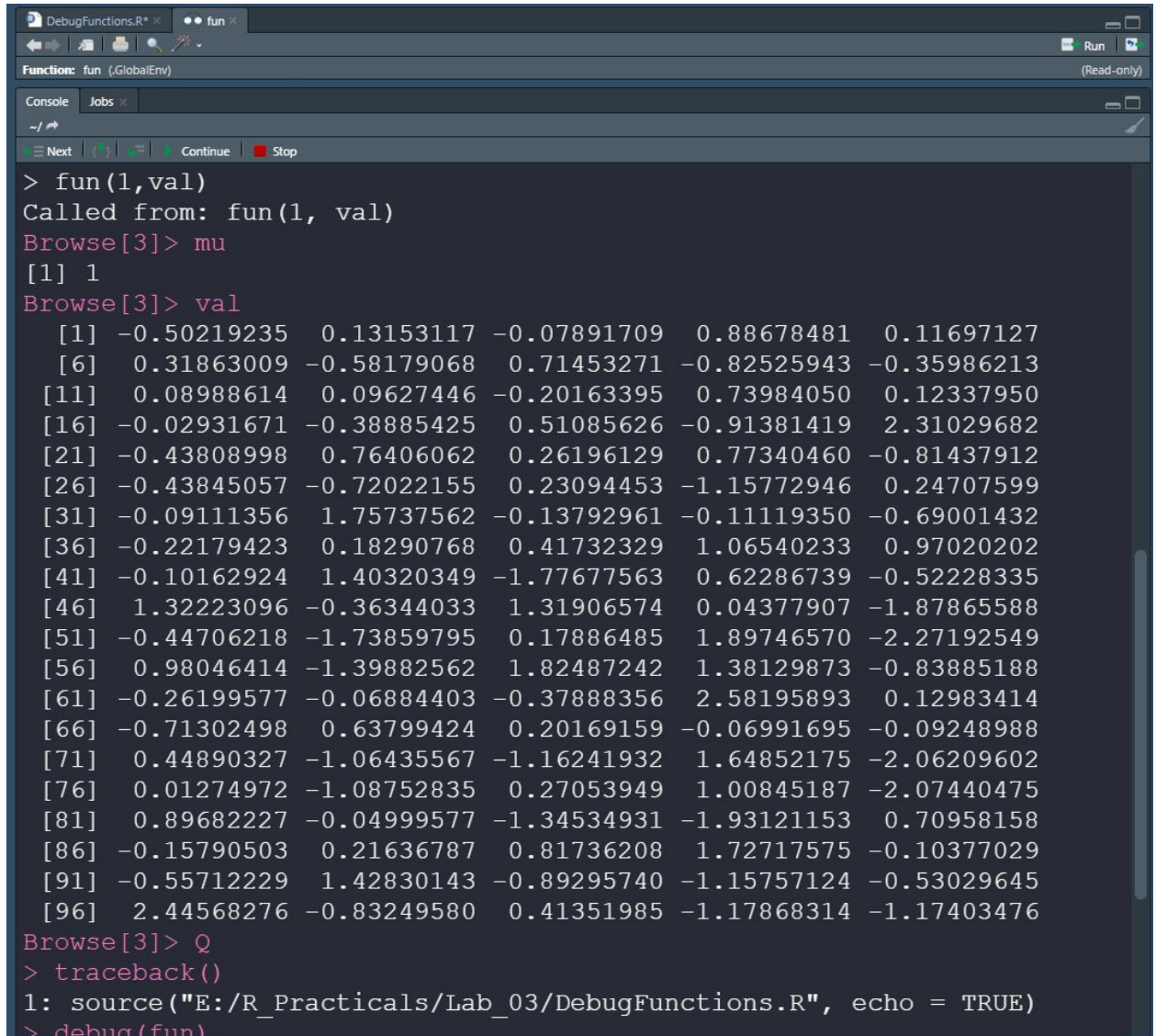
```
1. function(mu, val) {  
2.   browser() ##break point inserted here  
3.   sub <- val - mu  
4.   sqr <- sub^2  
5.   get <- sum(sqr)  
6.   get  
7. }
```

Console Jobs x

~/ ➡

Next | | | Continue | Stop

```
> trace("fun", quote(if(is.nan(mu)) {recover()}), at=3, print=FALSE)  
[1] "fun"  
> fun(1, val)  
Called from: fun(1, val)  
Browse[1]> val  
[1] -0.50219235 0.13153117 -0.07891709 0.88678481 0.11697127  
[6] 0.31863009 -0.58179068 0.71453271 -0.82525943 -0.35986213  
[11] 0.08988614 0.09627446 -0.20163395 0.73984050 0.12337950  
[16] -0.02931671 -0.38885425 0.51085626 -0.91381419 2.31029682  
[21] -0.43808998 0.76406062 0.26196129 0.77340460 -0.81437912  
[26] -0.43845057 -0.72022155 0.23094453 -1.15772946 0.24707599  
[31] -0.09111356 1.75737562 -0.13792961 -0.11119350 -0.69001432  
[36] -0.22179423 0.18290768 0.41732329 1.06540233 0.97020202  
[41] -0.10162924 1.40320349 -1.77677563 0.62286739 -0.52228335  
[46] 1.32223096 -0.36344033 1.31906574 0.04377907 -1.87865588  
[51] -0.44706218 -1.73859795 0.17886485 1.89746570 -2.27192549  
[56] 0.00046414 1.30002560 1.02407240 1.30130072 0.00005100
```



```

DebugFunctions.R* x  • fun x
Function: fun (.GlobalEnv) (Read-only)
Console Jobs x
~/
Next | Previous | Continue | Stop

> fun(1, val)
Called from: fun(1, val)
Browse[3]> mu
[1] 1
Browse[3]> val
[1] -0.50219235  0.13153117 -0.07891709  0.88678481  0.11697127
[6]  0.31863009 -0.58179068  0.71453271 -0.82525943 -0.35986213
[11]  0.08988614  0.09627446 -0.20163395  0.73984050  0.12337950
[16] -0.02931671 -0.38885425  0.51085626 -0.91381419  2.31029682
[21] -0.43808998  0.76406062  0.26196129  0.77340460 -0.81437912
[26] -0.43845057 -0.72022155  0.23094453 -1.15772946  0.24707599
[31] -0.09111356  1.75737562 -0.13792961 -0.11119350 -0.69001432
[36] -0.22179423  0.18290768  0.41732329  1.06540233  0.97020202
[41] -0.10162924  1.40320349 -1.77677563  0.62286739 -0.52228335
[46]  1.32223096 -0.36344033  1.31906574  0.04377907 -1.87865588
[51] -0.44706218 -1.73859795  0.17886485  1.89746570 -2.27192549
[56]  0.98046414 -1.39882562  1.82487242  1.38129873 -0.83885188
[61] -0.26199577 -0.06884403 -0.37888356  2.58195893  0.12983414
[66] -0.71302498  0.63799424  0.20169159 -0.06991695 -0.09248988
[71]  0.44890327 -1.06435567 -1.16241932  1.64852175 -2.06209602
[76]  0.01274972 -1.08752835  0.27053949  1.00845187 -2.07440475
[81]  0.89682227 -0.04999577 -1.34534931 -1.93121153  0.70958158
[86] -0.15790503  0.21636787  0.81736208  1.72717575 -0.10377029
[91] -0.55712229  1.42830143 -0.89295740 -1.15757124 -0.53029645
[96]  2.44568276 -0.83249580  0.41351985 -1.17868314 -1.17403476
Browse[3]> Q
> traceback()
1: source("E:/R_Practicals/Lab_03/DebugFunctions.R", echo = TRUE)
> debug(fun)

```

6. Create different plots using ggplot2

Use the dataset mtcars and create the following plots

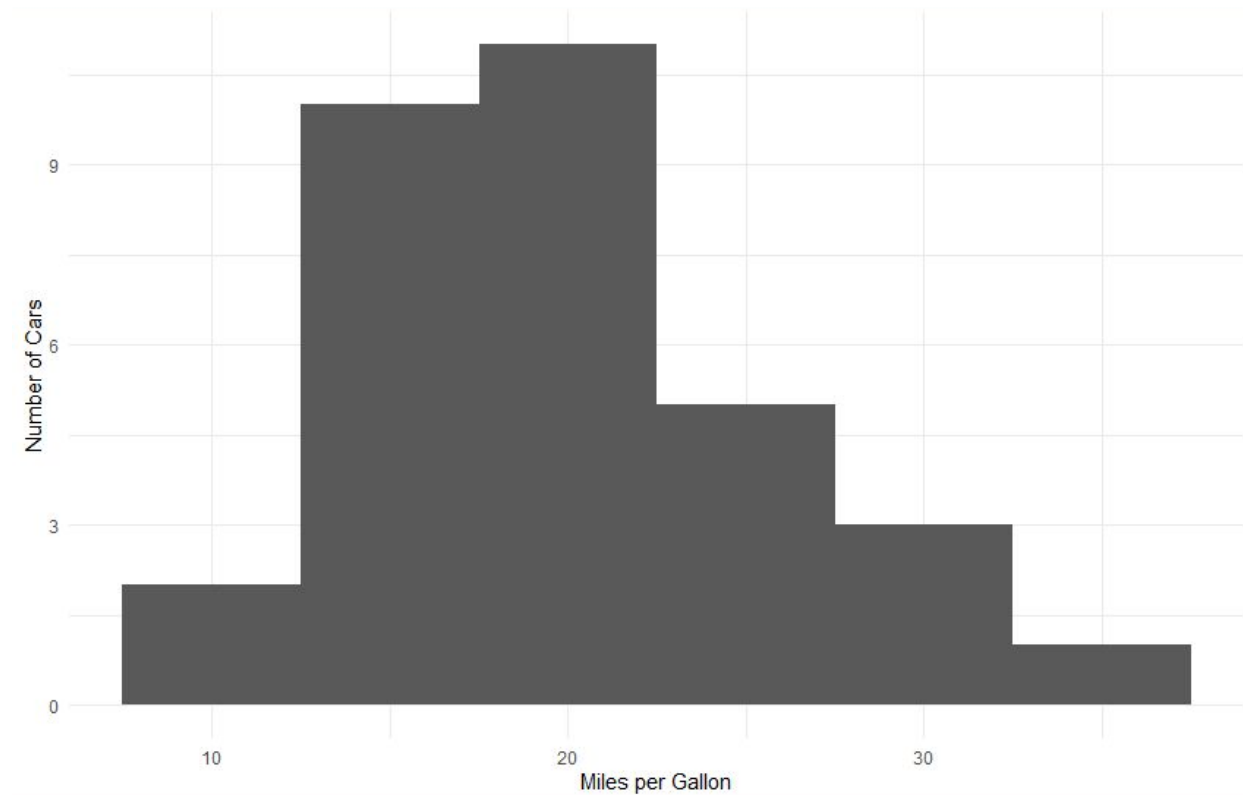
a. Histograms

Code:

```
my_histogram <- ggplot(mtcars, aes(x=mpg)) +
  geom_histogram(binwidth=5)
```

```
my_histogram + xlab('Miles per Gallon') + ylab('Number of Cars') +  
theme_minimal()
```

Screenshot:

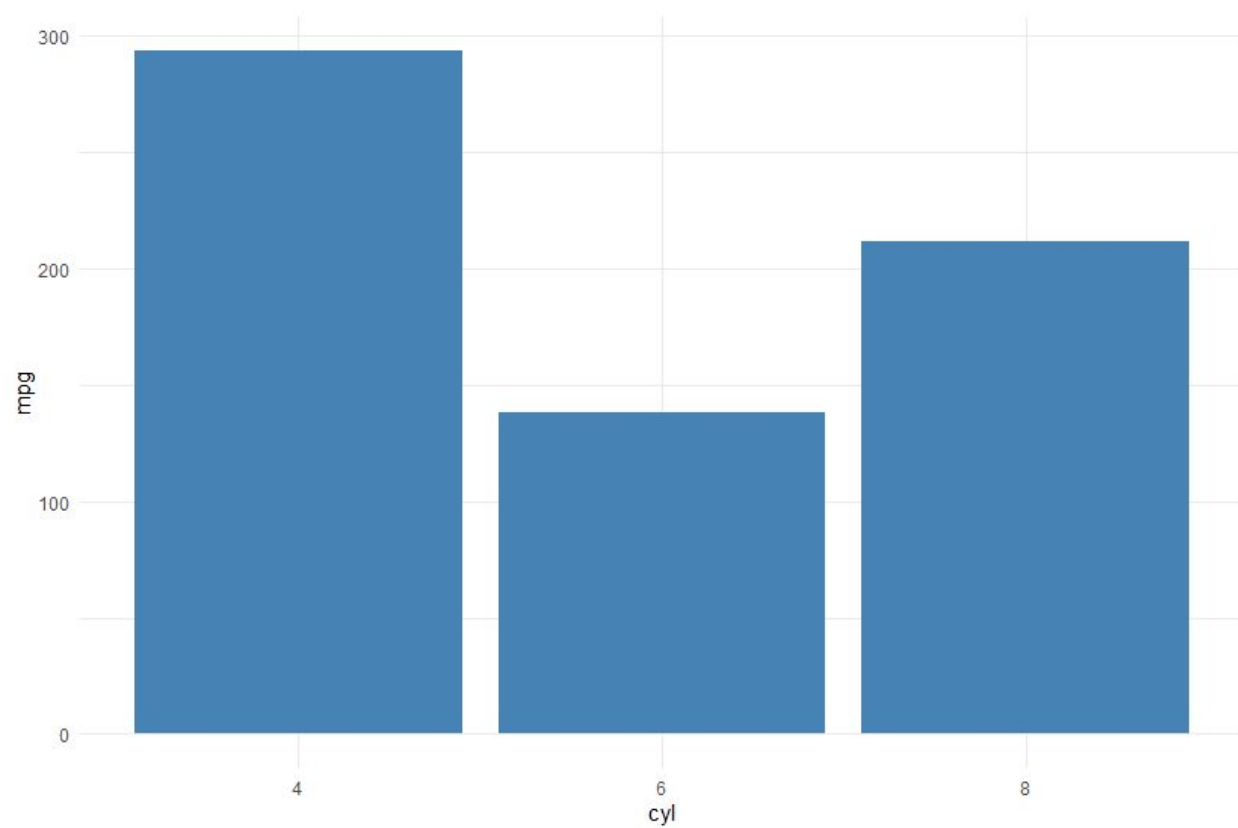


b. Bar and Column Chart

Code:

```
my_barplot<-ggplot(data=mtcars, aes(x=mpg, y=cyl)) +  
  geom_bar(stat="identity", fill="steelblue")+  
  theme_minimal()  
  
my_barplot + coord_flip()
```

Screenshot:



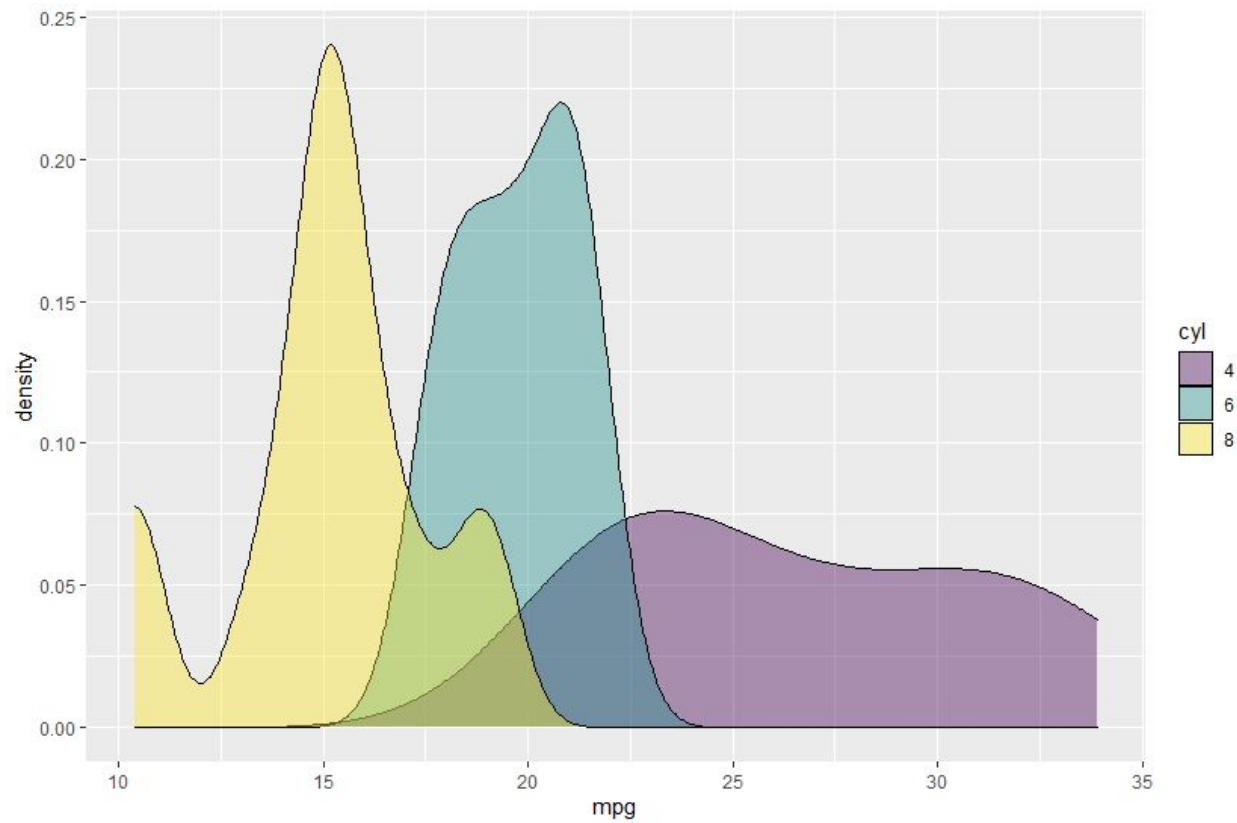
c. Density Chart

Code:

```
ggplot(data=mtcars, aes(x=mpg, fill=cyl)) +  
  geom_density(alpha=0.4)
```

```
p+geom_vline(data=mtcars, aes(xintercept=mpg.mean, color=cyl),  
             linetype="dashed")
```

Screenshot:



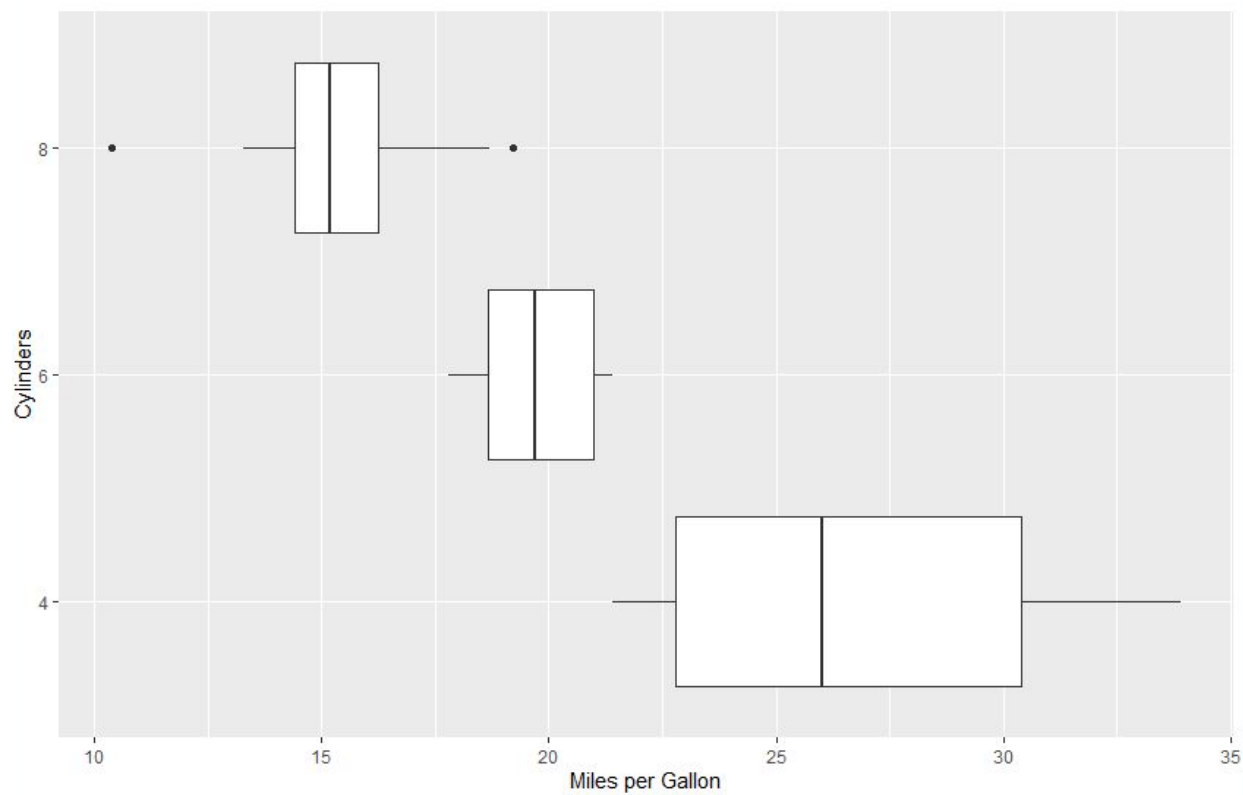
d. Box Plot

Code:

```
my_boxplot <- ggplot(mtcars,aes(x=cyl,y=mpg)) + geom_boxplot() +  
xlab('Cylinders') + ylab('Miles per Gallon')
```

```
my_boxplot + coord_flip()
```

Screenshot:



e. Scatter Plot

Code:

```
my_scatterplot <- ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point()
my_scatterplot + xlab('Weight (x 1000lbs)') + ylab('Miles per Gallon')
```

Screenshot:

