Exercise Answers: Introduction to R Workshop

Section A

1. Clear your workspace (remove all objects from your directory)

```
> rm(list=ls())
> ls()
character(0)
>
```

2. What is (1198/2) - 63 + 27

```
> (1198/2) - 63 + 27
[1] 563
```

3. Set *a* equal to 7 and *b* equal to 17 and *c* equal to their product

```
> a <- 7
> b <- 17
> c <- a * b
> c
[1] 119
>
```

4. Type the following into R

```
xx <- c(29,3,6,11,0,41,101)
```

```
> xx <- c(29,3,6,11,0,41,101)
> xx
[1]  29   3   6  11   0  41 101
>
```

7. Use the ls() command to view the objects in your workspace.

```
> ls()
[1] "a"  "b"  "c"  "xx"
>
```

8. Quit R, saving your workspace.

```
q()
```

9. Restart R and reload your workspace. Use the ls() command to make sure everything is still there. Check that the values of your objects haven't changed.

```
> ls()
[1] "a"  "b"  "c"  "xx"
> a
[1] 7
> b
[1] 17
> c
[1] 119
> xx
[1]  29   3   6  11   0  41 101
>
```

Section B

1. Print out the 4th element of xx.

```
> xx[4]

[1] 11
```

2. What is the length of xx?

```
> length(xx)
[1] 7
>
```

3. Print out the first three elements of xx

```
> xx[1:3]
[1] 29  3  6
>
```

4. Print out the 4th and 7th elements of xx

```
> xx[c(4,7)]
[1]  11 101
>
```

5. Create a vector of length 20 of the even integers starting with 34.

```
> help(seq)
> a.vector <- seq(from=34,by=2,length.out=20)
> a.vector
 [1] 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72
> length(a.vector)
[1] 20
>
```

6. (Optional) Set the vector "xvals" equal to the numbers from -2.5 to +2.5 in increments of .02. How many elements are in this vector?

```
> xvals <- seq(-2.5,2.5,.02)
> xvals
  [1] -2.50 -2.48 -2.46 -2.44 -2.42 -2.40 -2.38 -2.36 -2.34 -2.32 -2.30 -2.28
 [13] -2.26 -2.24 -2.22 -2.20 -2.18 -2.16 -2.14 -2.12 -2.10 -2.08 -2.06 -2.04
 [25] -2.02 -2.00 -1.98 -1.96 -1.94 -1.92 -1.90 -1.88 -1.86 -1.84 -1.82 -1.80
 [37] -1.78 -1.76 -1.74 -1.72 -1.70 -1.68 -1.66 -1.64 -1.62 -1.60 -1.58 -1.56
 [49] -1.54 -1.52 -1.50 -1.48 -1.46 -1.44 -1.42 -1.40 -1.38 -1.36 -1.34 -1.32
 [61] -1.30 -1.28 -1.26 -1.24 -1.22 -1.20 -1.18 -1.16 -1.14 -1.12 -1.10 -1.08
 [73] -1.06 -1.04 -1.02 -1.00 -0.98 -0.96 -0.94 -0.92 -0.90 -0.88 -0.86 -0.84
```

```
 [85] -0.82 -0.80 -0.78 -0.76 -0.74 -0.72 -0.70 -0.68 -0.66 -0.64 -0.62 -0.60
 [97] -0.58 -0.56 -0.54 -0.52 -0.50 -0.48 -0.46 -0.44 -0.42 -0.40 -0.38 -0.36
[109] -0.34 -0.32 -0.30 -0.28 -0.26 -0.24 -0.22 -0.20 -0.18 -0.16 -0.14 -0.12
[121] -0.10 -0.08 -0.06 -0.04 -0.02  0.00  0.02  0.04  0.06  0.08  0.10  0.12
[133]  0.14  0.16  0.18  0.20  0.22  0.24  0.26  0.28  0.30  0.32  0.34  0.36
[145]  0.38  0.40  0.42  0.44  0.46  0.48  0.50  0.52  0.54  0.56  0.58  0.60
[157]  0.62  0.64  0.66  0.68  0.70  0.72  0.74  0.76  0.78  0.80  0.82  0.84
[169]  0.86  0.88  0.90  0.92  0.94  0.96  0.98  1.00  1.02  1.04  1.06  1.08
[181]  1.10  1.12  1.14  1.16  1.18  1.20  1.22  1.24  1.26  1.28  1.30  1.32
[193]  1.34  1.36  1.38  1.40  1.42  1.44  1.46  1.48  1.50  1.52  1.54  1.56
[205]  1.58  1.60  1.62  1.64  1.66  1.68  1.70  1.72  1.74  1.76  1.78  1.80
[217]  1.82  1.84  1.86  1.88  1.90  1.92  1.94  1.96  1.98  2.00  2.02  2.04
[229]  2.06  2.08  2.10  2.12  2.14  2.16  2.18  2.20  2.22  2.24  2.26  2.28
[241]  2.30  2.32  2.34  2.36  2.38  2.40  2.42  2.44  2.46  2.48  2.50
> length(xvals)
[1] 251
>
```

7. (Optional) Create a vector with the sequence 1,3,5,7,1,3,5,7,1,3,5,7 (hint use `rep` function).

Method 1

```
> temp <- c(1,3,5,7)
> mary <- rep(temp,3)
> mary
 [1] 1 3 5 7 1 3 5 7 1 3 5 7
>
```

Method 2

```
> mary <- rep(c(1,3,5,7),3)
> mary
 [1] 1 3 5 7 1 3 5 7 1 3 5 7
>
```

Method 3

```
> mary <- rep(seq(1,7,2),3)
> mary
 [1] 1 3 5 7 1 3 5 7 1 3 5 7
```

Section C

1. Type the following commands into R:

```
set.seed(7)

test <- sample(1:10,size=40,replace=TRUE)
```

```
> set.seed(7)
> test <- sample(1:10,size=40,replace=TRUE)
> test
 [1] 10  4  2  1  3  8  4 10  2  5  2  3  8  1  5  1  6  1 10  4  7  3 10 10 10
[26]  1  7  5 10  4  7  3  2  2  4  9  5  8  9  5
>
```

2. How many elements in test?

```
> length(test)
[1] 40
>
```

3. What is the value of the first element of test? the last element?

```
First element
> test[1]
[1] 10

Last element method 1
> test[40]
[1] 5

Last element method 2
> test[length(test)]
[1] 5
>
```

4. What is the mean, median, standard deviation, min, max, of test?

   (refer to "List of helpful R functions.doc")

```
> mean(test)
[1] 5.275
> median(test)
[1] 5
> sd(test)
[1] 3.162176
> min(test)
[1] 1
> max(test)
[1] 10
>
```

5. (Optional) What is the mean of the first ten elements? the last ten elements?

```
First ten elements
> mean(test[1:10])
[1] 4.9

Last ten elements method 1
> mean(test[31:40])
[1] 5.4

Last ten elements method 2
> mean( test[ (length(test)-9):length(test) ] )
[1] 5.4
>
```

6. How many 1's in test?

```
> sum(test==1)
[1] 5
>
```

7. How many elements have values less than 4?

```
> sum(test<4)
[1] 14
>
```

8. (Optional) How many elements are between 6 and 9, inclusive?

```
> sum( (test>=6) & (test<=9) )
[1] 9
```

```
>
```

9. (Optional) How many elements at the extremes (less than 2 or greater than 8)?

```
> sum( (test < 2) | (test > 8) )
[1] 14
>
```

10. (Optional) What is the mean of the odd elements of test?

```
> seq(from=1,by=2,to=length(test))
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
> mean( test[ seq(from=1,by=2,to=length(test)) ] )
[1] 6.15
>
```

Section D

1. Type in the following command to R

```
mat <- matrix(seq(21,71,10),nrow=3)
```

```
> mat <- matrix(seq(21,71,10),nrow=3)
> mat
     [,1] [,2]
[1,]   21   51
[2,]   31   61
[3,]   41   71
>
```

a. print out the value in the 2nd row, 2nd column

```
> mat[2,2]
[1] 61
>
```

b. print out the last row

```
> mat[3,]
[1] 41 71
> nrow(mat)
[1] 3
> mat[nrow(mat),]
[1] 41 71
>
```

c. print out the first column

```
> mat[,1]
[1] 21 31 41
>
```

3. Use help to figure out what the function "sample" does.

```
> help(sample)
```

4. Use "sample" to randomly order the integers from 1 to 20 and store in "twenty"

```
> twenty <- sample(1:20,20)
> twenty
 [1] 16  8 14 19 15  5  2 11 13 12  6  7  9 20 17 18  1  4  3 10
>
```

5. (Optional) Use help to look up “sort”, “order”, and “rank”. Use these functions on “twenty” to figure out what they do.

```
> sort(twenty)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

> order(twenty)
 [1] 17  7 19 18  6 11 12  2 13 20  8 10  9  3  5  1 15 16  4 14
### the 17th element is 1, the 7th element is 2,  …,
###the 14th element is 20

> rank(twenty)
 [1] 16  8 14 19 15  5  2 11 13 12  6  7  9 20 17 18  1  4  3 10
>
```

6. (Optional) Use help to figure out what “set.seed” does and verify that you understand what it does.

```
### everytime you ask for random sample, you get something different
### set.seed allows you to get back the same sample
> set.seed(1234)
> sample(1:5,5)
[1] 1 3 2 4 5
> sample(1:5,5)
[1] 4 1 5 2 3
> sample(1:5,5)
[1] 4 3 1 2 5

> set.seed(1234)
> sample(1:5,5)
[1] 1 3 2 4 5
> set.seed(33)
> sample(1:5,5)
[1] 3 2 4 5 1
> set.seed(1234)
> sample(1:5,5)
[1] 1 3 2 4 5
> set.seed(33)
> sample(1:5,5)
[1] 3 2 4 5 1
```

Section E

1. Put the file example2.dat to your local folder. You’ll need to open this file in a text editor to determine whether it has a header or not. Then read it into your R workspace.

```
Reading file in same folder as R workspace
> dat <- read.table(file="example2.dat", header=T)
> dim(dat)
[1] 100   2
```

```
> head(dat)
  yrs   happy
1  10 21.6532
2  12 20.2488
3  12 28.8347
4   8 13.9754
5  11 26.5029
6  15 22.3995
>
```

2. a. How many variables are in the data?

```
> ncol(dat)
[1] 2
>
```

b. What are the names of the variables?

```
> colnames(dat)
[1] "yrs"   "happy"
>
```

c. How many cases are in the data?

```
> nrow(dat)
[1] 100
>
```

d. Print out cases 50 through 60.

```
> dat[50:60,]
   yrs   happy
50   7 19.0773
51  10 20.4393
52  16 20.2455
53  15 22.8212
54  16 27.7986
55  24 18.8662
56  14 14.5189
57   6 21.0589
58  14 23.2751
59  10 25.3203
60  15 21.7294
>
```

e. Print out the values of the variable in the second column for cases 15 through 20.

Method 1

```
> dat[15:20,2]
[1] 20.4796 23.0286 19.5692 24.2943 25.7817 25.3009
```

Method 2

```
> dat$happy[15:20]
[1] 20.4796 23.0286 19.5692 24.2943 25.7817 25.3009
>
```

3. What is the mean of column 1?

```
Method 1
> mean(dat[,1])
[1] 12.25

Method 2
> mean(dat$yrs)
[1] 12.25
>
```

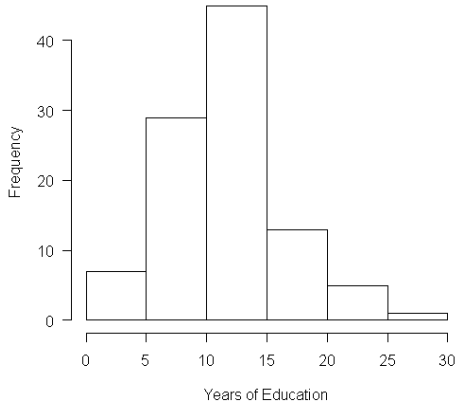
4. What is the median of column 2?

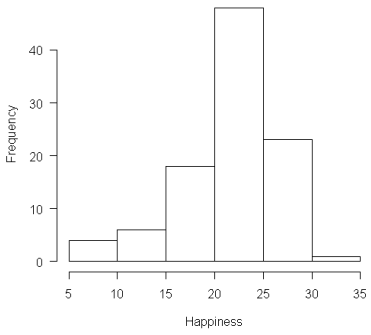
```
Method 1
> median(dat[,2])
[1] 21.7042

Method 2
> median(dat$happy)
[1] 21.7042
>
```

Section F

1. Make histograms of "yrs" and "happy" and save them to a word document.

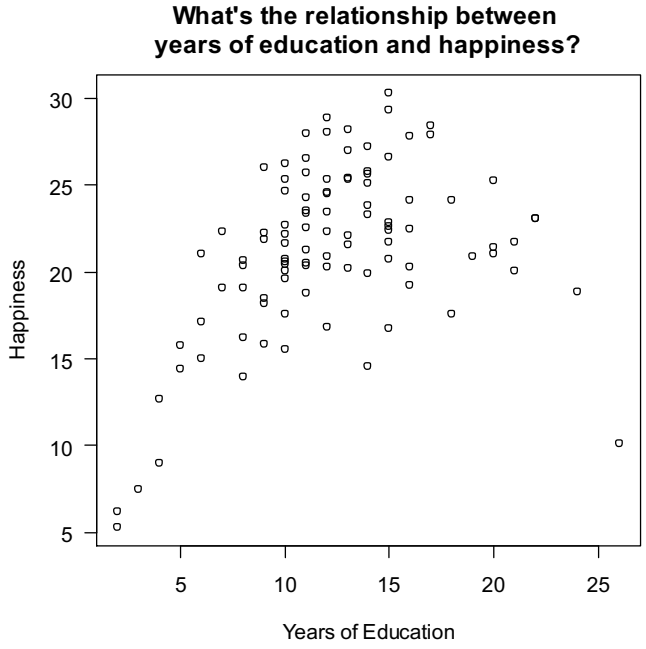
```
> hist(dat$yrs,xlab="Years of Education",las=1,main="")
> hist(dat$happy,xlab="Happiness",las=1,main="")
>
```





2. Plot yrs (years of education) on the x-axis versus happy (happiness) on the y-axis

```
> plot(dat$yrs,dat$happy,xlab="Years of Education",ylab="Happiness",
+ main="What's the relationship between \nyears of education and
happiness?",las=1)
>
```



3. Fit a linear regression model to the data with happy as the dependent variable

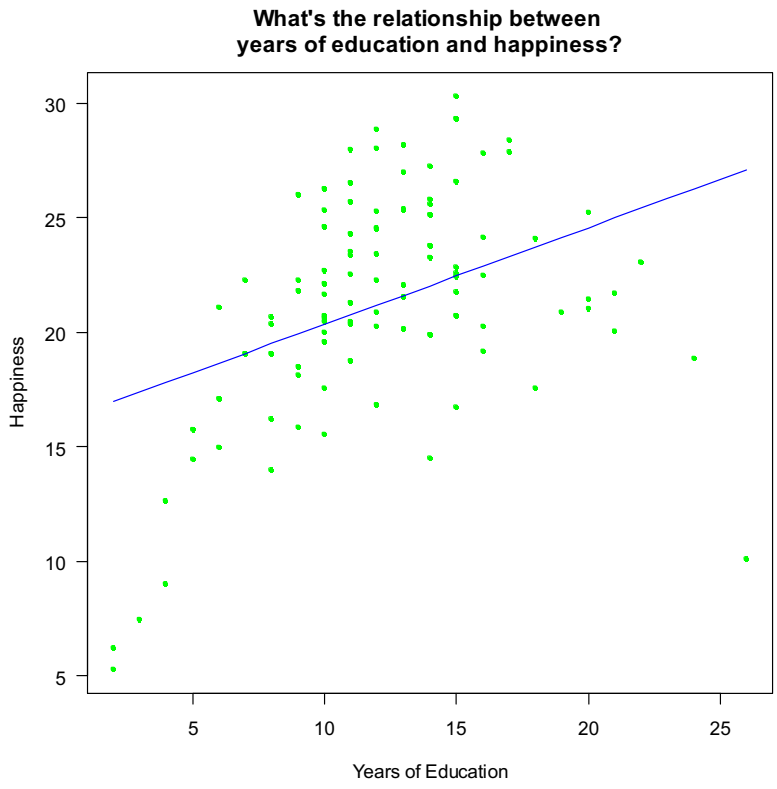
```
> fit <- lm(happy~yrs, data=dat)
> summary(fit)

Call:
lm(formula = happy ~ yrs, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-16.9890  -2.5429   0.3025   3.3167   7.8584

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 16.11834    1.28422  12.551  < 2e-16 ***
yrs          0.42271    0.09813   4.308 3.92e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.519 on 98 degrees of freedom
Multiple R-Squared: 0.1592,     Adjusted R-squared: 0.1506
F-statistic: 18.56 on 1 and 98 DF,  p-value: 3.923e-05

>
```

4. Create a new data set of only those cases who have 12 or more "yrs" (years of education).

```
> new.dat <- subset(dat,yrs>=12)
> dim(new.dat)
[1] 53  2
> head(new.dat)
   yrs   happy
2   12 20.2488
3   12 28.8347
6   15 22.3995
8   12 22.2869
11  14 19.8968
16  22 23.0286
>
```

5. (Optional) Add the linear regression fit in (3) to the raw data plot in (2)

```
### plot the points
> plot(dat$yrs,dat$happy,xlab="Years of Education",ylab="Happiness",
main="What's the relationship between \nyears of education and happiness?",
las=1,pch=20,col="green")

### create a vector that spans the years of education in this sample
> yrs <- seq(min(dat$yrs),max(dat$yrs),1)
> yrs
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

### get the coefficients from the linear regression
> fit$coef
(Intercept)         yrs
 16.1183425   0.4227068

### create the predicted values for the years of education vector
> pred.happy <- fit$coef[2] * yrs + fit$coef[1]

### check this calculation
> pred.happy[1]
[1] 16.96376
> 2 * .4227068 + 16.1183425
[1] 16.96376
> pred.happy[25]
[1] 27.10872
> 26 * .4227068 + 16.1183425
[1] 27.10872

### add this line to the plot
> lines(yrs,pred.happy,col="blue")
>
```

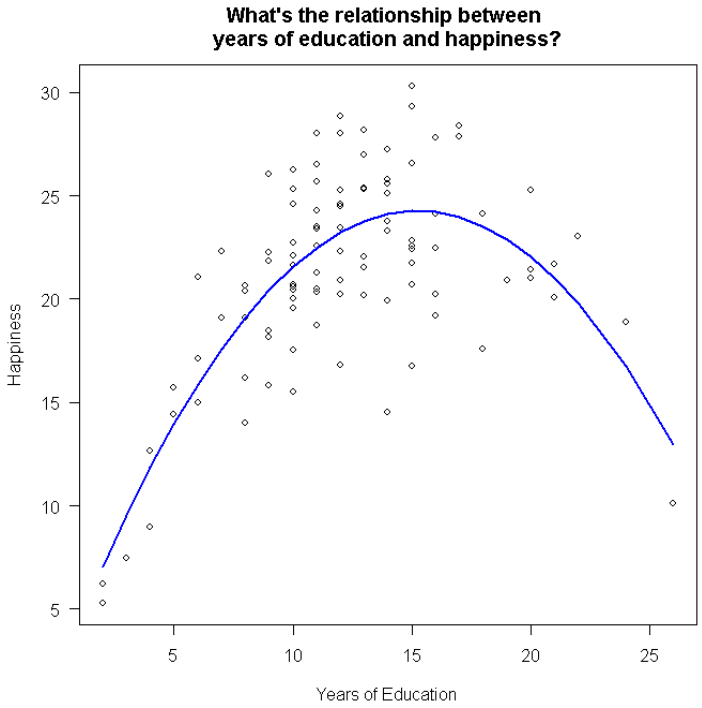


6. (Optional) Add new variable to the data frame called yrs_squared equal to years squared.

```
dat$yrs_squared <- dat$yrs * dat$yrs
fit2 <- lm(happy~yrs + yrs_squared,data=dat)
summary(fit2)
```

```
Call:
lm(formula = happy ~ yrs + yrs_squared, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5845 -1.9986  0.0806  1.8949  6.0656

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.468328   1.748898   0.840    0.403
yrs          2.987385   0.270221  11.055  < 2e-16 ***
yrs_squared -0.097900   0.009965  -9.825 3.20e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.216 on 97 degrees of freedom
Multiple R-Squared: 0.5786,     Adjusted R-squared: 0.5699
F-statistic: 66.58 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
plot(dat$yrs,dat$happy,xlab="Years of Education",ylab="Happiness",
main="What's the relationship between \nyears of education and
happiness?",las=1)
lines(sort(dat$yrs),y.hat[order(dat$yrs)],col="blue",lwd=2)
```



Section G

1. Sample with replacement 20 values from the first forty consecutive positive even integers.

```
> G1 <- sample(1:40,20,replace=TRUE)
> G1
 [1] 21 18 14  1  5 28 11 10 14 32 34 31 16  6 37 23  2 20 15 39
>
```

a. how many values in this sample are equal to 14?

```
> sum(G1 == 14)
[1] 2
>
```

b. how many values in this sample are greater than 10?

```
> sum(G1 > 10)
[1] 15
>
```

2. Sample *without* replacement 20 values from the first forty consecutive positive even integers.

```
> G2 <- sample(1:40,20,replace=FALSE)
> G2
 [1] 32 24 14 29 10 18 27 15 11 20  5 16 26  8 39  9  4 33 25 17
>
```

a. how many values in this sample are equal to 14?

```
> sum(G2==14)
[1] 1
>
```

b. how many values in this sample not equal to 13 ?

```
> sum(G2!=13)
[1] 20
> sum(G2==13)
[1] 0
>
```

3. Use the `rnorm()` function to generate a sample of 100 values from a normal distribution with mean of 5 and standard deviation of 2.

```
> set.seed(478)
> G3 <- rnorm(100,mean=5,sd=2)
```

a. What is the mean of this sample?

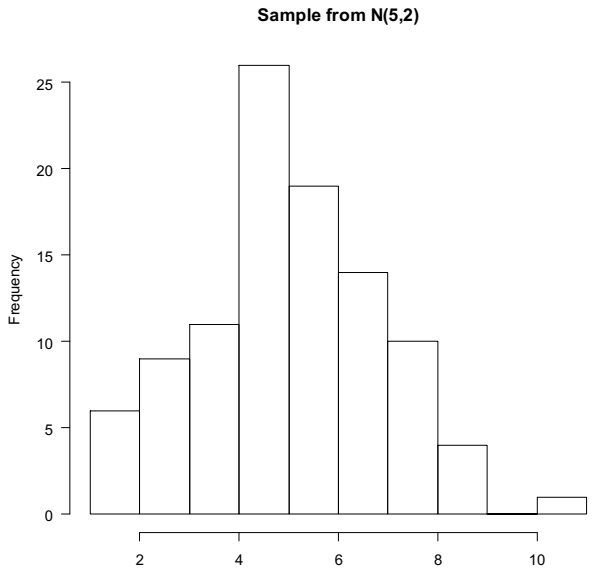
```
> mean(G3)
[1] 4.977619
```

b. What is the standard deviation of this sample?

```
> sd(G3)
[1] 1.889232
```

c. Plot a histogram of this sample

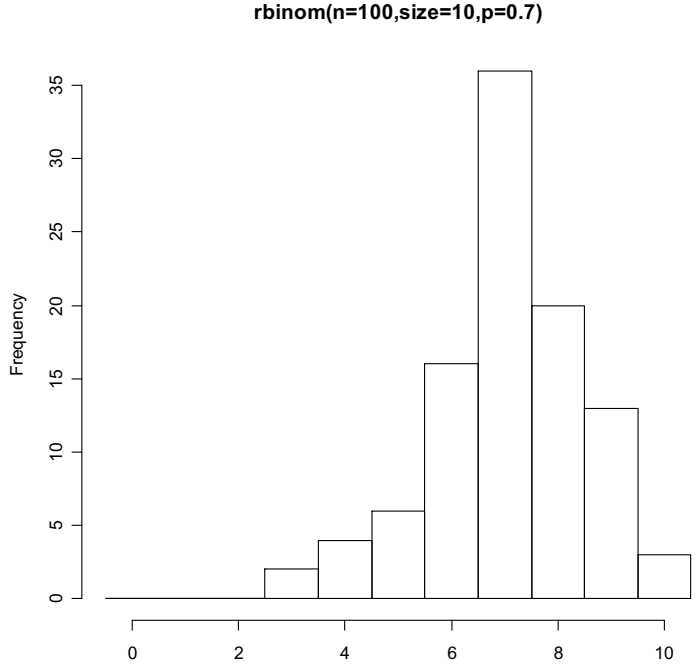
```
> hist(G3,las=1,xlab="",main="Sample from N(5,2)")
>
```



4. (Optional) Use the `rbinom()` function, let the number of samples be 100, the number of coin tosses be 10, plot a histogram of the distribution of the number of "successes" when the probability of a success is 0.7

```
> set.seed(111)
> G4 <- rbinom(n=100,size=10,p=0.7)
```

```
> G4
  [1]  7  6  8  7  8  7 10  7  7  9  7  7  9  9  8  7  8  4  8  7  7  8  8  7  4
 [26]  8  6  8  6  7  9  7  7  7  8  6  9  6  7  6  7  8  7  5  7  8  7  5  7  6
 [51]  6  7  9  8  3  7  7  7  9  6 10  7  8  8  5  8  9  8 10  8  5  6  7  9  7
 [76]  7  8  4  9  6  9  5  9  7  9  8  6  6  6  7  7  5  7  7  6  7  3  7  4  6
> hist(G4,breaks=seq(-
0.5,10.5,1),xlab="",main="rbinom(n=100,size=10,p=0.7)")
```



>

5. (Optional) What would be the percentage of test-takers expected to score an 800 on the GRE's assuming a normal distribution with a mean of 520 and a standard deviation of 110?

As proportion

```
> pnorm(800,mean=520,sd=110,lower.tail=FALSE)
[1] 0.005456779
```

As percentage

```
> pnorm(800,mean=520,sd=110,lower.tail=FALSE)*100
[1] 0.5456779
>
```