**Big Data Analytics-Lab-CSE6034\_Lab-Assessment - 2:**

**Implementation of R Programming**

**Submitted By: 20MAI0001 - NIHARIKA MAITRA**

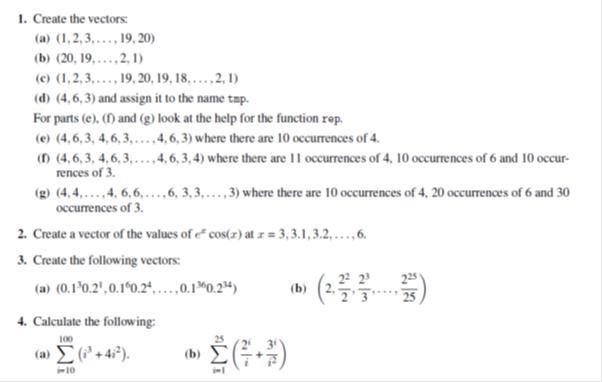
**Git Repo link :** [**https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01/blob/main/20MAI0001-Niharika-Maitra\_BDA-Lab-CSE6034\_Lab-Assessment%20-%202%20Implementation%20of%20R%20Programming.docx**](https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01/blob/main/20MAI0001-Niharika-Maitra_BDA-Lab-CSE6034_Lab-Assessment%20-%202%20Implementation%20of%20R%20Programming.docx)

[**https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01/blob/main/20MAI0001-Niharika-Maitra\_BDA-Lab-CSE6034\_Lab-Assessment%20-%202%20Implementation%20of%20R%20Programming.docx**](https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01/blob/main/20MAI0001-Niharika-Maitra_BDA-Lab-CSE6034_Lab-Assessment%20-%202%20Implementation%20of%20R%20Programming.docx)

[**https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01**](https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01)

[**https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01**](https://github.com/Niharika-20-MAI-01/Winter-2021-Big-Data-Analytics-LabCSE6034-Niharika-20MAI01)

**R Programming Code for Implementation on Vectors and their corresponding Outputs obtained on executing the R Programming Code in R Studio :**

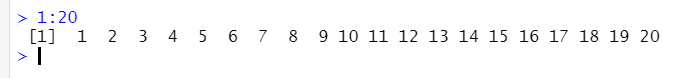
**Q. A) **

**Solution A:-**

**1a)**

**> 1:20**

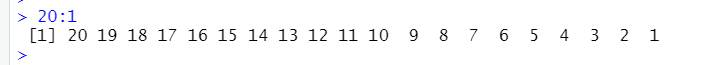
**[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20**

****

**1b)**

**> 20:1**

**[1] 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1**

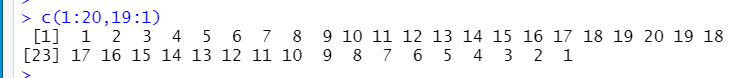
****

**1c)**

**> c(1:20,19:1)**

**[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 19 18**

**[23] 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1**

****

**1d)**

**> tmp <- c(4,6,3)**

**> tmp**

**[1] 4 6 3**

****

**1e)**

**> rep(tmp,10)**

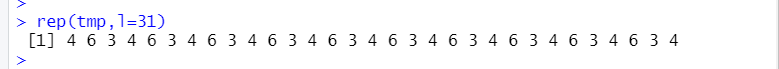
**[1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3**

****

**1f)**

**> rep(tmp,l=31)**

**[1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4**

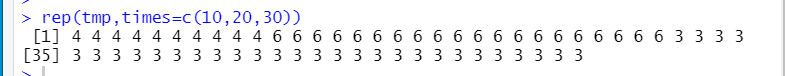
****

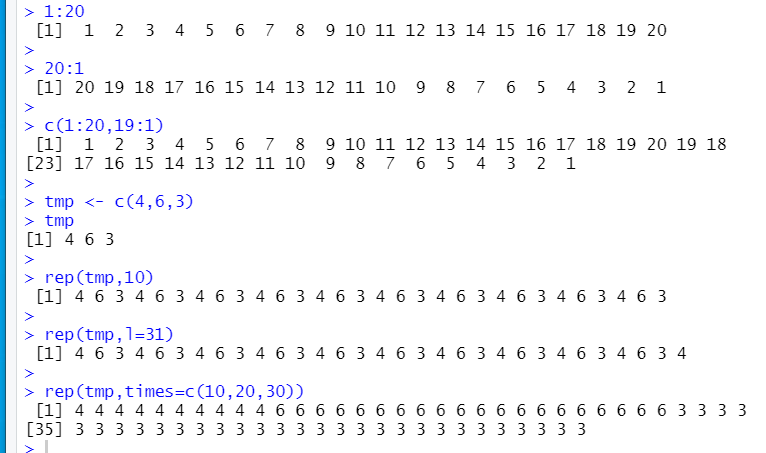
**1g)**

**> rep(tmp,times=c(10,20,30))**

**[1] 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 3 3 3 3**

**[35] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3**

****

****

**2.)**

**> tpm <- seq(3,6,by=0.1)**

**> exp(tpm)\*cos(tpm)**

**[1] -19.884531 -22.178753 -24.490697 -26.773182 -28.969238 -31.011186**

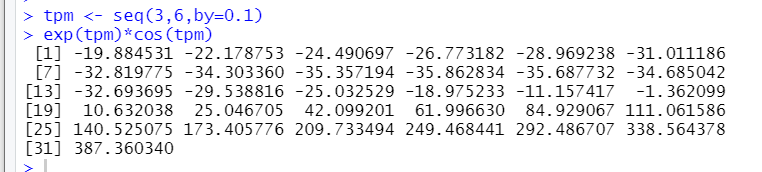
**[7] -32.819775 -34.303360 -35.357194 -35.862834 -35.687732 -34.685042**

**[13] -32.693695 -29.538816 -25.032529 -18.975233 -11.157417 -1.362099**

**[19] 10.632038 25.046705 42.099201 61.996630 84.929067 111.061586**

**[25] 140.525075 173.405776 209.733494 249.468441 292.486707 338.564378**

**[31] 387.360340**

****

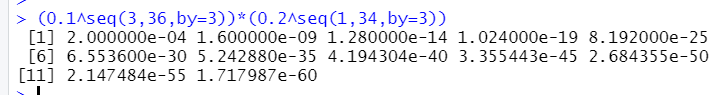
**3 a)**

**> (0.1^seq(3,36,by=3))\*(0.2^seq(1,34,by=3))**

**[1] 2.000000e-04 1.600000e-09 1.280000e-14 1.024000e-19 8.192000e-25**

**[6] 6.553600e-30 5.242880e-35 4.194304e-40 3.355443e-45 2.684355e-50**

**[11] 2.147484e-55 1.717987e-60**

****

**3 b)**

**> (2^(1:25))/(1:25)**

**[1] 2.000000e+00 2.000000e+00 2.666667e+00 4.000000e+00 6.400000e+00**

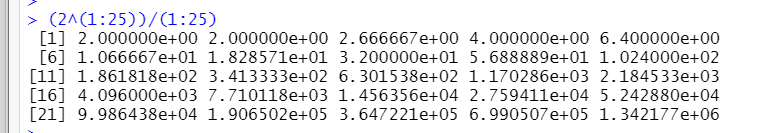
**[6] 1.066667e+01 1.828571e+01 3.200000e+01 5.688889e+01 1.024000e+02**

**[11] 1.861818e+02 3.413333e+02 6.301538e+02 1.170286e+03 2.184533e+03**

**[16] 4.096000e+03 7.710118e+03 1.456356e+04 2.759411e+04 5.242880e+04**

**[21] 9.986438e+04 1.906502e+05 3.647221e+05 6.990507e+05 1.342177e+06**

**>**

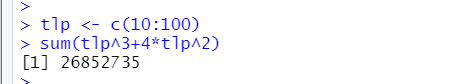
****

**4 a)**

**> tlp <- c(10:100)**

**> sum(tlp^3+4\*tlp^2)**

**[1] 26852735**

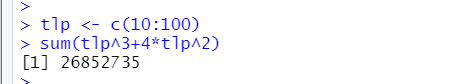
****

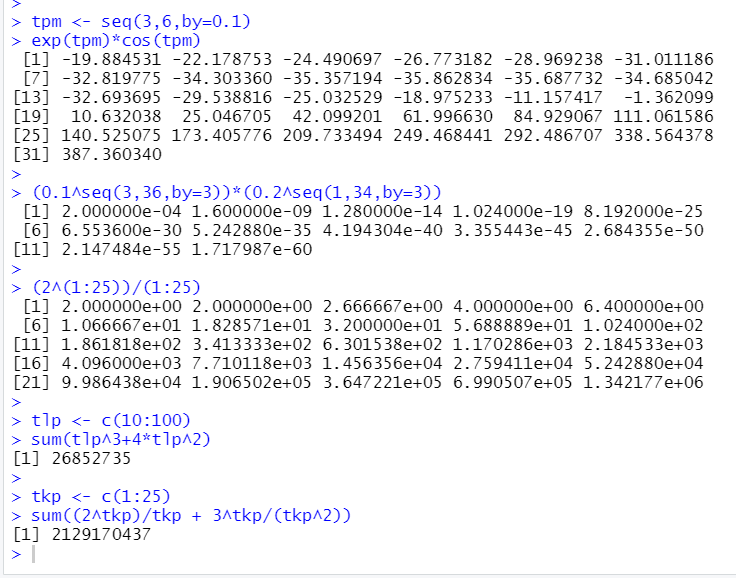
**4 b)**

**> tkp <- c(1:25)**

**> sum((2^tkp)/tkp + 3^tkp/(tkp^2))**

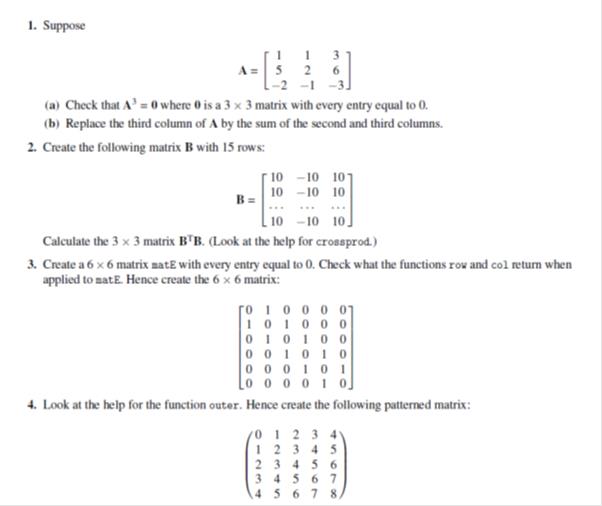
**[1] 2129170437**

****

****

**R Programming Code for Implementation on Arrays and Matrices and their corresponding Outputs obtained on executing the R Programming Code in R Studio :**

**Q. B).1**

****

**Solution B.1:-**

**1)**

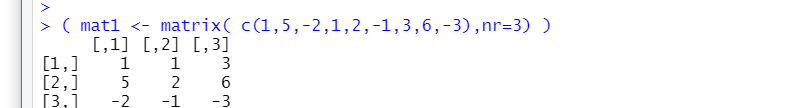
**> ( mat1 <- matrix( c(1,5,-2,1,2,-1,3,6,-3),nr=3) )**

**[,1] [,2] [,3]**

**[1,] 1 1 3**

**[2,] 5 2 6**

**[3,] -2 -1 -3**

****

**1 a)**

**> mat1%\*%mat1%\*%mat1**

**[,1] [,2] [,3]**

**[1,] 0 0 0**

**[2,] 0 0 0**

**[3,] 0 0 0**

****

**1 b)**

**> mat1[,3] <- mat1[,2]+mat1[,3]**

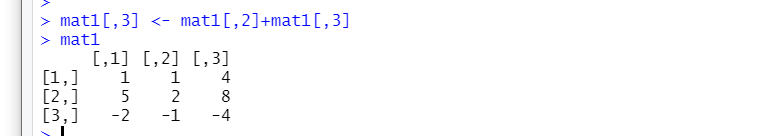
**> mat1**

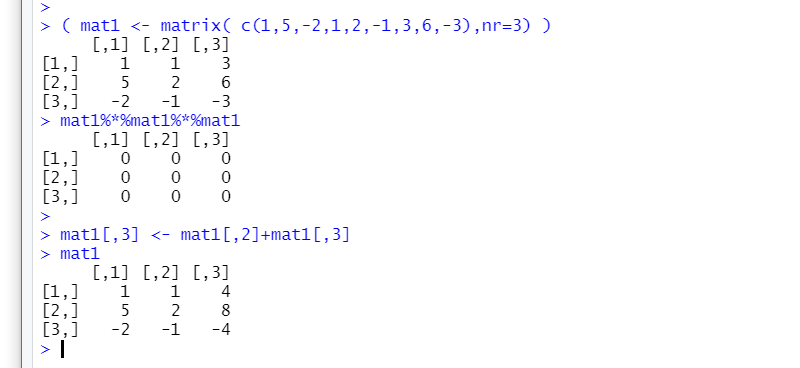
**[,1] [,2] [,3]**

**[1,] 1 1 4**

**[2,] 5 2 8**

**[3,] -2 -1 -4**

****

****

**2)**

**> mat2 <- matrix(c(10,-10,10), b=T, nc=3, nr=15)**

**> mat2**

**[,1] [,2] [,3]**

**[1,] 10 -10 10**

**[2,] 10 -10 10**

**[3,] 10 -10 10**

**[4,] 10 -10 10**

**[5,] 10 -10 10**

**[6,] 10 -10 10**

**[7,] 10 -10 10**

**[8,] 10 -10 10**

**[9,] 10 -10 10**

**[10,] 10 -10 10**

**[11,] 10 -10 10**

**[12,] 10 -10 10**

**[13,] 10 -10 10**

**[14,] 10 -10 10**

**[15,] 10 -10 10**

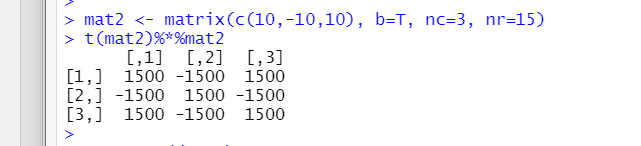
**> t(mat2)%\*%mat2**

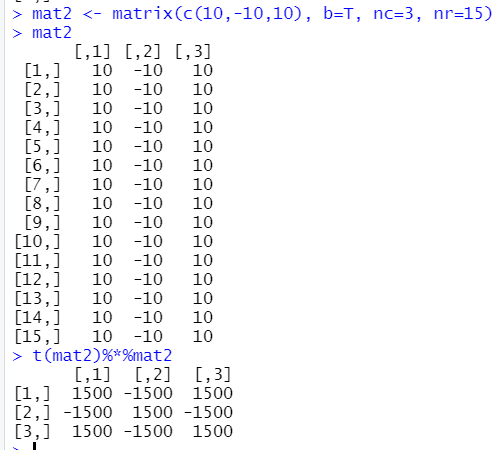
**[,1] [,2] [,3]**

**[1,] 1500 -1500 1500**

**[2,] -1500 1500 -1500**

**[3,] 1500 -1500 1500**

****

****

**3)**

**> mat3 <- matrix(0,nr=6,nc=6)**

**> mat3**

**[,1] [,2] [,3] [,4] [,5] [,6]**

**[1,] 0 0 0 0 0 0**

**[2,] 0 0 0 0 0 0**

**[3,] 0 0 0 0 0 0**

**[4,] 0 0 0 0 0 0**

**[5,] 0 0 0 0 0 0**

**[6,] 0 0 0 0 0 0**

**> mat3[ abs(col(mat3)-row(mat3))==1 ] <- 1**

**> mat3**

**[,1] [,2] [,3] [,4] [,5] [,6]**

**[1,] 0 1 0 0 0 0**

**[2,] 1 0 1 0 0 0**

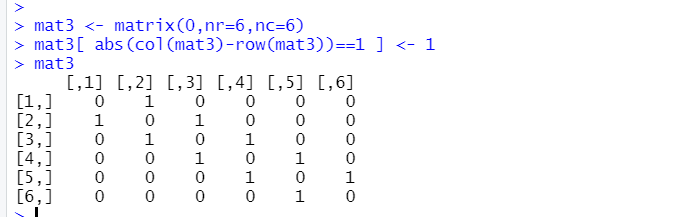
**[3,] 0 1 0 1 0 0**

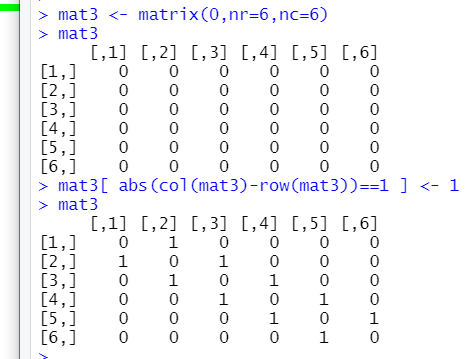
**[4,] 0 0 1 0 1 0**

**[5,] 0 0 0 1 0 1**

**[6,] 0 0 0 0 1 0**

**>**

****

****

**4)**

**> mat4 <- outer(0:4,0:4,"+")**

**> mat4**

**[,1] [,2] [,3] [,4] [,5]**

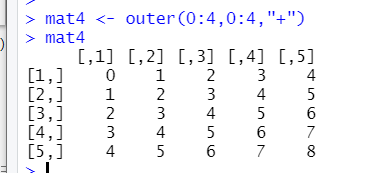
**[1,] 0 1 2 3 4**

**[2,] 1 2 3 4 5**

**[3,] 2 3 4 5 6**

**[4,] 3 4 5 6 7**

**[5,] 4 5 6 7 8**

****

**Q. B).2**

Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix

**Solution B.2:-**

**> v1 <- c(1:10)**

**> v2 <- c(11:20)**

**> m1 <- array(v1,c(3,3))**

**> m2 <- array(v2,c(3,3))**

**> v1**

**[1] 1 2 3 4 5 6 7 8 9 10**

**> v2**

**[1] 11 12 13 14 15 16 17 18 19 20**

**> m1**

**[,1] [,2] [,3]**

**[1,] 1 4 7**

**[2,] 2 5 8**

**[3,] 3 6 9**

**> m2**

**[,1] [,2] [,3]**

**[1,] 11 14 17**

**[2,] 12 15 18**

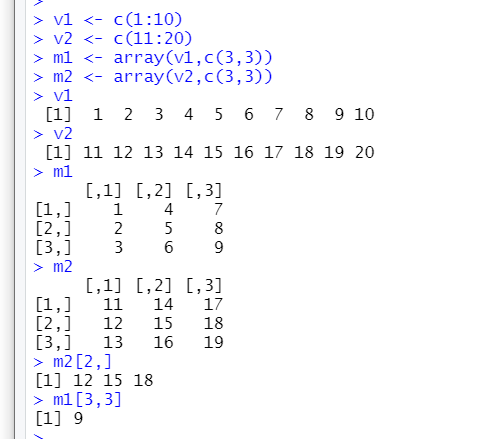
**[3,] 13 16 19**

**> m2[2,]**

**[1] 12 15 18**

**> m1[3,3]**

**[1] 9**

****

**R Programming Code for Implementation on data input and output and their corresponding Outputs obtained on executing the R Programming Code in R Studio :**

**Q. C.1)**

Write a function to print square of each number in the sequence

**Solution C.1:-**

**> sqinseq <- function(arg){**

**+ print(arg^2)**

**+ }**

**> x <- seq(1:10)**

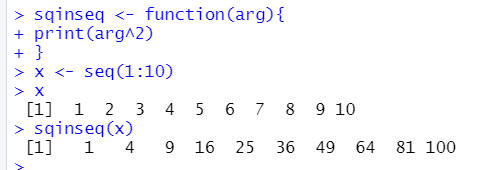
**> x**

**[1] 1 2 3 4 5 6 7 8 9 10**

**> sqinseq(x)**

**[1] 1 4 9 16 25 36 49 64 81 100**

**>**

****

**Q. C.2)**

Create a for loop that, given a numeric vector, prints out one number per line, with its square

and cube alongside.

**Solution C.2:-**

**> for(n in 1:10)**

**+ {**

**+ print(c(n,n^2,n^3))**

**+ }**

**[1] 1 1 1**

**[1] 2 4 8**

**[1] 3 9 27**

**[1] 4 16 64**

**[1] 5 25 125**

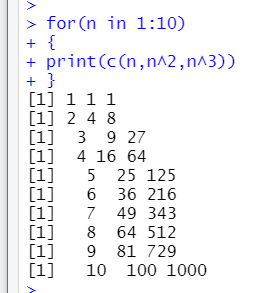
**[1] 6 36 216**

**[1] 7 49 343**

**[1] 8 64 512**

**[1] 9 81 729**

**[1] 10 100 1000**

****

**R Programming Code for Implementation on Functions and their corresponding Outputs obtained on executing the R Programming Code in R Studio :**

**Q. D.1)**

****

**Solution D.1:-**

**> funC <- function(n)**

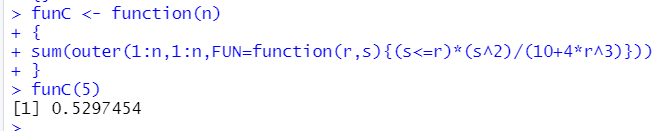
**+ {**

**+ sum(outer(1:n,1:n,FUN=function(r,s){(s<=r)\*(s^2)/(10+4\*r^3)}))**

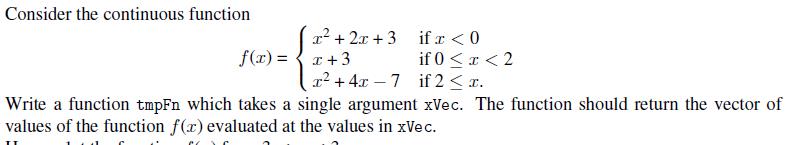
**+ }**

**> funC(5)**

**[1] 0.5297454**

****

**Q. D.2)**

****

**Solution D.2:-**

**> tmpFn <- function(x)**

**+ {**

**+ ifelse(x<0, x^2 +2\*x +3, ifelse(x<2, x+3, x^2 + 4\*x -7))**

**+ }**

**> var1 <- seq(-3, 3, len=100)**

**> var1**

**[1] -3.00000000 -2.93939394 -2.87878788 -2.81818182 -2.75757576**

**[6] -2.69696970 -2.63636364 -2.57575758 -2.51515152 -2.45454545**

**[11] -2.39393939 -2.33333333 -2.27272727 -2.21212121 -2.15151515**

**[16] -2.09090909 -2.03030303 -1.96969697 -1.90909091 -1.84848485**

**[21] -1.78787879 -1.72727273 -1.66666667 -1.60606061 -1.54545455**

**[26] -1.48484848 -1.42424242 -1.36363636 -1.30303030 -1.24242424**

**[31] -1.18181818 -1.12121212 -1.06060606 -1.00000000 -0.93939394**

**[36] -0.87878788 -0.81818182 -0.75757576 -0.69696970 -0.63636364**

**[41] -0.57575758 -0.51515152 -0.45454545 -0.39393939 -0.33333333**

**[46] -0.27272727 -0.21212121 -0.15151515 -0.09090909 -0.03030303**

**[51] 0.03030303 0.09090909 0.15151515 0.21212121 0.27272727**

**[56] 0.33333333 0.39393939 0.45454545 0.51515152 0.57575758**

**[61] 0.63636364 0.69696970 0.75757576 0.81818182 0.87878788**

**[66] 0.93939394 1.00000000 1.06060606 1.12121212 1.18181818**

**[71] 1.24242424 1.30303030 1.36363636 1.42424242 1.48484848**

**[76] 1.54545455 1.60606061 1.66666667 1.72727273 1.78787879**

**[81] 1.84848485 1.90909091 1.96969697 2.03030303 2.09090909**

**[86] 2.15151515 2.21212121 2.27272727 2.33333333 2.39393939**

**[91] 2.45454545 2.51515152 2.57575758 2.63636364 2.69696970**

**[96] 2.75757576 2.81818182 2.87878788 2.93939394 3.00000000**

**> tmpFn(var1)**

**[1] 6.000000 5.761249 5.529844 5.305785 5.089073 4.879706**

**[7] 4.677686 4.483012 4.295684 4.115702 3.943067 3.777778**

**[13] 3.619835 3.469238 3.325987 3.190083 3.061524 2.940312**

**[19] 2.826446 2.719927 2.620753 2.528926 2.444444 2.367309**

**[25] 2.297521 2.235078 2.179982 2.132231 2.091827 2.058770**

**[31] 2.033058 2.014692 2.003673 2.000000 2.003673 2.014692**

**[37] 2.033058 2.058770 2.091827 2.132231 2.179982 2.235078**

**[43] 2.297521 2.367309 2.444444 2.528926 2.620753 2.719927**

**[49] 2.826446 2.940312 3.030303 3.090909 3.151515 3.212121**

**[55] 3.272727 3.333333 3.393939 3.454545 3.515152 3.575758**

**[61] 3.636364 3.696970 3.757576 3.818182 3.878788 3.939394**

**[67] 4.000000 4.060606 4.121212 4.181818 4.242424 4.303030**

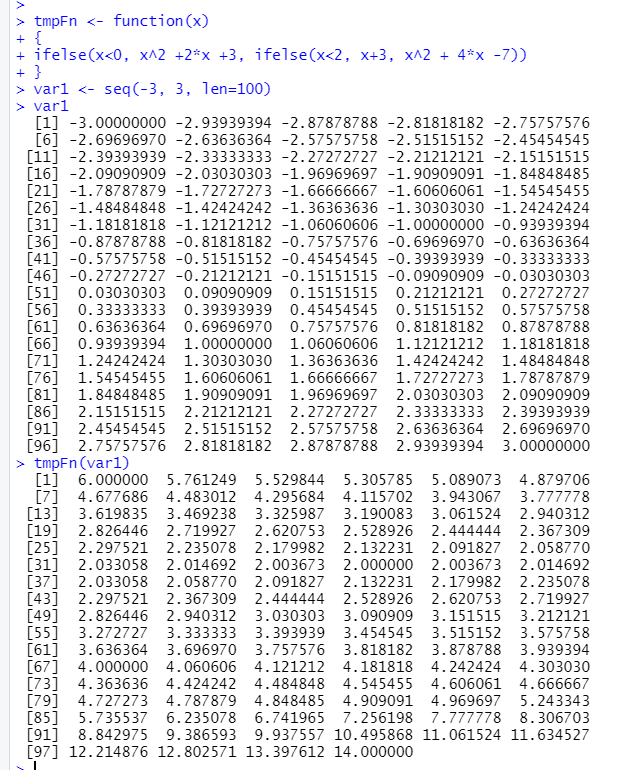
**[73] 4.363636 4.424242 4.484848 4.545455 4.606061 4.666667**

**[79] 4.727273 4.787879 4.848485 4.909091 4.969697 5.243343**

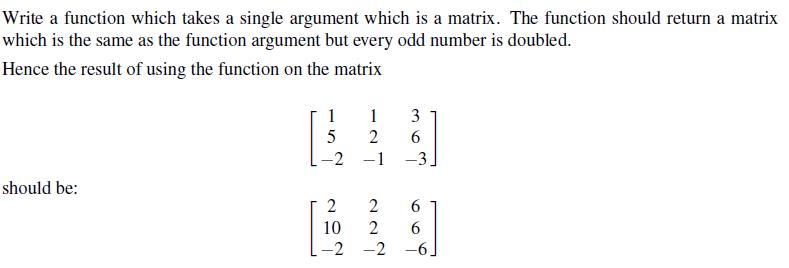
**[85] 5.735537 6.235078 6.741965 7.256198 7.777778 8.306703**

**[91] 8.842975 9.386593 9.937557 10.495868 11.061524 11.634527**

**[97] 12.214876 12.802571 13.397612 14.000000**

****

**Q. D.3)**

****

**Solution D.3:-**

**> matFn <- function(mat)**

**+ {**

**+ mat[mat%%2 == 1] <- 2\*mat[mat%%2 == 1]**

**+ mat**

**+ }**

**>**

**> (mat5 <- matrix( c(1,5,-2,1,2,-1,3,6,-3),nr=3,ncol=3) )**

**[,1] [,2] [,3]**

**[1,] 1 1 3**

**[2,] 5 2 6**

**[3,] -2 -1 -3**

**>**

**> matFn(mat5)**

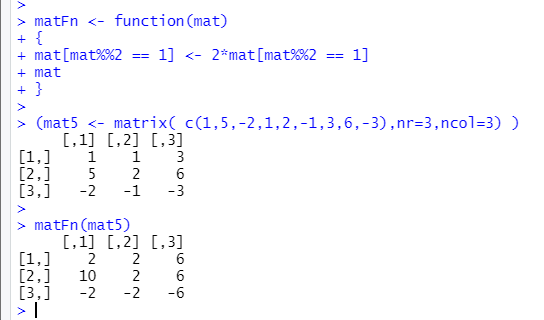
**[,1] [,2] [,3]**

**[1,] 2 2 6**

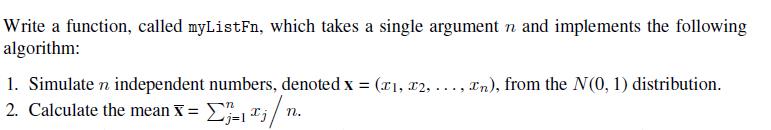
**[2,] 10 2 6**

**[3,] -2 -2 -6**

**>**

****

**Q. D.4)**

****

**Solution D.4:-**

**> myListFn <- function(n)**

**+ {**

**+ xv1 <- rnorm(n)**

**+ xb1 <- mean(xv1)**

**+ yv1 <- sign(xb1)\*rexp(n, rate=abs(1/xb1))**

**+ ct <- sum(abs(yv1)>abs(xv1))**

**+ list(xv1=xv1, xb1=xb1, yv1=yv1, ct=ct)**

**+ }**

**> myListFn(seq(0:1))**

**$xv1**

**[1] 0.6594827 1.0737367**

**$xb1**

**[1] 0.8666097**

**$yv1**

**[1] 2.35135191 0.01919242**

**$ct**

**[1] 1**

**> myListFn(seq(0:1))**

**$xv1**

**[1] -1.024691 1.955431**

**$xb1**

**[1] 0.4653698**

**$yv1**

**[1] 0.4916119 0.0311798**

**$ct**

**[1] 0**

**> myListFn(seq(0:1))**

**$xv1**

**[1] 0.1532382 -1.2035367**

**$xb1**

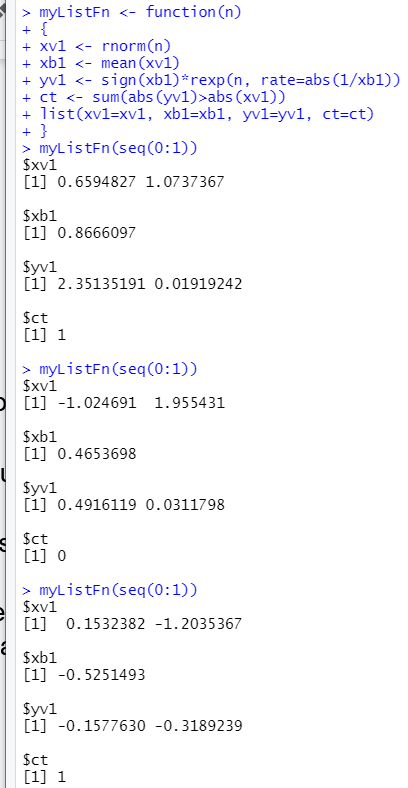
**[1] -0.5251493**

**$yv1**

**[1] -0.1577630 -0.3189239**

**$ct**

**[1] 1**

****

**Q. D.5)**

Use the functions mean() and range() to find the mean and range of:

(a) the numbers 1, 2, . . . , 21

(b) the sample of 50 random normal values, that can be generated from a normaL distribution

with mean 0 and variance 1 using the assignment y

y<- rnorm(50)

(c) the columns height and weight in the data frame women.

[The datasets package that has this data frame is by default attached when R is started.]

**Solution D.5:-**

**a)**

**> x1 <- seq(1:21)**

**> x1**

**[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21**

**> x1m <- mean(x1)**

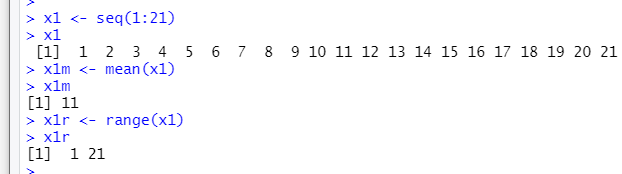
**> x1m**

**[1] 11**

**> x1r <- range(x1)**

**> x1r**

**[1] 1 21**

****

**b)**

**> y <- rnorm(50, mean=0, sd=1)**

**> y**

**[1] 0.15902560 2.02630104 1.36005933 0.36259809 1.34597850 -2.39944472**

**[7] -0.39465210 -1.56382449 -1.88111975 0.97510202 2.64582328 0.74898509**

**[13] 0.46108332 -0.43396174 1.29802821 -1.38356521 0.04454414 -0.45503048**

**[19] -1.43571559 1.52559358 0.25647758 -0.66304951 0.46609959 0.88501485**

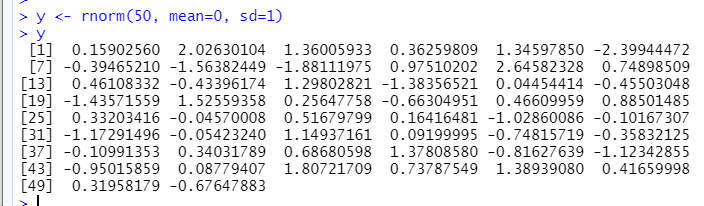
**[25] 0.33203416 -0.04570008 0.51679799 0.16416481 -1.02860086 -0.10167307**

**[31] -1.17291496 -0.05423240 1.14937161 0.09199995 -0.74815719 -0.35832125**

**[37] -0.10991353 0.34031789 0.68680598 1.37808580 -0.81627639 -1.12342855**

**[43] -0.95015859 0.08779407 1.80721709 0.73787549 1.38939080 0.41659998**

**[49] 0.31958179 -0.67647883**

****

**C**

**> class(women)**

**[1] "data.frame"**

**> df1 <- data.frame(women)**

**> df1**

**height weight**

**1 58 115**

**2 59 117**

**3 60 120**

**4 61 123**

**5 62 126**

**6 63 129**

**7 64 132**

**8 65 135**

**9 66 139**

**10 67 142**

**11 68 146**

**12 69 150**

**13 70 154**

**14 71 159**

**15 72 164**

**> df1$height**

**[1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72**

**> df1$weight**

**[1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164**

**> hm <- mean(df1$height)**

**> hm**

**[1] 65**

**> hr <- range(df1$height)**

**> hr**

**[1] 58 72**

**> wm <- mean(df1$weight)**

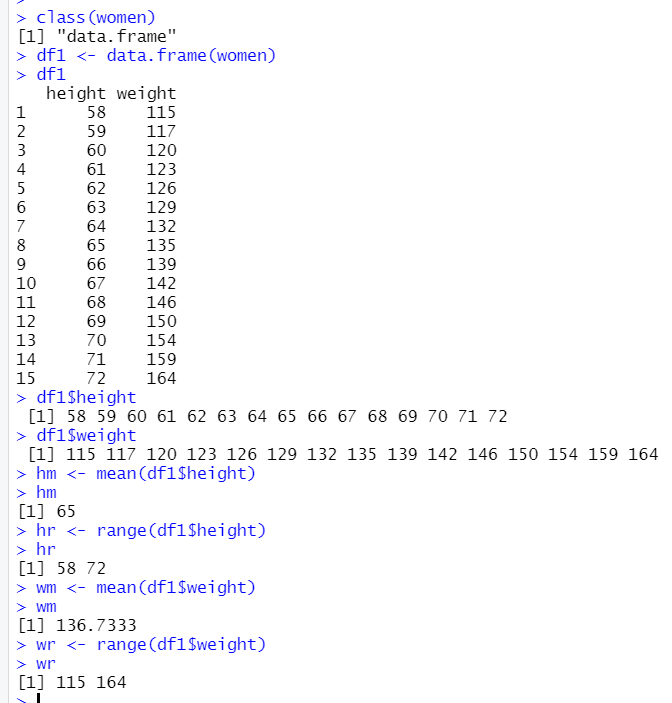
**> wm**

**[1] 136.7333**

**> wr <- range(df1$weight)**

**> wr**

**[1] 115 164**

****

**Q. D.6.1)**

mean(c(1,3,5,6,NA,7,9))

**Solution D.6.1:-**

> mean(c(1,3,5,6,NA,7,9))

[1] NA



**Q. D.6.2)**

mean(c(1,2,NA),na.rm = TRUE)

**Solution D.6.2:-**

**> mean(c(1,2,NA),na.rm = TRUE)**

**[1] 1.5**

****

**R Programming Code for Implementation on Graphs and Visualization and their corresponding Outputs obtained on executing the R Programming Code in R Studio :**

**Q. E.1)**

1.Write correct function to plot a scatter plot in R for the given data

x = c(112,95,101,85,90,117,120,125,80,70), y = c(35,25,31,28,30,40,35,36,25,29) z = c(70,60,70,65,60,80,75,50,52,78) ?

**Solution E.1:-**

> x <- c(112,95,101,85,90,117,120,125,80,70)

> x

[1] 112 95 101 85 90 117 120 125 80 70

> y <- c(35,25,31,28,30,40,35,36,25,29)

> y

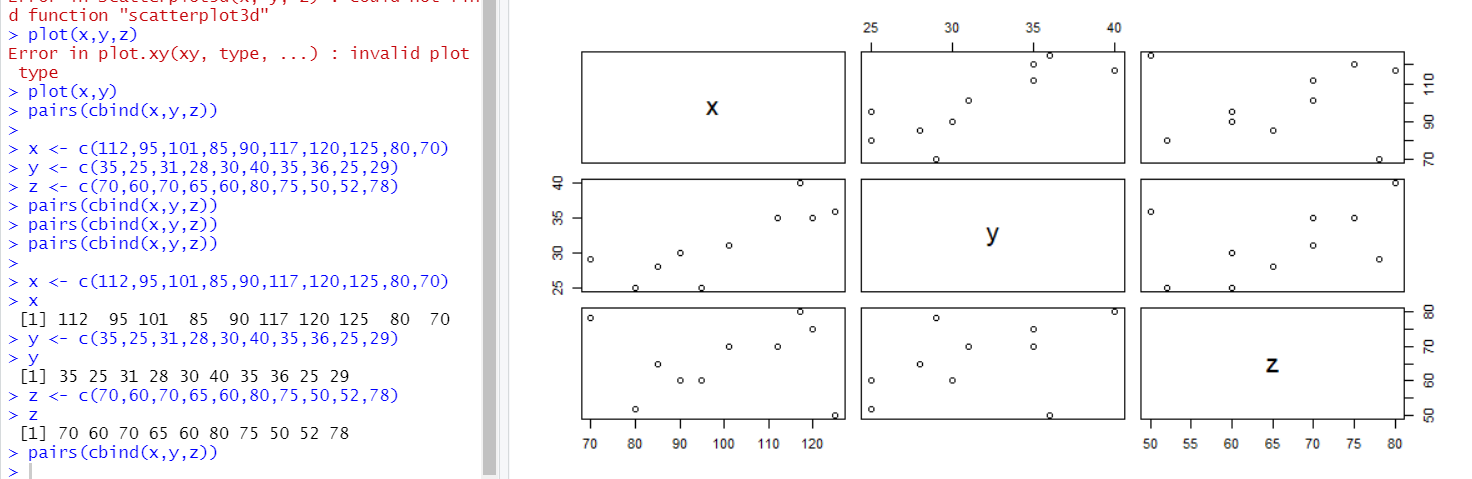
[1] 35 25 31 28 30 40 35 36 25 29

> z <- c(70,60,70,65,60,80,75,50,52,78)

> z

[1] 70 60 70 65 60 80 75 50 52 78

> pairs(cbind(x,y,z))



**Q. E.2)**

2.The paired data on height (x) in centimeters and weight (y) in kilograms of 10 children is obtained. Use correct command to obtain the scatter plot in R for the given data x = c(112,95,101,85,90,117,120,125,80,70), y = c(35,25,31,28,30,40,35,36,25,29) ?

**Solution E.2:-**

**> x1 <- c(112,95,101,85,90,117,120,125,80,70)**

**> x1**

**[1] 112 95 101 85 90 117 120 125 80 70**

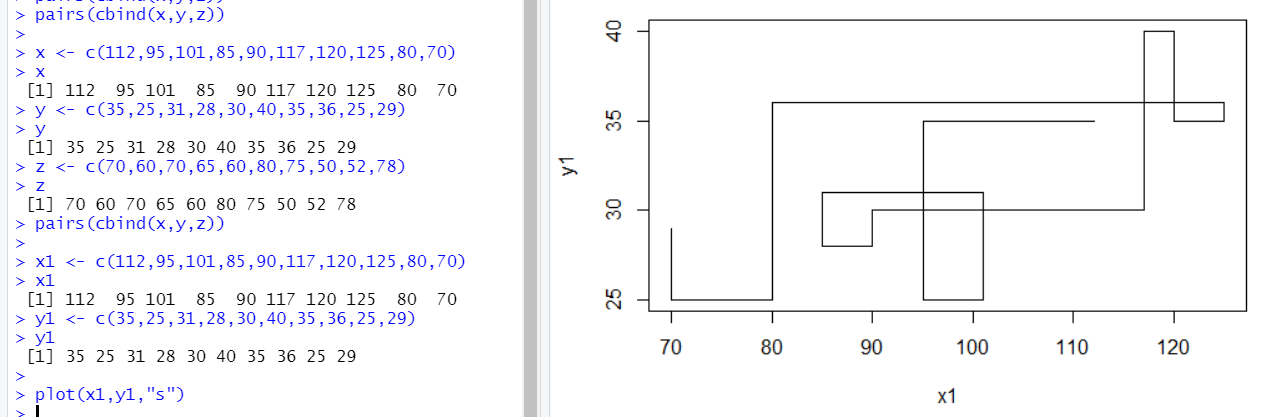
**> y1 <- c(35,25,31,28,30,40,35,36,25,29)**

**> y1**

**[1] 35 25 31 28 30 40 35 36 25 29**

**>**

**> plot(x1,y1,"s")**



**Q. E.3)**

3.In the R package MASS there is a dataset called cats. The variables Bwt and Hwt contain the weight of the body (kg) and the heart (g), respec[1]tively. There are both male and female cats. Try the following commands and explain what happens:

attach(cats)

hist(Hwt[Sex=="M"])

hist(Hwt[Sex=="M"], prob=T)

hist(Hwt[Sex=="M"],

breaks=c(5,10,15,20,25))

Use the xlab and main arguments to change the x-label and the title to something more appropriate.

**Solution E.3:-**

> library("MASS")

> data("cats")

> str(cats)

'data.frame': 144 obs. of 3 variables:

$ Sex: Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...

$ Bwt: num 2 2 2 2.1 2.1 2.1 2.1 2.1 2.1 2.1 ...

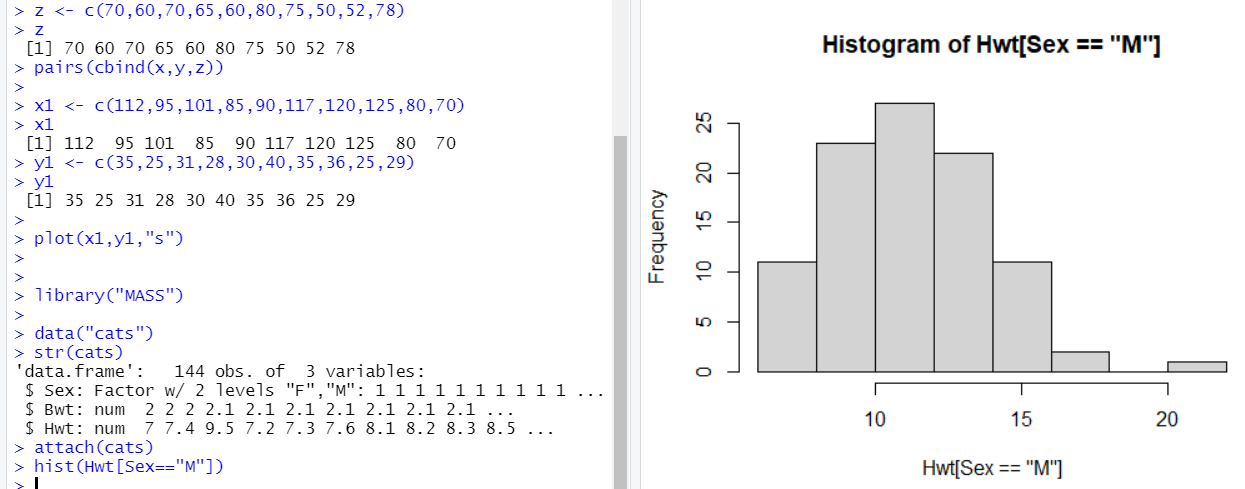
$ Hwt: num 7 7.4 9.5 7.2 7.3 7.6 8.1 8.2 8.3 8.5 …

attach(cats)

> attach(cats)

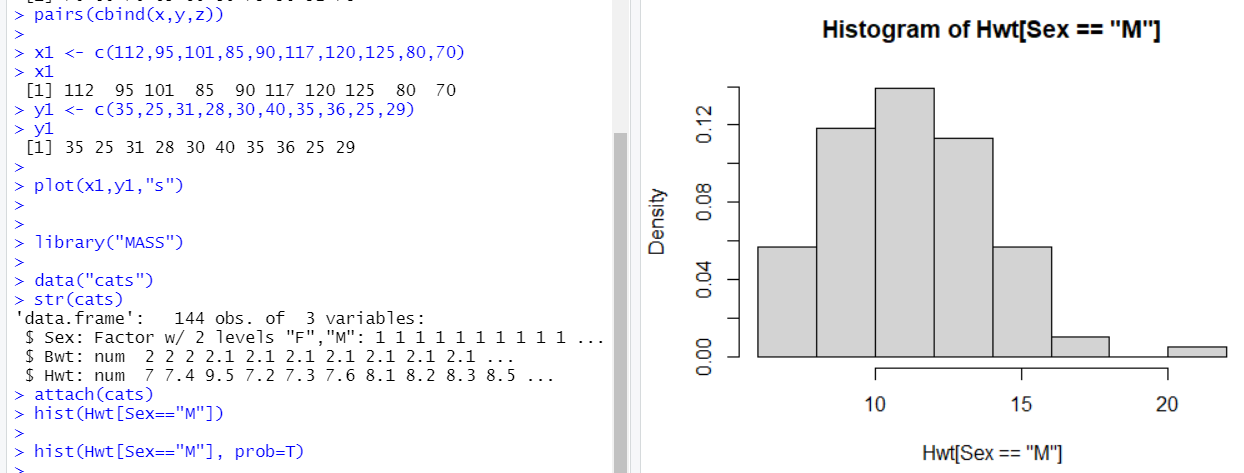
hist(Hwt[Sex=="M"])

> hist(Hwt[Sex=="M"])



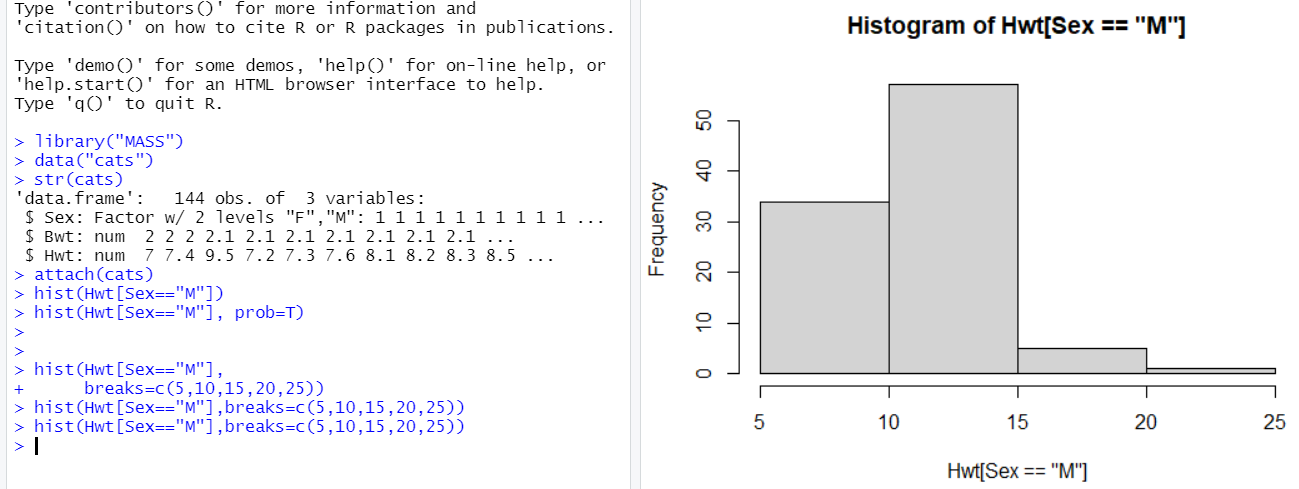
hist(Hwt[Sex=="M"], prob=T)

> hist(Hwt[Sex=="M"], prob=T)



hist(Hwt[Sex=="M"],breaks=c(5,10,15,20,25))

> hist(Hwt[Sex=="M"],breaks=c(5,10,15,20,25))



Use the xlab and main arguments to change the x-label and the title to something more appropriate

