R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

This is the 2nd in series of R Code Files.

Refer the GitHub Repository , for all Code files —> https://github.com/RohitDhankar/R-Beginners-Online-Virtual-Learning-S

Its a good practice from time to time to keep a track of our current Working Directory and list out all the Objects in our R ENVIRONMENT - specially so when we are committing changes to a Git Remote.

VECTOR Operations

```r
getwd()
```

```
## [1] "/home/dhankar/Desktop/R_Own/Proj_1"
```

```r
#
ls()
```

```
## character(0)
```

We could remove any object with command - rm("Object Name")

We can also use print() , to view any objects stored value.

```r
# Code Section -1
a1 <- "FINANCE"
b1 <- "MARKETING"
c1 <- "SALES"
d1 <- 3.1416
char_vector <- c("x","d","c","f")
print(a1)
```

```
## [1] "FINANCE"
```

```r
#
print(char_vector)
```

```
## [1] "x" "d" "c" "f"
```

```r
#
```

Going further with VECTORS .

We combine two or more vectors to get another vector .

```r
# Code Section -2
num_vector <- c(22,22,33,33,44)
print(num_vector)
```

```
## [1] 22 22 33 33 44
```

```r
num_vector1 <- c(11,12,13,14,15)
#
num_vector3 <- c(num_vector,num_vector1)
print(num_vector3)
```

```
##  [1] 22 22 33 33 44 11 12 13 14 15
```

Some basic Maths and Stats with VECTORS.

```r
# Code Section -3
num_vector3 + 5
```

```
##  [1] 27 27 38 38 49 16 17 18 19 20
```

```r
# Adds NUMERIC VALUE = 5 to all ELEMENTS of the Num Vector.
```

```r
# Code Section -4

num_vector1 * num_vector3
```

```
##  [1] 242 264 429 462 660 121 144 169 196 225
```

```r
# First 5 elements of - num_vector3 multiplied by the Five Elements
# of num_vector1 and again the Next 5 elements of num_vector3
# multiplied by the Five Elements of num_vector1
```

Check out the LENGTH of a VECTOR with length()

```r
# Code Section -5

length(num_vector1 * num_vector3)
```

```
## [1] 10
```

```r
# Code Section -6

#num_vector1 %*% num_vector3 # Error in num_vector1 %*% num_vector3 : non-conformable arguments

# Vectors are not of same Length above - below they are of same length

nv <- c(1,2,3,4,5)
nv1 <- c(6,7,8,9,10)

nv %*% nv1 # Inner Product of same Length Vectors
```

```
##      [,1]
## [1,]  130
```

```r
# Algeberic Dot Product as defined by WikiPedia - "https://en.wikipedia.org/wiki/Dot_product"
```

Operate upon a ELEMENT of the Vector.

```r
# Code Section -7


log(num_vector3[2]) # Log Base 2 of 22
```

```
## [1] 3.091042
```

```r
#
log(22)
```

```
## [1] 3.091042
```

```r
#
```

Converting a CHAR Vector into a NUMERIC Vector .

```r
# Code Section -8
ch_v <- c("11","12","13","14","15")
#
class(ch_v)
```

```
## [1] "character"
```

```r
#ch_v + 2 # Error in ch_v + 2 : non-numeric argument to binary operator
# Cant do a Math operation on CHAR Vector - lets Convert into NUM Vector
#
nm_v <- as.numeric(ch_v)
#
class(nm_v)
```

```
## [1] "numeric"
```

```r
nm_v + 2
```

```
## [1] 13 14 15 16 17
```

```r
#
#Summary of the Num Vector as below :-
#
summary(nm_v+2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      13      14      15      15      16      17
```

```r
#
summary(nm_v+5)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      16      17      18      18      19      20
```

```r
#
sum(nm_v+5)
```

```
## [1] 90
```

```r
#
sd(nm_v+5)
```

```
## [1] 1.581139
```

```r
#
max(nm_v+5)
```

```
## [1] 20
```

```r
#
min(nm_v+5)
```

```
## [1] 16
```

```r
#
mean(nm_v+5)
```

```
## [1] 18
```

```r
#
median(nm_v+5)
```

```
## [1] 18
```

```r
#
#The Quantile -
#
quantile(nm_v+5)
```

```
##   0%  25%  50%  75% 100%
##   16   17   18   19   20
```

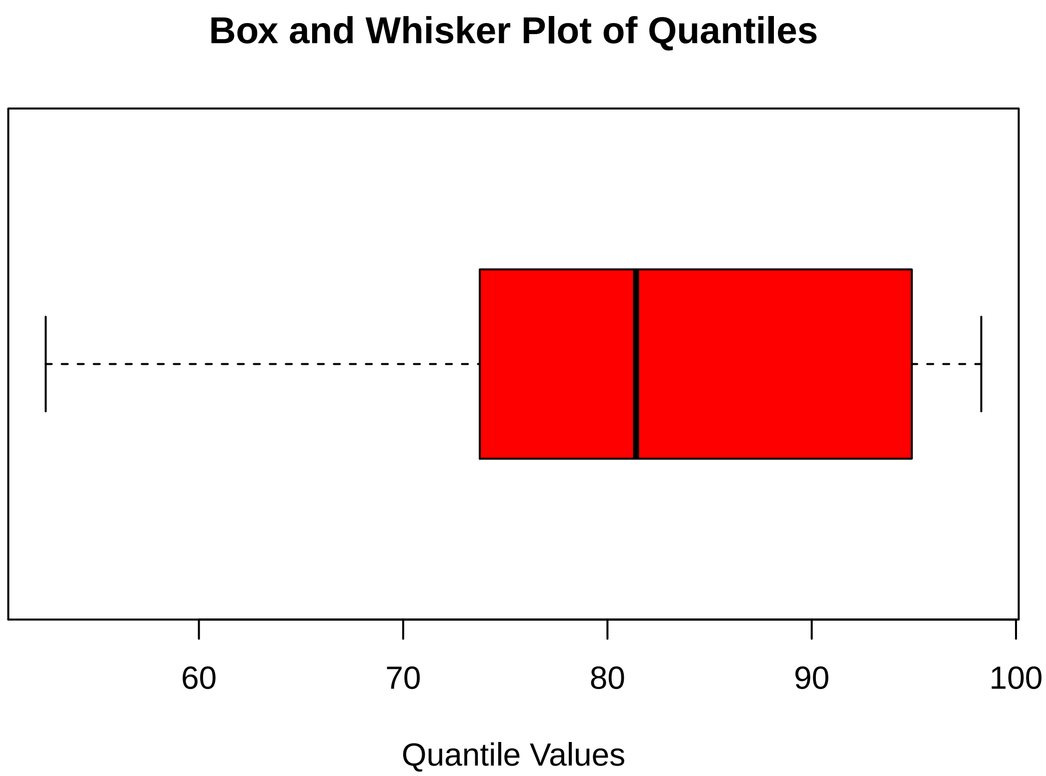
```r
#
quantile(nm_v+100)
```

```
##   0%  25%  50%  75% 100%
##  111  112  113  114  115
```

```r
#
#We can also specify the Quantile buckets or Percentiles as an argument to the Quantile function :-
#
nmv_q <- c(10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,100)
percent_1 <- quantile(nmv_q, c(.50,.75,.84, .97, .99))
percent_1
```

```
##   50%   75%   84%   97%   99%
## 52.50 73.75 81.40 94.90 98.30
```

```r
boxplot(percent_1,col = "red",horizontal = TRUE,
        main = "Box and Whisker Plot of Quantiles",
        xlab = "Quantile Values")
```




```
# Kindly note how the ARGUMENTs to boxplot()
# have been bumped to the next row - keeping in mind
# the Horizontal space of our PDF knit of the .Rmd file

# Seen above we have the MEDIAN quartile - 50% and the UPPER Quartile - 75% along with THREE more perce
```

Wiki reference – Percentile Rank - “https://en.wikipedia.org/wiki/Percentile_rank” #

Intro to ANOVA and BOXPLOTS

We also carry out ONE Way ANOVA or ANALYSIS of VARIANCE test with the BOX and WHISKERS plots as seen below :-

```
# Code Section -9
library(graphics)

nmv_q <- c(10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,100)
percent_1 <- quantile(nmv_q, c(.50,.75,.84, .97, .99))
percent_1
```

```
##   50%   75%   84%   97%   99%
## 52.50 73.75 81.40 94.90 98.30
```

```
percent_2 <- quantile(nmv_q, c(.1, .3, .16, .40, .50))
percent_2
```

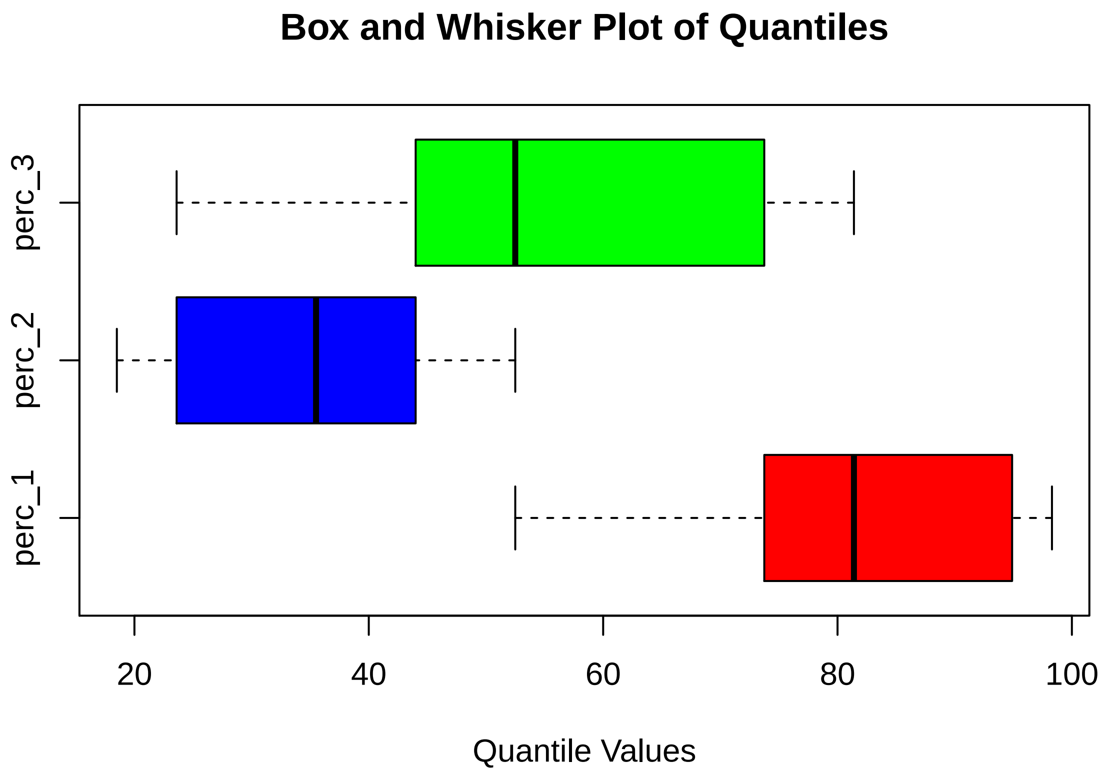
```
##  10%  30%  16%  40%  50%
## 18.5 35.5 23.6 44.0 52.5
```

```
percent_3 <- quantile(nmv_q, c(.16, .40, .50,.75,.84))
percent_3
```

```
##   16%   40%   50%   75%   84%
## 23.60 44.00 52.50 73.75 81.40
```

```
col_boxes = (c("red","blue","green"))

boxplot(percent_1,percent_2,percent_3,col = col_boxes,
        names = c("perc_1","perc_2","perc_3"),horizontal = TRUE,
        main = "Box and Whisker Plot of Quantiles",
        xlab = "Quantile Values")
```




```
# Kindly note the Quantiles are randomly chosen here
# this is not the best way to choose quantiles
# we shall come back for details later in this text
```

rainbow() for Coloring Boxplots -

```
percent_4 <- quantile(nmv_q, c(.16, .40, .50,.95,.99))
percent_4
```

```
##  16%  40%  50%  95%  99%
## 23.6 44.0 52.5 91.5 98.3
```

```
percent_5 <- quantile(nmv_q, c(.16, .24,.32 ,.40,.75))
percent_5
```

```
##   16%   24%   32%   40%   75%
```

```
## 23.60 30.40 37.20 44.00 73.75
```

```r
percent_6 <- quantile(nmv_q, c(.1, .5, .26, .45, .60))
percent_6
```

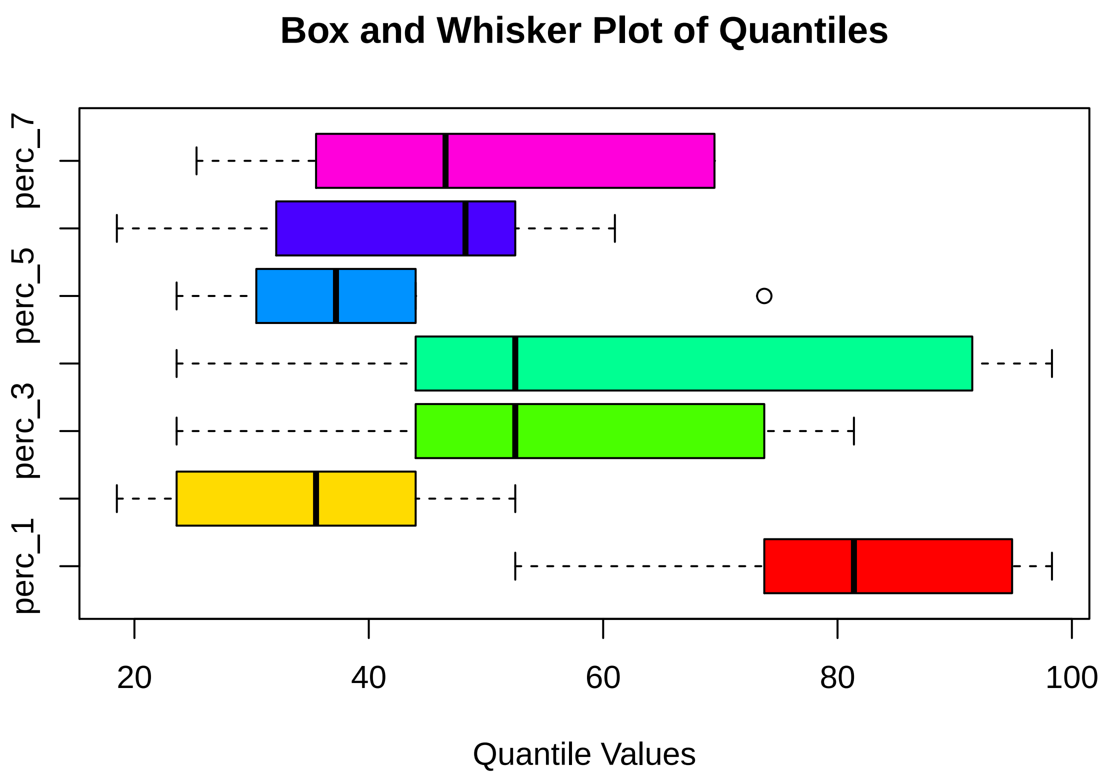
```
##   10%   50%   26%   45%   60%
## 18.50 52.50 32.10 48.25 61.00
```

```r
percent_7 <- quantile(nmv_q, c(.3, .7, .18, .43, .70))
percent_7
```

```
##   30%   70%   18%   43%   70%
## 35.50 69.50 25.30 46.55 69.50
```

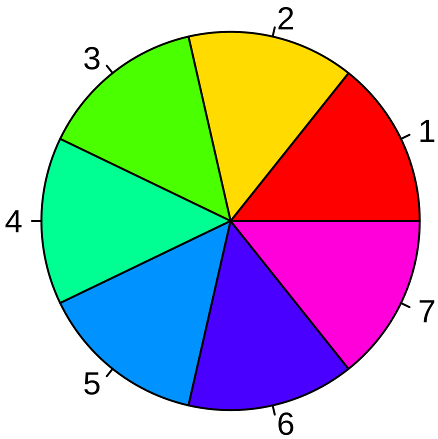
```r
col_rainbow <- rainbow(7)

boxplot(percent_1,percent_2,percent_3,percent_4,percent_5,percent_6,percent_7,col = col_rainbow,
        names = c("perc_1","perc_2","perc_3","perc_4","perc_5","perc_6","perc_7"),horizontal = TRUE,
        main = "Box and Whisker Plot of Quantiles",
        xlab = "Quantile Values")
```




```r
# Just for Fun a PIE Graph --- you always
# need to avoid PIE Graphs


pie(rep(1, 7), col = rainbow(7))
```



MATRICE Operations - TRANSPOSE of a MATRIX

Coming back to MATRICES lets further look at some MATRIX Operations :-

```
# Code Section -10

m1 <- matrix(data=66:69,nrow=2,ncol=2)
m1
```

```
##      [,1] [,2]
## [1,]   66   68
## [2,]   67   69
```

```
# Lets now TRANSPOSE this MATRIX - for more on TRANSPOSE of MATRICES
# kindly refer this Wiki Link -- https://en.wikipedia.org/wiki/Transpose


t(m1)
```

```
##      [,1] [,2]
## [1,]   66   67
## [2,]   68   69
```

```
# As seen below - the DIAGONAL Elements remain as -is .
# 66 and 69 do not move .
# 67 and 68 switch places , thus giving us a Transpose Matrix.

# Lets look at another example of TRANSPOSE ....

m2 <- matrix(data=10:25,nrow=4,ncol=4)
m2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10   14   18   22
## [2,]   11   15   19   23
## [3,]   12   16   20   24
## [4,]   13   17   21   25
```

```
class(m2)
```

```
## [1] "matrix"
```

```
## Note  in the above sequence - 10:25 - both 10 and 25 are included.
# Lets now TRANSPOSE this MATRIX - for more on TRANSPOSE of MATRICES
# kindly refer this Wiki Link -- https://en.wikipedia.org/wiki/Transpose

t(m2)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10   11   12   13
## [2,]   14   15   16   17
## [3,]   18   19   20   21
## [4,]   22   23   24   25
```

```
# As seen below - the DIAGONAL Elements remain as-is.
# 10, 15 , 20 , 25 -- do not move .
# Non Diagonal elements are Transposed ,giving the Transpose Matrix.
```

The Semicolon Notation - RANGE or SEQUENCE

```
# Code Section -11

# Quick recap of the SEQUENCE

a_seq <- 66:69
a_seq
```

```
## [1] 66 67 68 69
```

```
# In the earlier section we create a MATRIX by using a sequence within the COMBINE function
# We can also use the - seq - sequence function as seen below

b_seq <- seq(from=66, to=69, by=1)
b_seq
```

```
## [1] 66 67 68 69
```

```
#

b_seq <- seq(from=66, to=69, by=2)
b_seq
```

```
## [1] 66 68
```

```
#

c_seq <- seq(from=1, to=10, by=2)
c_seq
```

```
## [1] 1 3 5 7 9
```

```
class(c_seq)
```

```
## [1] "numeric"
```

The CBIND and RBIND Functions

We can COLUMN Bind and ROW Bind more than one data structures as seen below -

```
ma1 <- matrix(data=10:15,nrow=3,ncol=2)
ma1
```

```
##      [,1] [,2]
## [1,]   10   13
## [2,]   11   14
## [3,]   12   15
```

```
#
class(ma1)
```

```
## [1] "matrix"
```

```
#
ma2 <- matrix(data=20:25,nrow=3,ncol=2)
ma2
```

```
##      [,1] [,2]
## [1,]   20   23
## [2,]   21   24
## [3,]   22   25
```

```
#
class(ma2)
```

```
## [1] "matrix"
```

```
# ROW Bind the Matrices
ma3 <- rbind(ma1,ma2)
ma3
```

```
##      [,1] [,2]
## [1,]   10   13
## [2,]   11   14
## [3,]   12   15
## [4,]   20   23
## [5,]   21   24
## [6,]   22   25
```

```
#
# COLUMN Bind the Matrices
ma4 <- cbind(ma1,ma2)
ma4
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10   13   20   23
## [2,]   11   14   21   24
## [3,]   12   15   22   25
```

```
# As seen below we need to have same COLUMN Numbers to do a RBIND
#m3 <- rbind(m1,m2)

# # As seen below we need to have same ROW Numbers to do a RBIND
#m3 <- cbind(m1,m2)
```

ROW Bind Two Data Frames -