# Data Visualization in R - Quiz

Question 1

Please write down the structures or syntax of two types of if else in R in the correct layout.

**Answer:**

Type1 for vectors:

```
ifelse(condition, true action, false action)
```

Type2 - not for vectors:

```
if(test_expression){

  statement

}else{

  statement

}
```

Question 2

Please  write the structure or syntax to create a your own function in R with correct layout.

**Answer:**

```
myfunc <- function(Y,X){

    statements

    return(result)

}
```

Question 3

Please generate a vector of 100 normal random variables with mean 5 and standard deviation 2

**Answer:**

Given mean = 5, standard deviation = 2.

Let 'x' be the resultant vector that

generates a vector of 100 normal random variables with mean 5 and standard deviation = 2.

```
x <- rnorm(100,5,2)
```

Question 4

Please write down the difference between package and library?

**Answer:**

A library is a directory that contains a set of packages whereas a package includes functions, data and belongs to one library.

In short library() function is used to load a package and package is not a library.

Question 5

Assume there is a CSV file, datafile.csv where has header and separated using semicolon. Please write down the functions to load this file with header.

**Answer:**

```
d <- read.csv("datafile.csv", sep=";", header=TRUE)
```

Note: delimiters can be separated using sep=" "

Question 6

Assume there is a xlsx file, datafile.xlsx. Please write down the functions to load the first sheet of this xlsx file with header.

**Answer:**

```
install.packages("xlsx")
```

```
library(xlsx)

data <- read.xlsx("datafile.xlsx",1)
```

Question 7

Please write down an example to create a data frame with integer, string, and boolean vectors.

**Answer:**

```
x = c(20,30,50)

y = c("All", "the", "best")

z = c(TRUE,TRUE, FALSE)

df1= data.frame(x,y,z)
```

Question 8

Please write down the structures or syntax of three kind of loops in R in the correct layout.

**Answer:**

loops - for, while, repeat:

```
for(i in 1: n){

   statement

}

while(test-expression){

     statement

}

repeat{
```

```
    statement
}
```

ifelseloop:

```
if (test_expression) {
   statement
} else{
  statement
}
```

---

Question 1

Write down basic functions to draw a line, a scatter plot, boxplot, and a histogram in ggplot2 package. The dataframe is BOD where column 1 is Time and column 2 is demand

**Answer:**

Basic functions to draw a line, a scatter plot, boxplot, and a histogram in ggplot2 package. The dataframe is BOD where column 1 is Time and column 2 is demand:

Firstly, install the required package 'ggplot2'  and that's helps in drawing the required plots as below:

```
install.packages("ggplot2")



#Then loading

library(ggplot2)
```

Drawing a line:

```
ggplot(BOD, aes(x = Time, y = demand)) + geom_line()
```

Drawing  a scatter plot:

```
ggplot(BOD, aes(x = Time, y = demand)) + geom_point()
```

Drawing  a box plot:

```
ggplot(BOD, aes(x = Time, y = demand)) + geom_boxplot()
```

Drawing  a histogram:

```
ggplot(BOD, aes(x = Time, y = demand)) + geom_histogram()
```

Question 2

Write down basic steps to draw a function curve using a data frame and ggplot2 package. The function is f(x) = x^2 + 5*x + 9, x is in [0 20]

**Answer:**

```
install.packages("ggplot2")

library(ggplot2)

x <-c(0:20)

curve<- function(x){

        y=x^2 +5*x+9

       return(y)

}

y<-curve(x)
```

```
data <- data.frame(x,y)

ggplot(data,aes(x=x, y=y)) +geom_line()
```

Question 3

How to deal with different degree of overplotting? How to deal with overplotting when the data is discrete on one or both axes?

**Answer:**

Overplotting occurs when
1)many points obscure each other
2)prevent the viewer from accurately assessing the distribution of the data
W tend to see two types of overplotting: They are namely low and high and can be dealt as follows:
1)Low degree of overplotting
->using smaller points
->using a different shape (like shape 1, a hollow circle)
2)High degree of overplotting
->Make the points semitransparent
->Bin the data into rectangles
->Bin the data into hexagons
->Use box plots

When the data is discrete on one or both axes,overplotting can be dealt as follows:
->randomly jitter the points with position_jitter().
->default the amount of jitter is 40% of the resolution of the data in each direction
->controlled with width and height:
For example, if we take BOD dataframe in which we have two columns as time and demand and plot them on X-axis and Y-axis respectively as follows:

```
op <-ggplot(BOD,aes(x=Time,y=demand))
op + geom_point()
op + geom_point(position="jitter")
op + geom_point(position=position_jitter(width=.5,height=0))
```

Question 4

What is the difference between kernel density, frequency polygon, and violin plot?
**Answer:**
A kernel density curve is an estimate of the population distribution, based on the sample data.
The amount of smoothing depends on the kernel bandwidth: the larger the bandwidth, the more smoothing there is.
The bandwidth can be set with the adjust parameter, which has a default value of 1.
ggplot(faithful,aes(x=waiting)) + geom_line(stat="density", adjust=.25,colour="red") + geom_line(stat="density") + geom_line(stat="density", adjust=2,colour="blue")
A frequency polygon appears similar to a kernel density estimate curve
but it shows the same information as a histogram.
it shows what is in the data, whereas a kernel density estimate is just an estimate requires you to pick some value for the bandwidth.
ggplot(faithful,aes(x=waiting)) + geom_freqpoly(binwidth=4)
A violin plot is a kernel density estimate, mirrored so that it forms a symmetrical shape.
Traditionally, they also have narrow box plots overlaid, with a white dot at the median.
Additionally, the box plot outliers are not displayed by setting outlier.colour=NA.
p + geom_violin() +geom_boxplot(width=.1,fill="black", outlier.colour=NA) + stat_summary(fun.y=median, geom="point",fill="white", shape=21,size=2.5)

—----------

Question 1
**15 / 15 pts**
Please write down three ways to remove the legend.
**Answer:**
–Use guides(), and specify the scale that should have its legend removed

p <-ggplot(PlantGrowth,aes(x=group,y=weight,fill=group)) + geom_boxplot()

# Remove the legend for fill

p + guides(fill=FALSE)

Another way:
–set guide=FALSE in the scale.

p + scale_fill_discrete(guide=FALSE)

Another way:

–use the theming system

p + theme(legend.position="none")

If more than one aesthetic mapping with a legend, this will remove legends for all of them

Question 2

Write down two ways to change the appearance of a legend title's text.

**Answer:**
Type1

–Use theme(legend.title=element_text())
•p <-ggplot(PlantGrowth,aes(x=group,y=weight,fill=group)) + geom_boxplot()
•p + theme(legend.title=element_text(face="italic",family="Times",colour="red",size=14))
??

type2

–p + guides(fill=guide_legend(title.theme=
element_text(face="italic",family="times",colour="red",size=14)))

Question 3

Please write down two ways to set the range of an axis in a plot.

**Answer:**
use  xlim() or  ylim() to set the minimum and maximum values of a continuous axis.

p <-ggplot(PlantGrowth, aes(x=group,y=weight)) + geom_boxplot()

p +ylim(0,max(PlantGrowth$weight))

Another method:

p + scale_y_continuous(limits=c(0, 10),breaks=NULL)

Inshort:

xlim()/ylim()

scale_x_continuous()/scale_y_continuous()

Question 4

Write down two ways to remove x-axis Labels and discuss their difference.

**Answer:**
Method1:

–For the x-axis label, use theme(axis.title.x=element_blank())
–For the y-axis label, do the same with axis.title.y
•p <-ggplot(PlantGrowth,aes(x=group, y=weight)) + geom_boxplot()
•p + theme(axis.title.x=element_blank())

Method2

Also set axis labels to an empty string as xlab(" ") / ylab(" ")
Difference

When you use theme() to set axis.title.x=element_blank(), the name of the x or y scale is unchanged, but the text is not displayed and no space is reserved for it.
•When you set the label to "", the name of the scale is changed and the (empty) text does display.

Question 5

Write down the functions to hide grid lines

**Answer:**
–The major grid lines

those that align with the tick marks
controlled with panel.grid.major
–The minor grid lines

the ones between the major lines
controlled with panel.grid.minor
p <-ggplot(heightweight,aes(x=ageYear,y=heightIn)) + geom_point()

p + theme(panel.grid.major =element_blank(), panel.grid.minor =element_blank())

It's possible to hide just the vertical or horizontal grid lines.Example: # Hide the vertical grid lines (which intersect with the x-axis)
p + theme(panel.grid.major.x = element_blank(), panel.grid.minor.x = element_blank())
# Hide the horizontal grid lines (which intersect with the y-axis)
p + theme(panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank())

Question 6

Write down two basic functions to add text annotations in a graph.
**Answer:**
–Use annotate() and a text geom
•

p <-ggplot(heightweight,aes(x=ageYear,y=heightIn)) + geom_point()

•p + annotate("text",x=3,y=48,label="Group 1") +
annotate("text",x=4.5,y=66,label="Group 2")

–p + geom_text(aes(label=weightLb),size=4, family="Times",colour="red")

With Mathematical Notation

annotate()

geom_text()
–Use annotate(geom="text") and set parse=TRUE
•p <-ggplot(data.frame(x=c(-3,3)),aes(x=x)) + stat_function(fun =dnorm)
•p +annotate("text", x=2,y=0.3,parse=TRUE, label="frac(1, sqrt(2 * pi)) * e ^ {-x^2 / 2}")

Question 7

Write down the functions to hide just the vertical grid lines
**Answer:**
p <-ggplot(heightweight,aes(x=ageYear,y=heightIn)) + geom_point()

For the above plot, vertical grid lines can be hid in the following way:

# Hide the vertical grid lines (which intersect with the x-axis)
p + theme(panel.grid.major.x = element_blank(), panel.grid.minor.x = element_blank())

—----------

Question 1

Write down the graph package's name in Python from our slides and how to load the package in Python.
**Answer**:
The graph package's name in Python:

matplotlib

It can be loaded in Python as:

import matplotlib.pyplot as plt

Question 2

Write down basic functions to load graph package and plot a horizontal bar chart, a pie chart, and histogram in Python.

**Answer:**

The graph package's name in Python:

matplotlib which has to be installed firstly for loading it as and when required.

Basic functions to load graph package:

```
import matplotlib.pyplot as plt
```

Plot of a horizontal bar chart, a pie chart, and histogram in Python:

Horizontal bars: barh() function
–equivalent of bar()

–giving horizontal rather than vertical bars

```
import matplotlib.pyplot as plt

data = [5., 25., 50., 20.]

plt.barh(range(len(data)), data)

plt.show()
```

Piechart:

To compare the relative importance of quantities, use pie chart
pyplot.pie() function takes a list of values as the input.
Note that the input data is a list or a NumPy array.
it will automatically compute the relative areas of the pie chart.

```
import matplotlib.pyplot as plt

data = [5, 25, 50, 20]

plt.pie(data)

plt.show()
```

Plotting histograms:

A histogram is just a specific kind of a bar chart. We could easily use matplotlib's bar chart function and do some statistics to generate histograms.
Example:

The following script draws 1000 values from a normal distribution and then generates histograms with 20 bins:

```
import numpy as np

import matplotlib.pyplot as plt

X = np.random.randn(1000)

plt.hist(X, bins = 20)

plt.show()
```

The pyplot.hist() function takes a list of values as the input.
The range of the values will be divided into equal-sized bins (10 bins by default).
Generate a bar chart, one bar for one bin.
The height of one bar is the number of values following in the corresponding bin.
By setting the optional parameter normed to True, the bar height is normalized and the sum of all bar heights is equal to 1.

Question 3

Write down the steps to load package and function to read a txt file into an array in Python.

**Answer:**
1) Firstly install the required package:

In ubuntu numpy can be installed as sudo apt install numpy/pip install numpy

2) Load the installed required package : import numpy as np

3) Write down a function to read a text file into an array in python in python script using a notepad text file or any IDE that supports python,for example even it can be done in google colab notebook.

```
data_array = np.loadtxt('my_data.txt')
```

#We can use delimiters too while reading from the text file as follows:

```
data_array = np.genfromtxt('data.txt', delimiter=',')
```

4) Access the array: by printing

```
print(data_array)
```

Summary:

```
import numpy as np

data_array = np.loadtxt('my_data.txt')

print(data_array)
```

Question 4

Write down the steps to load package and function to generate random 2 dimensional numbers in Python.

**Answer:**
The steps to load package and function to generate random 2 dimensional numbers in Python:

1) install and load the package

2) Required code function

3) Access the data of the code

```
import numpy as np #install and load the package
```

```
random_2Dnum = np.random.rand(10, 2) # Required code function

print(random_2Dnum) #Access the data of the code
```

—----------THE END—----------