ASSIGNMENT 0 - SRT411

Mahdy Aryamand | Student No : 011-913-076

May 2, 2016

Introduction

The assignment 0 includes all the different ToDos in the Short Introduction to R. I am using knitr to produce the Rmd document which I will be naming SRT411A0.Rmd as specified. Below are the 15 ToDos as required.

ToDo 1

Question: Compute the difference between 2014 and the year you started at this university and divide this by the Difference between 2014 and the year you were born. Multiply this with 100 to get the percentage of your life you have spent at this university. Use brackets if you need them.

Answer: I was born in 1983 and we are currently in 2016. I joined IFS program at Seneca in 2012. Hence the percentage of my life I have spent in college can be calculated as shown below:

```
((2016-2012)/(2016-1983))*100
```

```
## [1] 12.12121
```

As shown above, I have spent 12.1 % of my life at college.

ToDo 2

Question : Repeat the previous ToDo, but with several steps in between. You can give the variables any name you want, but the name has to start with a letter.

Answer : We have now been requested to make use of variables to calculate the percentage of life spent at college. It is exactly as the previous one but just that this time I will be using variables to save each output individually. The results are the same as shown below.

```
CollegeYears = (2016-2012) #Numbers of years spent at college
Age = (2016-1983) #My current Age
Percentage = ((CollegeYears/Age) * 100) #Saving percentage spent at college as Percentage
Percentage #Calling the variable Percentage
```

```
## [1] 12.12121
```

ToDo 3

Question : Compute the sum of 4, 5, 8 and 11 by first combining them into a vector and then using the function sum.

Answer : In this step we are required to create a vector with the numbers 4, 5, 8 and 11. Once the vector is created, we will proceed to calculate the sum of the vector. We will make use of a variable to make our task easier.

```
vector = c(4, 5, 8, 11) #saving the vector as a variable called vector
sum(vector) #calculating the sum of the vector
```

```
## [1] 28
```

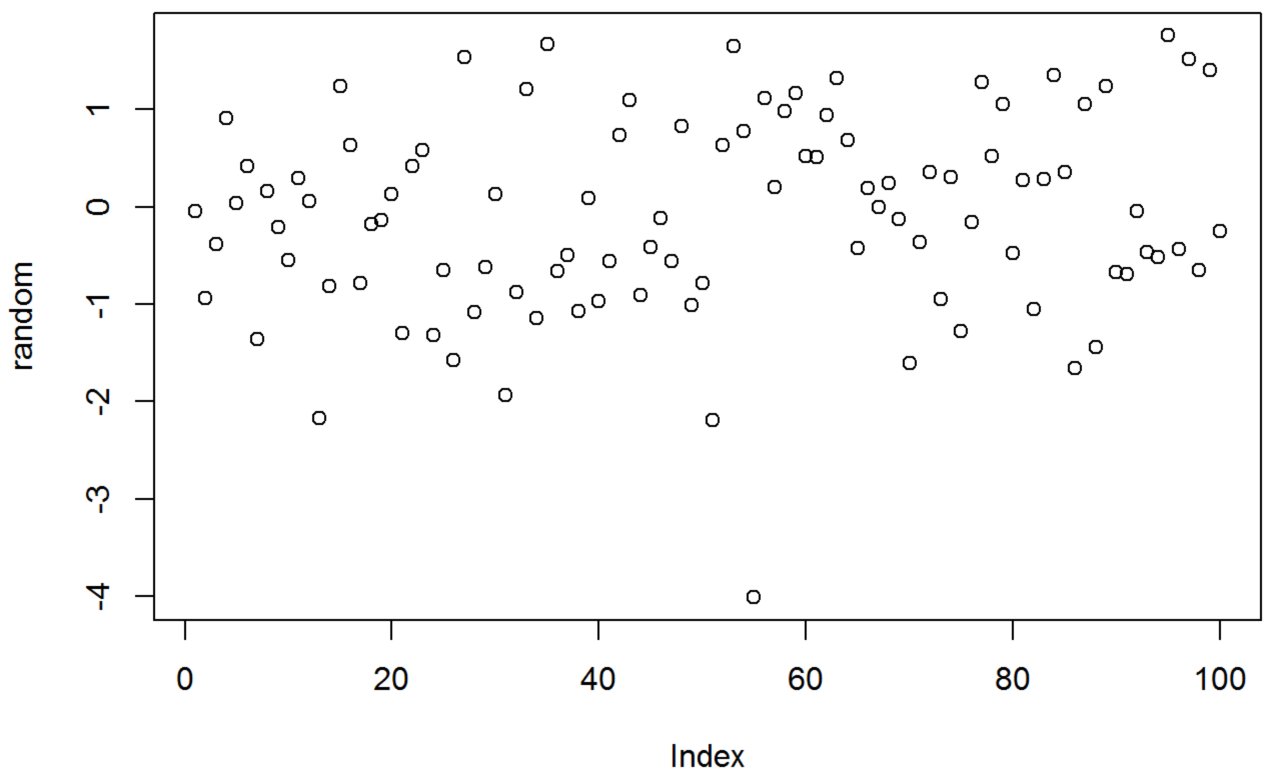
The sum is as shown above which gives us a result of 28 as expected.

ToDo 4

Question : Plot 100 normal random numbers.

Answer : The rnorm function can be used to generate random numbers. We will be using this function to generate 100 random numbers and use the plot function to plot the numbers on a graph. The code is as shown below.

```
random = rnorm(100) #saving 100 numbers to a variable called random
plot(random) # plotting the 100 numbers
```



ToDo 5

Question : Find help for the sqrt function.

Answer : The code is as shown below.

```
help(sqrt)
```

```
## starting httpd help server ...
```

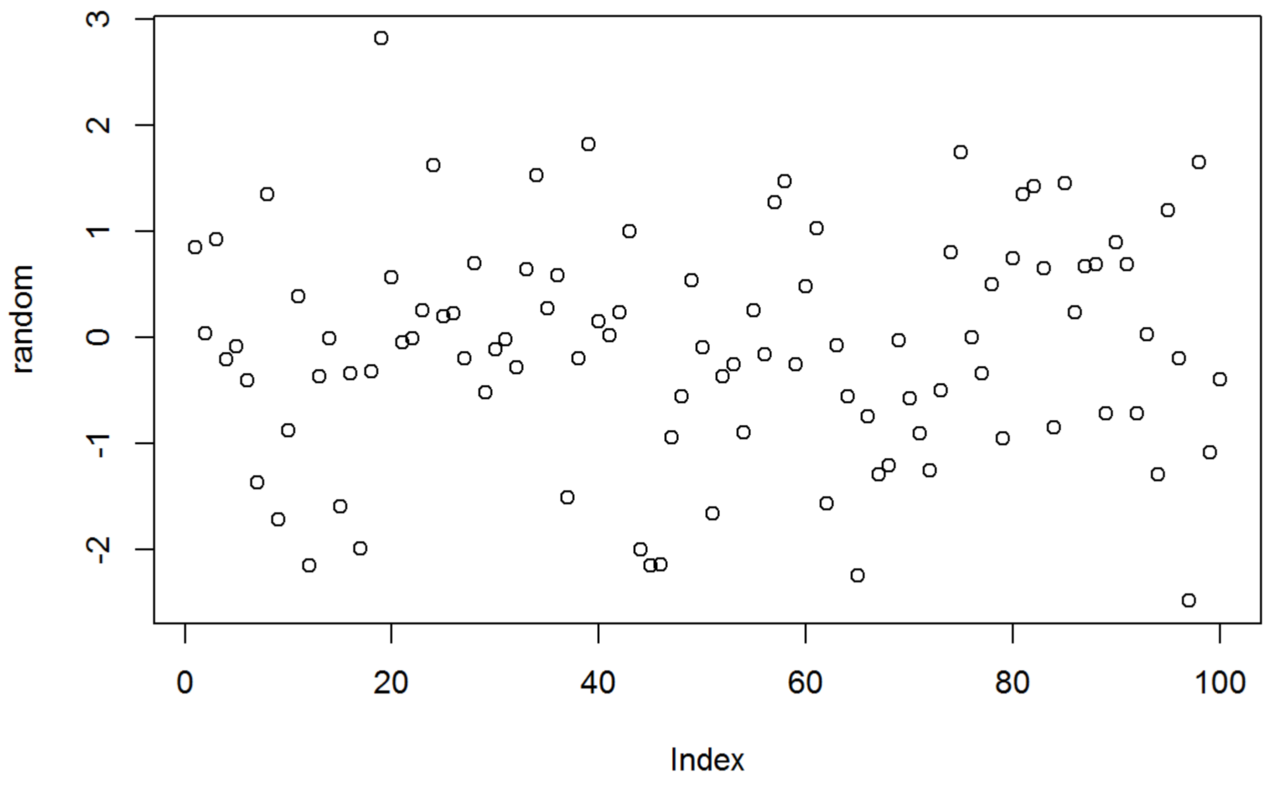
```
##  done
```

ToDo 6

Question : Make a file called firstscript.R containing R-code that generates 100 random numbers and plots them, and run this script several times.

Answer : Here we will be saving the code in ToDo 4 as firstscript.R. We will then call then anytime we want. The code used to call the script is as shown below.

```
source("firstscript.R")
```



ToDo 7

Question : Put the numbers 31 to 60 in a vector named P and in a matrix with 6 rows and 5 columns named Q. Tip: use the function seq. Look at the different ways scalars, vectors and matrices are denoted in the workspace window.

Answer : The code is as shown below.

```
Q = matrix((P = seq(from=31, to=60, by=1)), nrow=6, ncol=5)
Q # Printing the matrix
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   31   37   43   49   55
## [2,]   32   38   44   50   56
## [3,]   33   39   45   51   57
## [4,]   34   40   46   52   58
## [5,]   35   41   47   53   59
## [6,]   36   42   48   54   60
```

ToDo 8

Question : Make a script le which constructs three random normal vectors of length 100. Call these vectors x1, x2 and x3. Make a data frame called t with three columns (called a, b and c) containing respectively x1, x1+x2 and x1+x2+x3. Call the following functions for this data frame: plot(t) and sd(t). Can you understand the results? Rerun this script a few times.

Answer : I will create 3 vectors each of length 100 and name them x1, x2 and x3 respectively. I will then create a data frame with a (a = x1), b (b = x1 + x2) and c (c = x1 + x2 + x3). The data frame will be saved as a variable called ‘t’. I will proceed to plot ‘t’ and calculate the standard deviation.

```
#saving 3 vectors x1, x2 and x3 of length 100
x1 = c(rnorm(100))
x2 = c(rnorm(100))
x3 = c(rnorm(100))
#Calling vectors create
x1 #calling vector x1
```

```
##   [1]  0.10726770 -0.36272270 -0.76327852  0.35005693  0.64132585
##   [6]  0.58113579 -0.19259826  1.04883152 -0.38062720 -1.12117939
##  [11] -0.79608878  0.17223672  0.59549367  0.01584443  0.17285109
##  [16] -0.81599409  0.46306492 -1.50672539 -0.65760637 -0.60064561
##  [21] -1.93133765  1.68517093  0.48513620 -0.20557318  0.47655910
##  [26]  0.36463040 -0.47879867  1.27193786  0.08921363 -1.45205305
##  [31] -1.16862621 -1.34601857 -0.33709852  1.05653275 -0.33080858
##  [36] -0.05567940 -0.28624212 -2.03443867  1.29626469  0.67889977
##  [41] -0.13862517 -0.62546488 -0.17493607 -0.36828372 -0.24501792
##  [46] -0.51263724  0.07413560 -0.72548583  0.45055402 -0.11718583
##  [51]  0.79844173  0.70526135  0.33505906 -0.75006499 -0.56925174
##  [56] -0.32214328  0.21654797  0.62116176 -1.59043142  0.02533881
##  [61] -0.85392618  1.55333928  2.54418761 -1.09793069 -0.01598921
##  [66] -0.25556795 -0.15759620  0.78775372  0.35461557 -2.12385241
##  [71]  0.50898279 -0.21119925  1.05772902  0.18381157  0.25745573
##  [76]  0.60743961 -0.20341604  0.40564612  0.61795857 -0.27112707
##  [81] -0.05737130 -0.25310971 -0.98776063 -0.04591504  0.03715361
##  [86]  0.97221176 -0.97890707  0.99097125  0.32918101 -1.12770501
##  [91] -1.25367646 -0.56139964 -1.14698833  0.63974477  0.66881273
##  [96]  0.39373408  0.56611630 -0.38785445  0.06917210  0.87164887
```

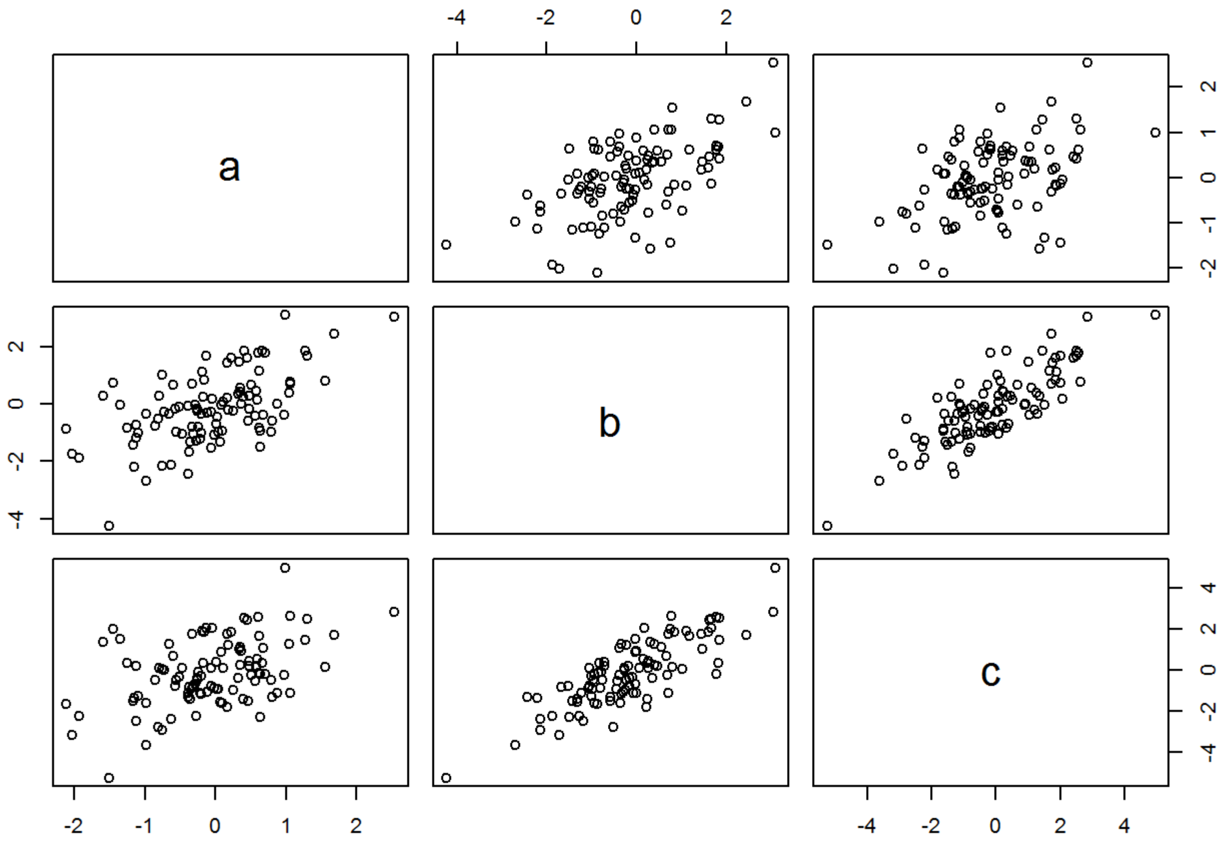
```
x2 #calling vector x2
```

```
##   [1] -0.03971317 -0.95826247 -1.37771517  0.19870590 -1.57508260
##   [6] -0.11530859  1.29090348 -0.65881380 -2.05290442 -0.06155523
##  [11]  1.06608405  1.27226148 -0.44602953 -0.72333400  0.03866094
##  [16]  0.29106622 -1.05216382 -2.72810258  0.32052933  1.25605350
##  [21]  0.04999581  0.76219778 -0.21662997 -0.79929109 -0.65850457
##  [26] -0.37396416 -0.58057906  0.56911586 -1.03074754  2.19589715
##  [31] -0.26175593  1.31456274 -0.46616523 -0.35995900  1.02834450
##  [36] -0.21423324  0.26171169  0.30744028  0.36297514 -1.05578563
##  [41]  1.79839836 -1.50826120  0.42079619 -1.31827721 -0.54844329
##  [46]  0.41508367 -0.10372842  0.46098326  1.16872727 -0.18574594
##  [51] -1.37599696  1.06584019  1.13367100  1.76660998  0.39927811
##  [56] -0.73021879  1.38032571  0.54515943  1.88615854 -0.48000746
##  [61]  0.08770830 -0.76009566  0.49864056  0.09508322 -1.05735162
##  [66]  0.02201497  0.99669419 -1.75165686  0.05140699  1.25399790
##  [71]  0.16725313 -1.02092255 -0.27790525 -0.40210052 -0.51224448
##  [76]  1.15805767 -0.14015044  1.43897261 -1.46615846 -1.00608544
##  [81] -1.47189027  0.08992799  0.63555227  0.22567980 -1.02564199
##  [86] -1.35481840 -1.71349208  2.09628426  0.02185970  0.40462363
##  [91]  0.41876136 -0.40068109 -1.05488715 -2.13573404  1.16059611
##  [96] -0.15830576 -0.99611908  0.32556564 -1.39077562 -0.86777044
```

```
x3 #calling vector x3
```

```
##   [1]  0.01893484 -0.07380717 -0.77841218  0.55373794  0.73700386
##   [6] -0.26954343  0.79405068  0.88550599  1.13969671 -1.31845192
##  [11] -0.18544373  0.30219719  0.37946530  1.10680197 -2.02640797
##  [16] -2.25321470 -0.89976210 -1.01718235  1.62379957  0.02239676
##  [21] -0.33438548 -0.72909617  0.17465907  0.69025816  0.38486311
##  [26]  0.93328232  1.14020076 -0.39156545 -0.68406215  1.25113263
##  [31] -0.07900029  1.53300258 -0.09103555 -1.82923422  1.03765493
##  [36]  0.36367489 -0.64805776 -1.44373188  0.83336964  1.42539210
##  [41]  0.36104982 -0.25361297  0.11360502  0.84444631  0.67341011
##  [46] -0.25789762  0.92611684  0.28512341  0.80536449 -0.74865294
##  [51] -0.71036790 -1.94108902 -0.45092201 -0.96884782 -0.62448144
##  [56]  0.25900433  0.27060665  0.50160395  1.06269413 -0.48515478
##  [61]  0.27334035 -0.65920694 -0.20002551 -0.26896094  0.18073870
##  [66] -0.22848802  1.01057975  0.47379231 -0.17237688 -0.76271292
##  [71] -0.93401851  0.09998365  1.84362222  1.42902187 -0.73566256
##  [76]  0.79969477 -0.83867038  0.67596763  0.69328745 -0.93716812
##  [81]  0.74353237 -0.24437036 -1.25319171  1.88228097  0.06849142
##  [86]  0.12207530 -0.93026857  1.88386133 -0.73471350  0.91478348
##  [91]  1.17180428  0.49308955  0.83731427 -0.79289109 -1.50404786
##  [96] -1.62498443 -0.11789569 -1.04047393 -0.23675748 -1.12859148
```

```
t = data.frame(a = x1, b = x1+x2, c = x1+x2+x3) #creating the data frame
plot(t) #plotting the graph
```

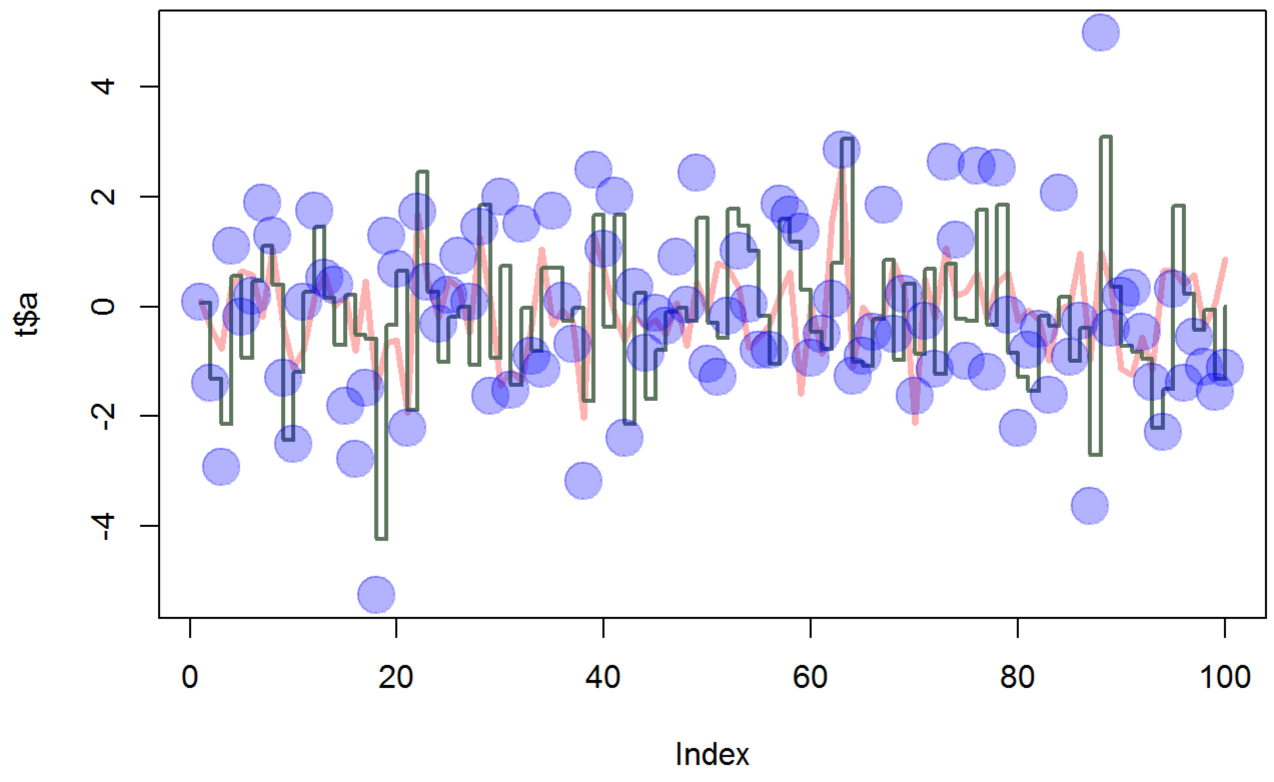


ToDo 9

Question : Add these lines to the script file of the previous section. Try to find out, either by experimenting or by using the help, what the meaning is of rgb, the last argument of rgb, lwd, pch, cex.

Answer : The following lines will be added the previous script.

```
plot(t$a, type="l", ylim=range(t),
lwd=3, col=rgb(1,0,0,0.3))
lines(t$b, type="s", lwd=2,
col=rgb(0.3,0.4,0.3,0.9))
points(t$c, pch=20, cex=4,
col=rgb(0,0,1,0.3))
```



Explanation:

- rgb : This stands for Red, Green and Blue. By changing the number of the 3 different values, I can create the colour of my choice and can even change the intensity of the graph with the last number.
- lwd : This represents the thickness of the line which we will be using to plot the graph. We can increase the thickness by increasing the value.
- pch : This is the sign of the plot we use to plot graphs. Different numbers represent different shapes including squares, triangles, cross etc.
- cex : This is the size of the coordinates being used. We can make the coordinates bigger by increasing the number.

ToDo 10

Question : Make a file called tst1.txt in Notepad from the example in Figure 4 and store it in your working directory. Write a script to read it, to multiply the column called g by 5 and to store it as tst2.txt.

Answer : The code is as shown below.

```
dataset = data.frame(a = c(1,2,4,8,16,32),
               g = c(2,4,8,16,32,64),
               x = c(3,6,12,24,48,96))
write.table(dataset,file="tst1.txt", row.names=FALSE) #saving dataset to tst1.txt
read.table(file="tst1.txt", header=TRUE) #reading the file
```

```
##    a  g  x
## 1  1  2  3
## 2  2  4  6
## 3  4  8 12
## 4  8 16 24
## 5 16 32 48
## 6 32 64 96
```

We can now multiply the g column by 5 and save the result to a file named tst2.txt. The code is as follows:

```
dataset[2] = dataset[2]*5 #multiply the column g by 5
write.table(dataset,file="tst2.txt", row.name=FALSE) #saving result to tst2.txt
```

ToDo 11

Question : Compute the mean of the square root of a vector of 100 random numbers. What happens?

Answer : The code and results is as shown below. As it can be seen NaNs are produced which means not a valid number. It usually represents an imaginary number.

```
sqrt(rnorm(100))
```

```
## Warning in sqrt(rnorm(100)): NaNs produced
```

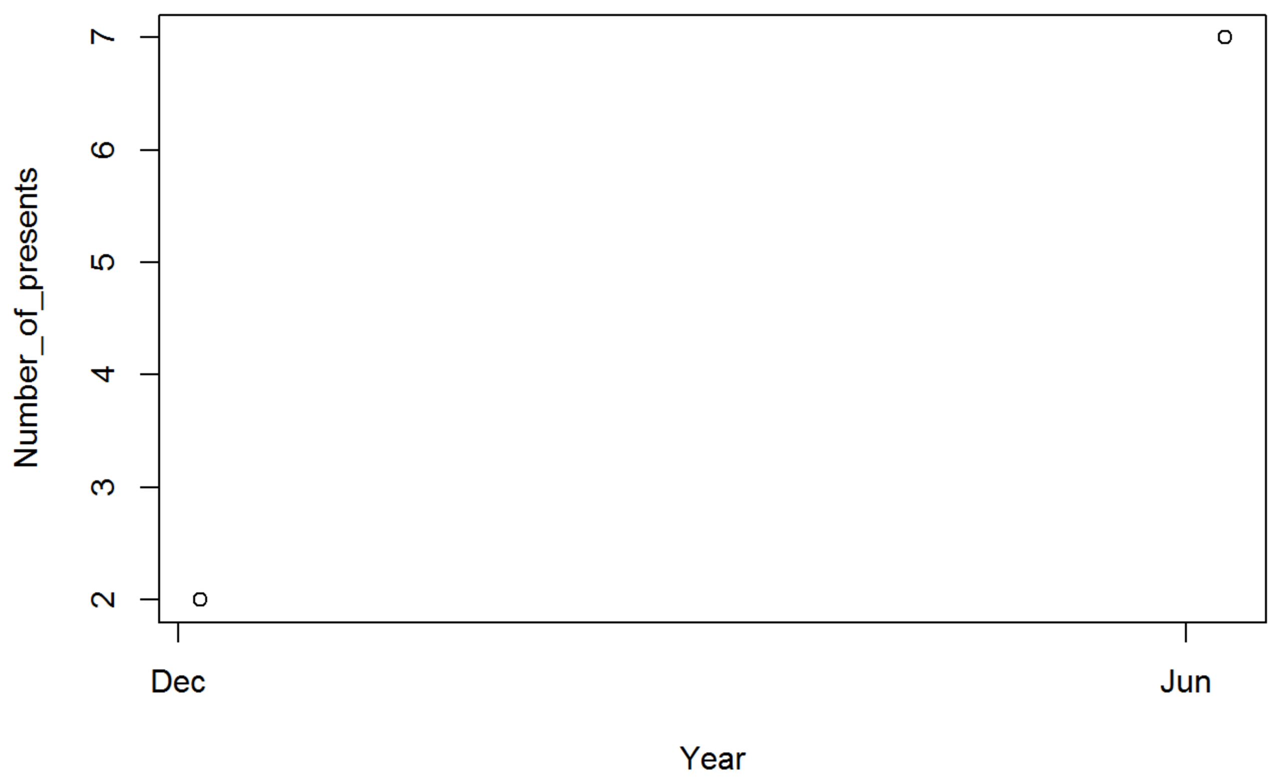
```
##   [1] 0.2862224       NaN       NaN       NaN       NaN       NaN 0.2950465
##   [8] 1.3365755       NaN       NaN       NaN       NaN 0.4514541       NaN
##  [15]       NaN 0.3239132       NaN 0.8055378 0.3787708       NaN       NaN
##  [22]       NaN 0.6961868       NaN       NaN 0.8706010       NaN       NaN
##  [29]       NaN 1.0380623 1.0085152       NaN 0.6064411       NaN       NaN
##  [36] 0.4348952       NaN 1.2814164       NaN 0.2093602       NaN 0.4168109
##  [43] 1.1147413 0.6907288 0.7100368       NaN       NaN       NaN       NaN
##  [50] 0.6258258 0.9692077       NaN       NaN 0.6091442       NaN 1.5396034
##  [57]       NaN       NaN 0.6952502 1.1078844 0.5988214       NaN 1.0452950
##  [64]       NaN 0.5594336 0.3372955 0.4867674 0.4414295       NaN       NaN
##  [71] 0.5531264       NaN 0.5909807       NaN 1.0391725       NaN 0.7516892
##  [78] 0.4476838 1.2877874 0.7803969       NaN       NaN 0.3425599 1.3451943
##  [85]       NaN       NaN 1.2286903       NaN 1.2100940 0.4076529 0.3488852
##  [92]       NaN       NaN       NaN 1.4847141 0.5618213 0.6723230 1.0639477
##  [99]       NaN 1.0035622
```

ToDo 12

Question : Make a graph with on the x-axis: today, Sinterklaas 2014 and your next birthday and on the y-axis the number of presents you expect on each of these days. Tip: make two vectors first.

Answer : As requested we will create 2 vectors first. The code is as follows:

```
Year = strptime(c("20141205", "20150608"),format="%Y%m%d")
Number_of_presents = c(2,7)
plot(Year, Number_of_presents)
```



ToDo 13

Question : Make a vector from 1 to 100. Make a for-loop which runs through the whole vector. Multiply the elements which are smaller than 5 and larger than 90 with 10 and the other elements with 0.1.

Answer : We will be using a for loop as follows.

```
vector = seq(from=1, to=100) # creating a vector with numbers from 1-100
for (i in 1:100)
{
  if (vector[i] < 5 | vector[i] > 90)
  {
    vector[i] = vector[i] * 10
  }
  else
  {
    vector[i] = vector[i] * 0.1
  }
}
```

ToDo 14

Question : Write a function for the previous ToDo, so that you can feed it any vector you like (as argument). Use a for-loop in the function to do the computation with each ele- ment. Use the standard R function length in the specification of the counter.

Answer : This can be achieved by using the length function in r. The code is as shown below.

```
vector = function(argument)
{
  for (i in 1:length(argument))
  {
    if (argument[i] < 5 | argument[i] > 90)
    {
      argument[i] = argument[i] * 10
    }
  else
    {
      argument[i] = argument[i] * 0.1
    }
  }
}
```

We can now pass any argument we want!

ToDo 15

Question : Achieve the previus ToDo without a for-loop.

Answer : This can be achieved with the following code.

```
vector = c(1:100)
ifelse(vector < 5 | vector > 90, vector * 10, vector * 0.1)
```

```
##   [1]   10.0   20.0   30.0   40.0    0.5    0.6    0.7    0.8    0.9    1.0
##  [11]    1.1    1.2    1.3    1.4    1.5    1.6    1.7    1.8    1.9    2.0
##  [21]    2.1    2.2    2.3    2.4    2.5    2.6    2.7    2.8    2.9    3.0
##  [31]    3.1    3.2    3.3    3.4    3.5    3.6    3.7    3.8    3.9    4.0
##  [41]    4.1    4.2    4.3    4.4    4.5    4.6    4.7    4.8    4.9    5.0
##  [51]    5.1    5.2    5.3    5.4    5.5    5.6    5.7    5.8    5.9    6.0
##  [61]    6.1    6.2    6.3    6.4    6.5    6.6    6.7    6.8    6.9    7.0
##  [71]    7.1    7.2    7.3    7.4    7.5    7.6    7.7    7.8    7.9    8.0
##  [81]    8.1    8.2    8.3    8.4    8.5    8.6    8.7    8.8    8.9    9.0
##  [91]  910.0  920.0  930.0  940.0  950.0  960.0  970.0  980.0  990.0 1000.0
```