SRT411A0

Pol Govergun

2/15/2018

This assignment is a short task to complete as an introduction to the R programming language, it has 14 todo examples that I have answered below, plus one last todo in the footnote of this doc "https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf". The assignment starts with 3 sections that describe how to download and install R and Rstudio, both resources are necessary to learning how to code with R. The first two sections describe advantages of R as a freeware and why it is a great resource for source coding. Furthermore, in the getting started section we learn how to configure and install Rstudio along with a breif toutorial on the layout of the console and how to navigate to workspace mode, files and plots and the editor. Additionally, in sections 2.4 and 2.5, we get a breif intro into setting working directories and installing library packages.

3 some first examples of R commands

3.1 Calculator

ToDo: Computing the difference between 2014 and the year you started at this university (2015), divided by the difference between 2014 ad the year I was born (1997) and then muliply the result by 100, to get the percentage of your life youve spent in college.

```
(2015 - 2014) / (2014 - 1997) * 100
```

```
## [1] 5.882353
```

3.2 Workspace

ToDo: This todo is the same as the previous, except we add variables to some of the values we calculated.

```
a <- 2015 - 2014
b <- 2014 - 1997
c <- 100

a / b * c
```

```
## [1] 5.882353
```

3.3 Scalars, vectors and matrices

This section explains a bit about scalar, vector and matrice values. A scalar is a single value like A assigned to 'c' in the pervious todo. A matrice is like a table with 2 dimensional values and lastly a vector is a one dimentional structure that concatenates any defined values in the function.

3.4 Functions

This section teaches us about how to create vectors and use functions, it gives a useful exmaple using the 'rnorm' function, which generates a random sample from a normal distribution, which you must specify, ex rnorm(10, mean=1.2). This command will generate 10 random numbers with a mean of 1.2. Furthermore, the resulting 10 numbers will automatically become a vector.

Todo: Compute the sum of the vector 4,5,8 and 11, by first creating a vector and using the sum function.

```r
d <- c(4,5,8,11)
sum(d)
```

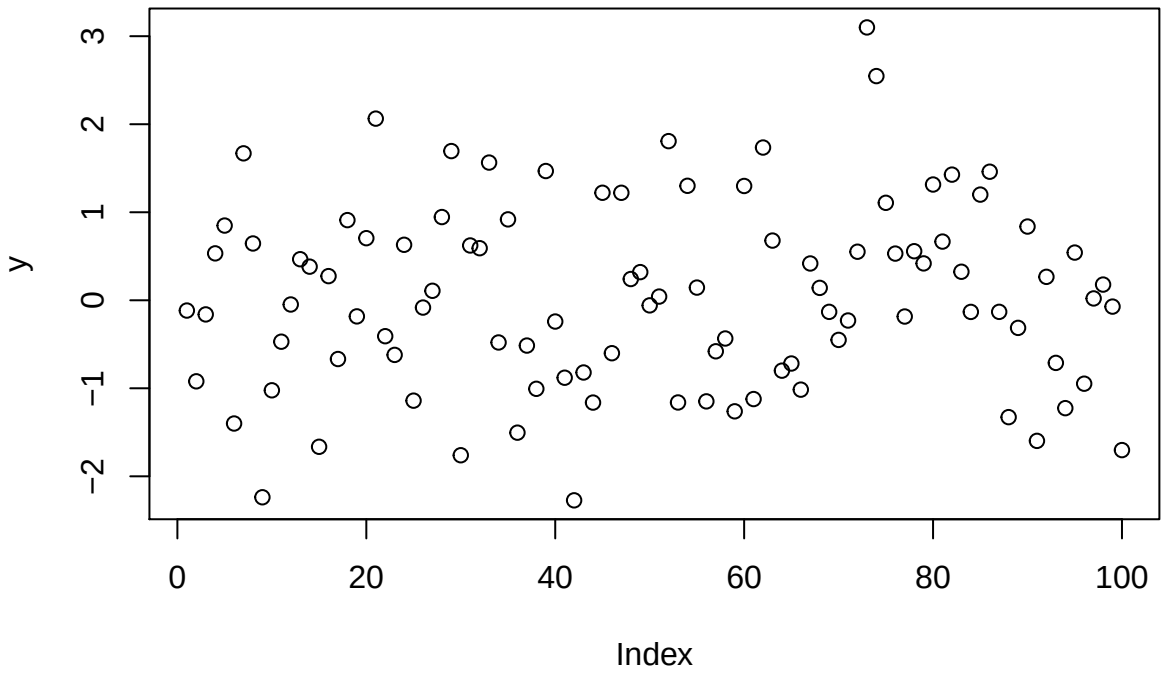
```
## [1] 28
```

3.5 Plots

This section teaches us about how to create a simple graph and plot 100 random numbers on this graph, with the help of the rnorm function used previously.

ToDo: Plot 100 normal random numbers.

```r
y <- rnorm(100)
plot(y)
```



4 Help and Documentation

This section teaches how a user can get help with any command in Rstudio, using the help(), function you can get a simple example and guidlines of how to use the function. Furthermore, for online web browsing help you can use the help.start() function.

ToDo: Find help for the sqrt function.

```r
help(sqrt)
```

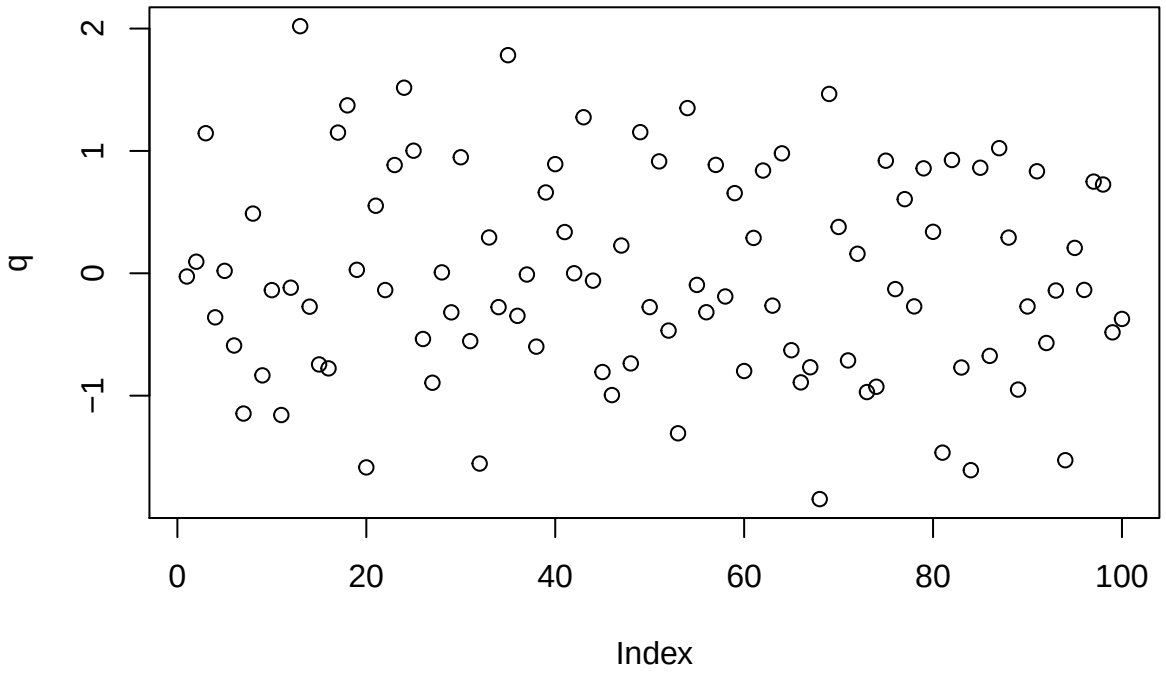
5 Scripts

This section describes the use of scripts in R as collection of multiple R commands stored in one file, usually with a .R extension. There are multiple ways to run your scripts, you can use the Rstudio windows or type CTRL + Shift + S or CTRL + ENTER.

ToDo: Make a file called firstscript.R containing code that generates 100 random numbers and plots them.

Firstly I typed CTRL + SHIFT + N, to create a new script file named firstscript.R and saved it in my documents. Then to run the script I typed 'source("firstscript.R")'.

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



6 Data Structures

In the 6.1 section, this guide describes vectors in more detail, it gives diffrent ways to create vectors, such as using the seq() function and displays different calculations that are common amongst vectors.

The 6.2 section teaches how matrices work and how to construct them.

ToDo: Put the numbers 31 to 60 in a vector named P and then in a matrix with 6 rows and 5 columns named Q.

```
P <- seq(from=31, to=60)
P
```

```
##  [1] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
## [24] 54 55 56 57 58 59 60
```

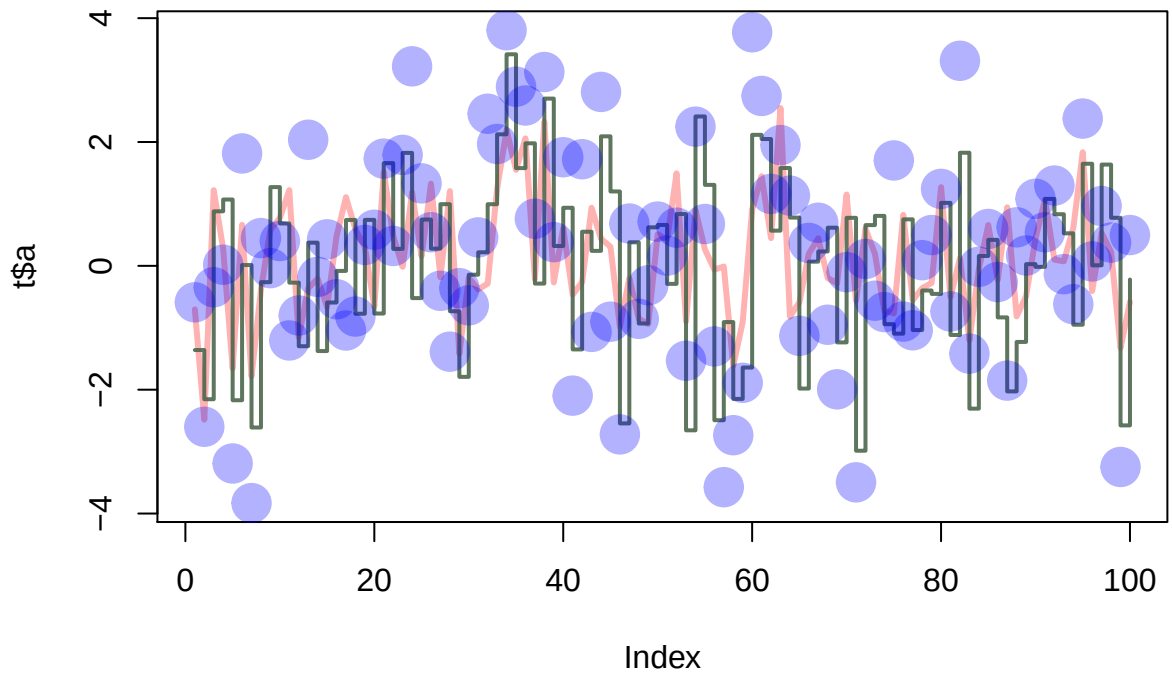
```
Q <- matrix(data=P,ncol=5,nrow=6)
Q
```

```
##      [,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
```

The next section 6.3, teaches about data frames and how to use them in combination with matrices and how to create and query their values.

ToDo: Make a script file which constructs three random vectors of length 100. Then make a data frame called t with three columns. Then apply the functions plot(t) and sd(t) and examine the results.

```
source("dataframescript.R")
```



The next line displays the code contained in the script dataframescript.R.

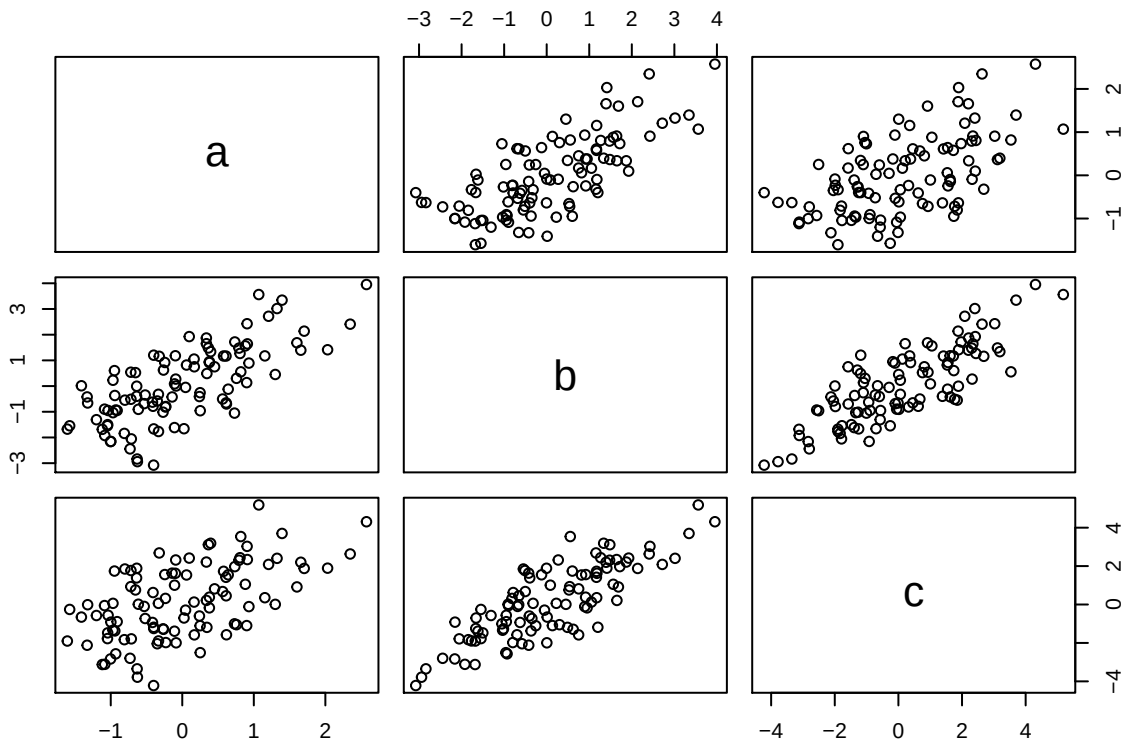
```
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
t = data.frame(a = x1, b = x1+x2, c = x1+x2+x3)
t
```

```
##               a           b           c
## 1   -0.96852932  0.22630954  0.06130954
## 2   -0.26860878 -1.01646125 -1.26688203
## 3   -0.22996574 -0.79644444 -1.98110895
## 4    1.29998804  0.45123687  0.00867821
## 5    2.03141381  1.41320821  1.89252219
## 6    0.56402347 -0.50342360  0.67481231
## 7   -1.11808906 -1.68304146 -3.12408408
## 8   -0.09653852  1.17963734  1.62488787
## 9   -1.04269789 -1.50068911 -1.47697294
## 10  -0.71902534  0.53958868  0.93052437
## 11  -0.80210660 -0.54780926  1.84878366
## 12   0.80738272  1.26896321  2.43635448
## 13   0.90622923  2.42602525  3.02781986
## 14  -1.60647951 -1.67627154 -1.90067312
## 15  -0.93037351 -0.93105132 -2.56841213
## 16   2.57501734  3.95234717  4.30833114
## 17  -0.09157339  0.27167055  2.31751219
## 18  -0.41650484 -0.62547318 -0.93377569
## 19  -0.10792872  0.08569664  1.00612378
## 20  -0.71722420 -0.51285280  1.76903121
## 21   0.33732620  1.64979546  0.21325994
## 22   0.36532276  1.48355441  3.11420774
```

```
## 23  -0.51854157 -0.347940882 -0.728443201
## 24  -0.65382170  0.524076313  0.757159906
## 25   0.09748199  1.925666065  2.416156780
## 26   1.60264219  1.685138742  0.915294087
## 27  -0.39843065 -1.666387677 -1.249280846
## 28  -1.04549115 -1.543463798 -1.774644463
## 29   0.05934666  0.818358174  1.537089991
## 30   1.20648025  2.716592539  2.086920760
## 31  -0.53193822 -0.683246659 -0.098821643
## 32   2.34743441  2.409781737  2.630235284
## 33   0.04576509 -0.051766740 -0.292680732
## 34   0.37870531  0.914324649  0.381337843
## 35  -0.14175769 -0.423467420  1.632550139
## 36  -1.57387346 -1.545298370 -0.256249697
## 37   0.16519942  1.051155875  0.126714240
## 38  -0.40852301 -0.787073609  0.623515972
## 39   1.15430627  1.174226809  0.354102894
## 40   0.75770927  0.300657747 -1.049952698
## 41   0.02422710 -1.659413508 -0.701944461
## 42   0.24914387 -0.257346016 -1.104589632
## 43  -1.40996454  0.011713513 -0.654492611
## 44   0.37908746  0.949081238 -0.173976059
## 45  -0.34971701 -0.580416913 -2.043117649
## 46  -0.94785038  0.601828034  1.741534122
## 47   0.93185563  0.894043865 -0.109117061
## 48   0.73022059 -1.052107120 -1.005274755
## 49   0.61115286  1.168090913  1.416889470
## 50  -0.96804866 -1.035916433 -1.339471511
## 51  -1.00262315 -2.160252455 -2.833828065
## 52  -1.32954653 -0.422399973 -2.119788402
## 53   0.61287778 -0.653256092  0.452849524
## 54   0.23985598 -0.408068545 -0.598264239
## 55  -1.19993119 -1.307980671 -0.572413603
## 56  -1.32251833 -0.657821565 -0.011361285
## 57  -0.26270276  0.622378502 -1.289841961
## 58  -0.63689586 -0.004954154  1.885861417
## 59  -1.08652190 -0.897919829 -0.055431753
## 60   0.90043021  0.134137067 -1.098143426
## 61  -1.08317512 -1.922591313 -3.115130390
## 62   0.39647784  1.338837105  3.188535307
## 63  -0.08634128  0.003489423 -1.993844628
## 64  -0.10886883 -1.615605620 -1.389720024
## 65  -0.39936896  1.202057931 -1.183974294
## 66  -0.24456943  0.918602924  1.558950108
## 67   0.33694111  1.870195426  2.209828291
## 68  -0.33234516 -1.768783191 -1.896160437
## 69   0.25106736 -0.959500802 -2.506888538
## 70  -0.80983430 -1.842095711 -1.831026257
## 71   0.90987075  1.643010380  2.340405781
## 72  -0.23522414 -0.813824019  0.318187630
## 73   0.34416945  0.488679688 -1.189728182
## 74  -0.63091597 -2.838296784 -3.348454453
## 75  -0.33336821 -0.320162064  0.060743573
## 76   1.32491171  3.016443243  2.403869615
```

```
## 77  -1.03660078 -0.950446974 -0.561303987
## 78  -0.99802086 -2.154626960 -0.918172654
## 79   0.45151860  0.751008796  0.815555828
## 80  -0.61417253 -0.905537426  0.014543038
## 81   1.65711843  1.395502130  2.199571076
## 82   0.88145783  1.563973586  1.052207119
## 83   1.70438291  2.137528324  1.877423296
## 84  -0.71214073 -2.055554963 -1.787625504
## 85  -0.91012719 -0.949028447 -0.892031743
## 86  -0.32170570  1.158408373  2.683914568
## 87  -0.72971105 -2.445002273 -2.800288231
## 88  -0.94268158 -0.366830316 -1.375218722
## 89  -0.62713795 -2.941215324 -3.783031562
## 90   0.64210215 -0.120973596  1.546628058
## 91   0.73438037  1.717344006  1.971847156
## 92   0.57756474  1.170903437  1.727617839
## 93   0.81802689  0.553878952  3.538042936
## 94   0.79300645  1.477674025  2.305267775
## 95  -0.40062923 -3.076948779 -4.218747495
## 96  -0.63851529 -0.398743963  1.384347184
## 97   1.06929897  3.561933254  5.183526031
## 98   0.61741562 -0.697851666 -1.575545489
## 99   1.39484794  3.344456675  3.698780798
## 100  0.16825126  0.750476883 -1.580742251
```

```
plot(t)
```



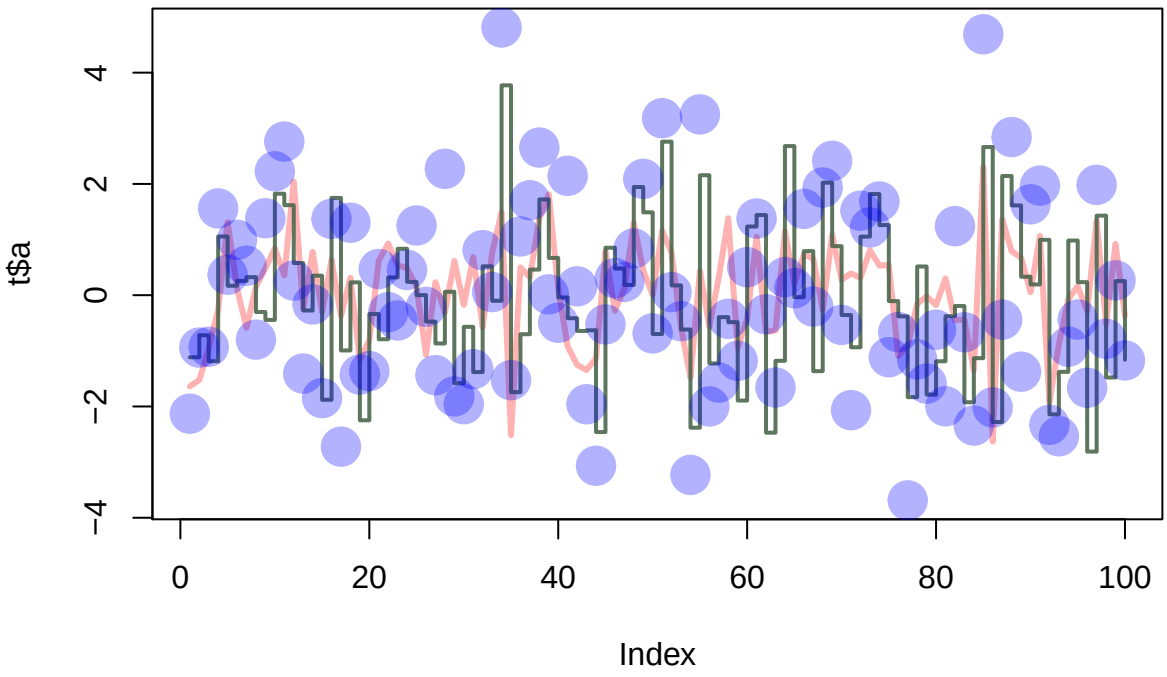
The next section 6.4, describes how to create a list using the list() function, how to print a list, how to find out whats inside your list and how to apply the values in your list. A list is more useful than a matrice or data frame because you can add as many columns as you perfer with no order or specific length.

7 Graphics

This section describes different ways to plot vectors and different graphical designs that can be made by R, anything from a histogram to a lined graph can be made.

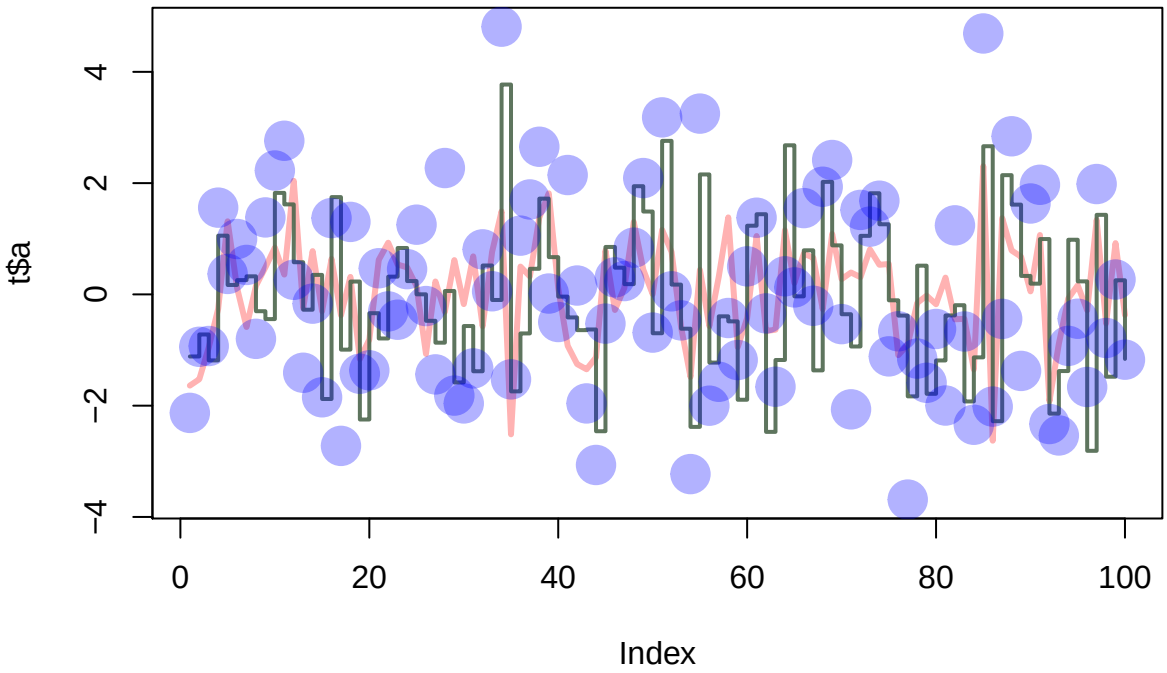
ToDo: Add the lines above to your script from the previous todo and figure out what the rgb, lwd, pch and cex arguments mean.

```
source("dataframescript.R")
```



The code that was added is visible below

```
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))
```



Essentially, the 'cex' argument is a number assigned to it that dictates the magnified level of text and symbols relative to the default. The 'lwd' argument specifies the line width, it must be a positive number. The 'pch' argument must be assigned an integer or a single character which is used as the default in plotting points. Lastly, the 'rgb' function is specifed when you want your graph to contain colors on the red, green and blue scale, you must also assign values of color to each column.

8 Reading and writing data files

This section specifies how to create files and interact with them using R, it describes how to read files and write into files using different read/write functions.

ToDo: Make a text file called tst1.txt 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.

command: source("readwrite8.R")

The contents of the script are show below

r = read.table(file="tst1", header = TRUE) r w = write.table(tst1$g * 5, file="tst2") e = read.table(file="tst2", header = TRUE) e

9 Not available data

ToDo: Compute the mean of the square root of a vector of 100 random numbers and see what happens

```
n <- rnorm(100)
s <- sqrt(n)
```

```
## Warning in sqrt(n): NaNs produced
```

```
mean(s)
```

```
## [1] NaN
```

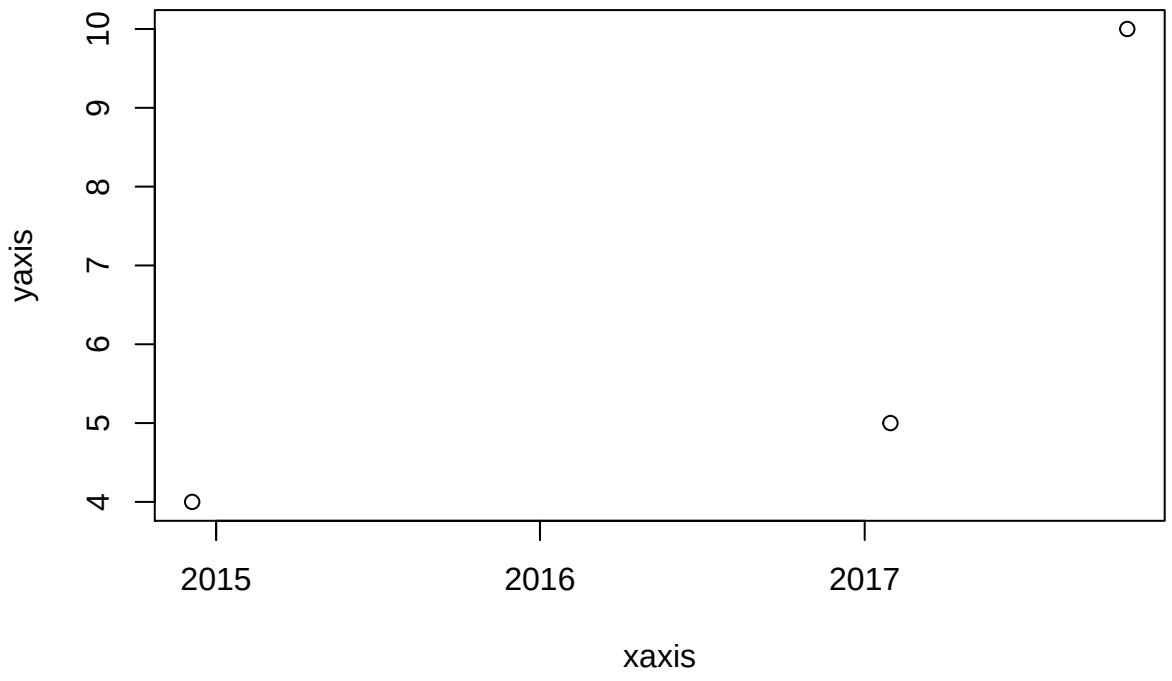
The output displays a series of non applicable numbers. When any non-applicable number is encountered it is displayed by the 'NA' characters. You can use the option 'na.rm=TRUE' to compute a non-applicable number without displaying the 'NA'

10 Classes

This section teaches about the different classes that R has, the three main ones are numbers, characters and POSIX, which is data and time.

ToDo: 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 to recieve on each day.

```
xaxis = strptime(c("20170130", "20141205", "20171024"), format="%Y%m%d")
yaxis = c(5, 4, 10)
plot(xaxis, yaxis)
```



11 Programming tools

In this section it is descibed how to create for-loops and if-else statments to either execture a command a certain amount of times or to set a conditional statements that does one thing if the condition is met and another if the condition is not met.

ToDo: 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 larer than 90 with 10 and the other elements with 0.1.

```
v = seq(from=1, to=100, by=1)
s = c()
for(i in 1:100)
{
  if(v[i] < 5)
  {
    s[i]=v[i]*10;
  }
  else if(v[i] > 90)
```

```
  {
    s[i]=v[i]*10;
  }
  else
  {
    s[i]=v[i]*0.1;
  }
}
s
```

```
##   [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
```

ToDo: Write a function for the previous todo, so that you can feed it any vector you like. Use a for-loop in the function to do the computation with each element. Use the standard R function length in the specification of the counter.

```
fun1 = function(arg1, arg2)
{
  v[i] = arg1[i];
  for(i in length(v))
  {
  }
}
```