Assignment 0 SRT411 Github: odamicostudent

The purpose of this assignment was to go through the todo sections in a short introduction to R at https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf

Section 2.4 Working Directory

Setting a working directory in R can be done using:

```
setwd("~")
```

where "~" can be replaced by the desired directory, you can also use tools –> Set working directory in RStudio

Section 2.5 Libraries

R, like any language, has many packages. These can be listed using:

```
library()
```

To install and use new packages, use:

```
install.packages("geometry")
library("geometry")
```

Section 3.1 Calculator

Equations can simply be typed to give answers as so:

```
10^2 + 36
```

```
## [1] 136
```

R will return your answer as seen above.

Todo task number 1

Percentage of my life spent at college:

```
(2017 - 2014) / (2014 - 1999) * 100
```

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

The equals operator can be used to assign values to variables, simply entering the variable name will tell R to print the value stored in the variable, you can use the variable name in equations and re-assign the values as needed:

```
a = 5
a
```

```
## [1] 5
```

```
(a * 3)^2
```

```
## [1] 225
```

```
a = a *20
a
```

```
## [1] 100
```

To remove all variables from R's memory, use rm:

```
rm(list=ls())
```

To remove specific variables, use their name(s) instead:

```
rm(a)
```

Todo task number 2

Repeating the percentage question using variables:

```
start = 2017
compare_date = 2014
born= 1999
(start - compare_date) / (compare_date - born) * 100
```

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

3.3 Scalars, vectors, and functions

Scalars are single values, vectors are a row of numbers like an array. Before, we created a scalar value, to create a vector, use the c(n1, n2, n3, … n) function:

```
a = c(1, 2, 3, 5, 1234, 3.1)
a
```

```
## [1]    1.0    2.0    3.0    5.0 1234.0    3.1
```

to use the ‘mean’ function we use mean(n) where n is the vector to use.

```
mean(a)
```

```
## [1] 208.0167
```

Todo task number 3

```
sum(4, 5, 8, 11)
```

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

Todo task number 4

R lets you create random normal distributions using rnorm()

```
rnorm(100)
```

```
##   [1] -0.30746606  1.03004584  0.28687874  1.04704028 -0.94710317
##   [6]  1.47220457  1.09518173 -0.74579769  2.18172628 -1.11849893
##  [11] -1.54750893  0.58280768  0.56169998 -1.90504942 -0.16288745
##  [16]  0.97930234  0.41161749 -1.16339722 -0.46238939 -0.45691079
##  [21]  1.01054929  2.18358487 -0.45428847 -1.09289855 -1.27896544
##  [26] -1.02236154  1.46655431  0.24671438 -0.13822504  0.02633851
##  [31] -0.39053597 -0.98566020  0.85794433 -1.18800234  0.47219478
##  [36] -0.83095975  0.93789718 -1.06358231  1.63068969 -1.18268548
##  [41] -0.45706647  0.24218608 -0.66534396  0.53520844 -1.39521121
##  [46] -0.76236635 -0.65044322  1.42840121  1.53991341  2.02033852
##  [51] -0.61009197  1.02693608  0.63851573 -1.21953287 -0.37230563
##  [56]  1.24955836 -0.25474775  0.95904513 -0.04343589 -0.90158403
##  [61] -0.18016361  1.13957227 -0.22801236  0.68848324 -0.90323774
##  [66] -0.30948351  0.97913653  0.66462977 -0.90348430  0.44761876
##  [71]  1.49814079  1.43302635 -0.77412672  1.21618641  0.99676256
##  [76]  1.36148680 -0.06207824 -1.90052454 -1.59445894 -0.24467659
##  [81] -0.04096030 -0.35635279  0.68775018 -0.71166304  1.30528920
##  [86]  0.78132145  0.96331722 -0.15273159  0.76603709 -0.75415745
```

```
##  [91] -1.20522751 -1.14279897  1.15809442 -0.16722243 -0.33124989
##  [96] -0.97213868 -0.06129168  0.23706216  0.82052413 -1.22511584
```

Todo task number 5

R allows you to access useful help pages using help()

```
help(sqrt)
```

This example will output an HTML page with useful information about the sqrt() function

Todo task number 6

loading firstscript.R into knitr

```
knitr::read_chunk('firstscript.R')
```

running the code found at firstscript.R

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

```
##   [1] -0.255893099 -1.392802079  0.645423831 -0.041984629 -1.289646229
##   [6]  0.841354344 -0.861339008 -0.387976479  0.963872213 -0.675738990
##  [11]  0.854677504  1.821295187  1.690359731 -1.377192802  0.578883473
##  [16] -0.305099722  0.496064760  1.160260219  0.451452192 -0.763414953
##  [21] -0.165748677  0.627265193 -1.101669452  1.517779667  0.351383714
##  [26]  0.971228722 -0.604273345  0.232937550 -0.682869719  0.113367256
##  [31] -0.216048647  0.762670764  0.414303416  0.102251662  0.231222151
##  [36] -0.349356207  0.720532039  0.903097901 -0.684783953  0.086329530
##  [41] -0.458477193 -2.173273522  2.700668230  1.362413124  1.239042878
##  [46]  1.588098284 -0.001576933  1.389653521 -0.298014597 -1.000362327
##  [51] -1.364506750  0.292137102 -0.792003391 -1.227910868  0.926891320
```

```
##  [56]  0.476496473 -1.004279185  0.030895858  0.597023934 -0.857932861
##  [61]  0.208656532 -1.685122502  0.445257794  1.095452314 -1.525678272
##  [66]  0.105903581 -1.957672311  1.218274867 -0.125585626 -0.879063758
##  [71]  2.904095121  0.298246726  1.437232162  0.370785048  0.544288644
##  [76]  0.445806737  0.487352768 -0.119855614  0.109195804  0.456917211
##  [81]  0.458859252  0.424524290  1.652737908  0.646448300  1.220863234
##  [86] -1.331995476 -1.558224647  0.440824918 -0.732815421  0.005171846
##  [91] -0.950461364  0.358570612 -0.081648010 -1.557407042  0.178518347
##  [96] -0.231324380 -1.732168958  0.305289951  0.263558288  0.026066871
```

Todo task number 7

You can create a matrix using the syntax matrix(data=v, ncol=n) where v is the vector and n is the number of columns to use

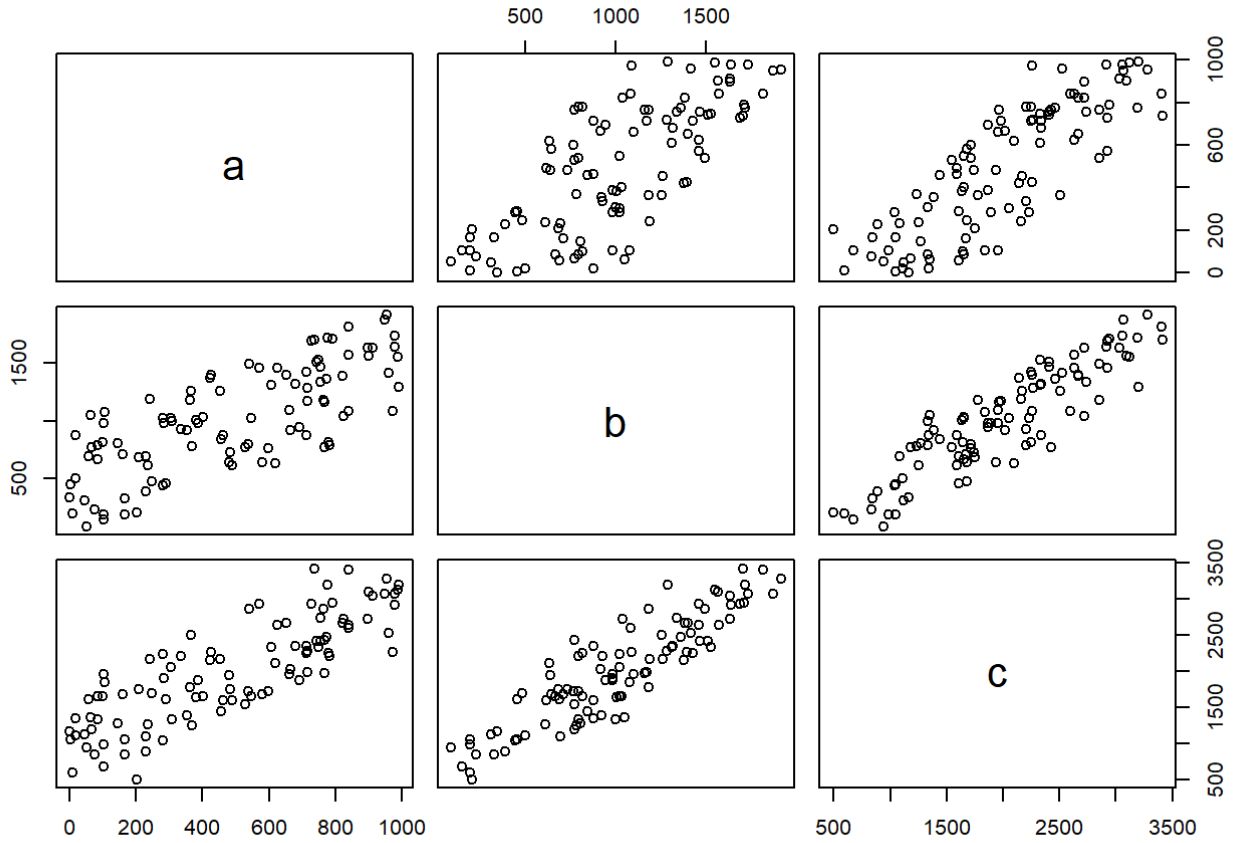
```
P = seq(from=31,to=60,by=1)
M = matrix(data=P, ncol=5, dimnames = list(c(1, 2, 3, 4, 5, 6),c("Q", "Q", "Q", "Q", "Q")))
M
```

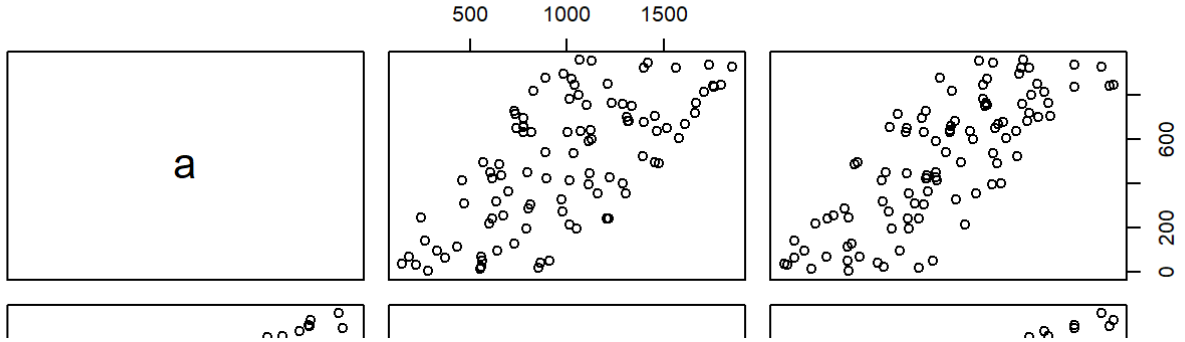
```
##    Q  Q  Q  Q  Q
## 1 31 37 43 49 55
## 2 32 38 44 50 56
## 3 33 39 45 51 57
## 4 34 40 46 52 58
## 5 35 41 47 53 59
## 6 36 42 48 54 60
```

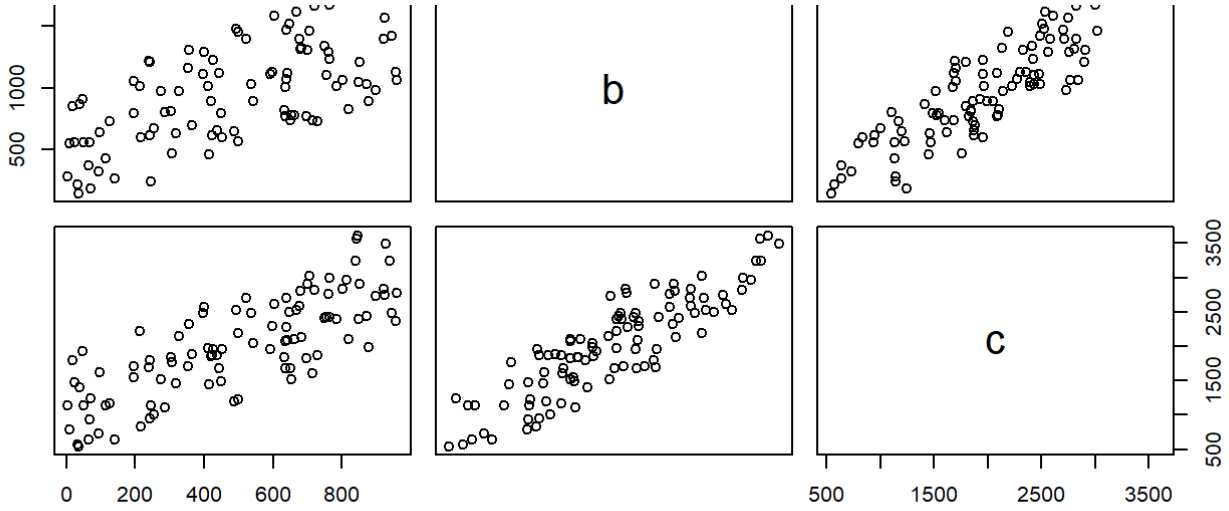
Todo task number 8

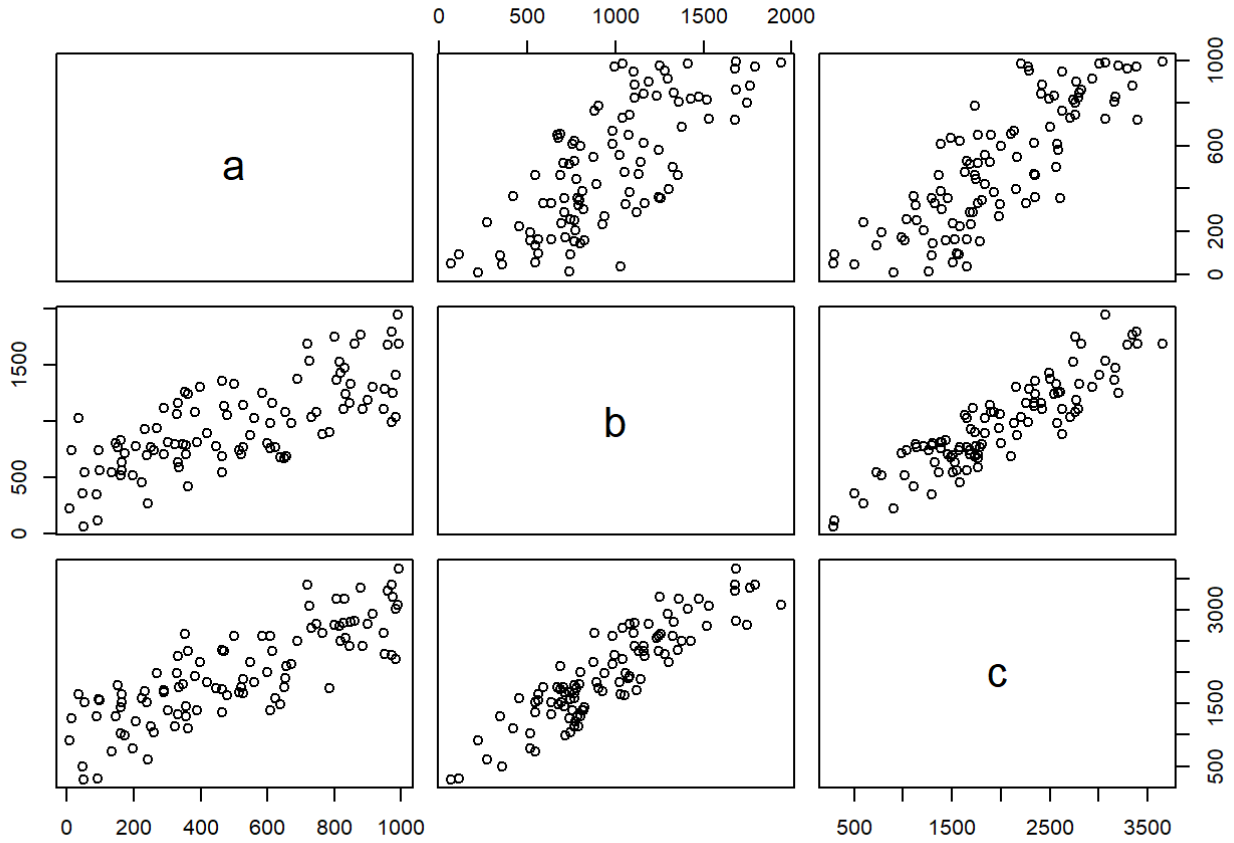
```
for(j in c(1, 2, 3, 4, 5)){
x1 = (as.integer(runif(100, 0, 1000)))
x2 = (as.integer(runif(100, 0, 1000)))
x3 = (as.integer(runif(100, 0, 1000)))
x2 = x1+x2
```

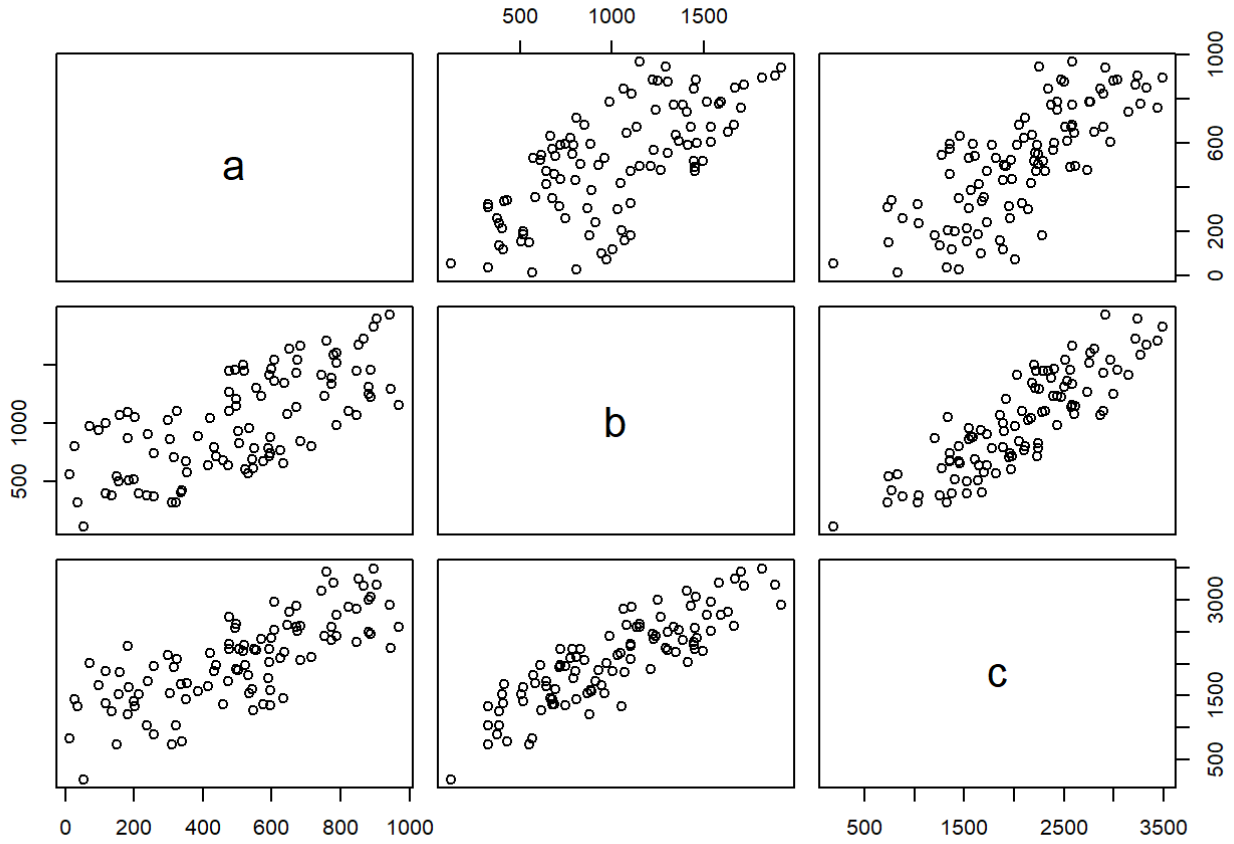
```
x3 = x1+x2+x3
t = data.frame(a= x1,b=x2, c=x3)
plot(t)
rm(list=ls())
}
```

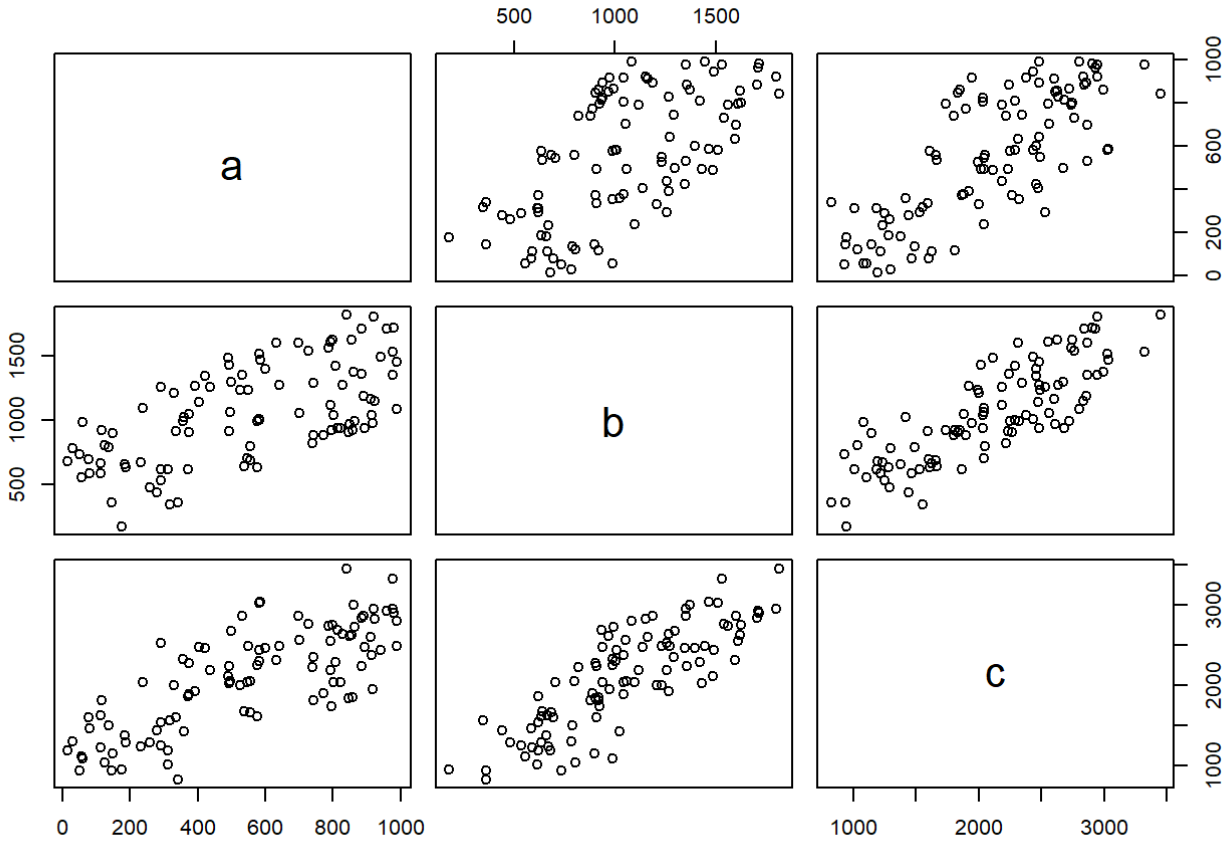












The sd output can be generated using sapply(), sd has been defunct in R for quite some time and no longer works

```
x1 = (as.integer(runif(100, 0, 1000)))
x2 = (as.integer(runif(100, 0, 1000)))
x3 = (as.integer(runif(100, 0, 1000)))
x2 = x1+x2
x3 = x1+x2+x3
t = data.frame(x1,x2,x3)
sapply(t, sd)
```

```
##       x1       x2       x3
## 287.6885 409.0725 682.2343
```

```
rm(list=ls())
```

running this script a few more times produces the following outputs

```
##       x1       x2       x3
## 287.4779 412.9901 786.0489
```

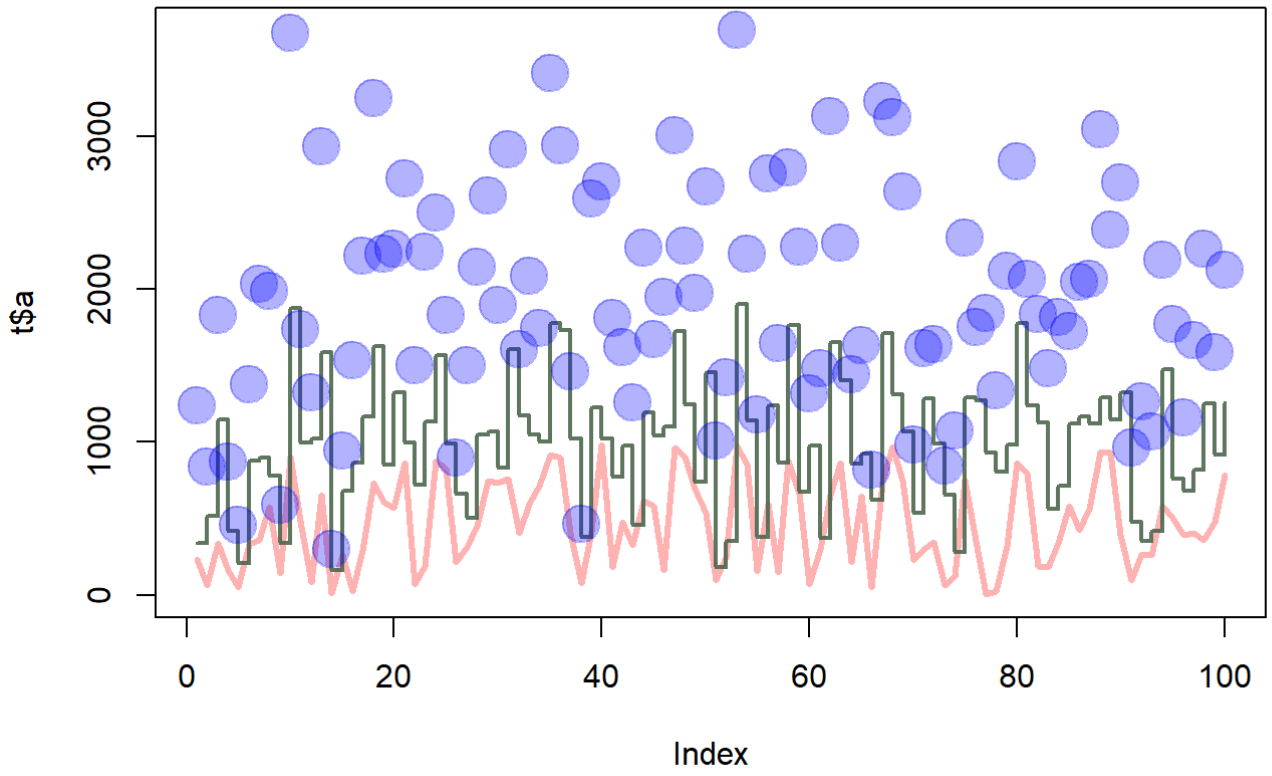
```
##       x1       x2       x3
## 306.2511 355.6550 625.7174
```

```
##       x1       x2       x3
## 290.0505 401.4679 709.9772
```

```
##       x1       x2       x3
## 292.7091 449.7869 791.6638
```

R also has lists whose elements don't have to be of equal sizes

```
x1 = (as.integer(runif(100, 0, 1000)))
x2 = (as.integer(runif(100, 0, 1000)))
x3 = (as.integer(runif(100, 0, 1000)))
x2 = x1+x2
x3 = x1+x2+x3
t = data.frame(a=x1,b=x2,c=x3)
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))
```



rgb is “RedGreenBlue”, it allows us to choose colors for our points. lwd is line width. pch is the type of symbol to use, R has various symbols that can be used for points which are indexed by numbers, 20 is a dot symbol. cex is the scale of the symbols

todo task number 9

```
d = read.table(file="tst2.txt", header = TRUE)
d
```

```
##    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
```

```
d$g = d$g*5
d
```

```
##    a   g  x
## 1  1  10  3
## 2  2  20  6
## 3  4  40 12
## 4  8  80 24
## 5 16 160 48
## 6 32 320 96
```

todo task number 10

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

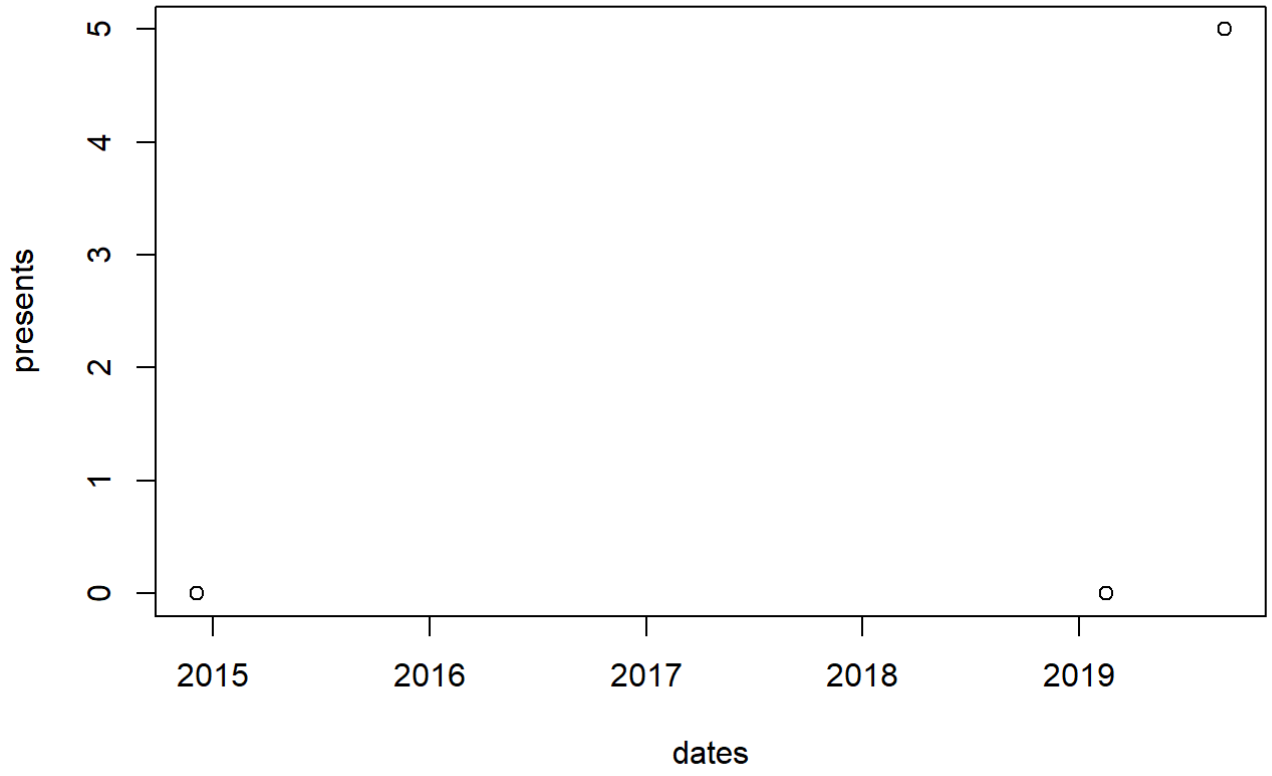
```
## Warning in sqrt(rnorm(100)): NaNs produced
```

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

I get an error, therefore, NA has to be inserted where data is unavailable. If you want to ignore NAs just use na.rm=TRUE as an argument to ignore them.

todo task number 11

```
dates=strptime(c(20190215, 20141205, 20190903), format="%Y%m%d")
presents=c(0,0,5)
df <- data.frame(dates, presents)
plot(df)
```



todo task number 12

```
n = seq(from=1, to=100, by=1)
f = c()
for(x in n){
if(x < 5 | x > 90){
f[x] = n[x] * 10
}
else{
f[x] = n[x] * 0.1
}

}
f
```

```
##   [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 task number 13

```
func = function(arg1){
f=c()
for(i in arg1){
if(i < 5 | i > 90){
f[i] = arg1[i] * 10
```

```
}
else{
f[i] = arg1[i] * 0.1
}
}
f
}
func(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
```

```
##  [1] 10.0 20.0 30.0 40.0  0.5  0.6  0.7  0.8  0.9  1.0
```

todo task number 14

Here, I have to prove the following footnote “… people often use more for-loops than nec-essary.The ToDo above can be done more easilyand quickly without a for-loop but with regular vector-computations.”

```
x=c(1, 2, 30, 4, 10, 20, 90, 123, 14567)
x
```

```
## [1]     1     2    30     4    10    20    90   123 14567
```

```
f <- ifelse(x<5,x*10, ifelse(x>90, x*10, x*0.1))
f
```

```
## [1]     10     20      3     40      1      2      9   1230 145670
```

No For loop was needed here