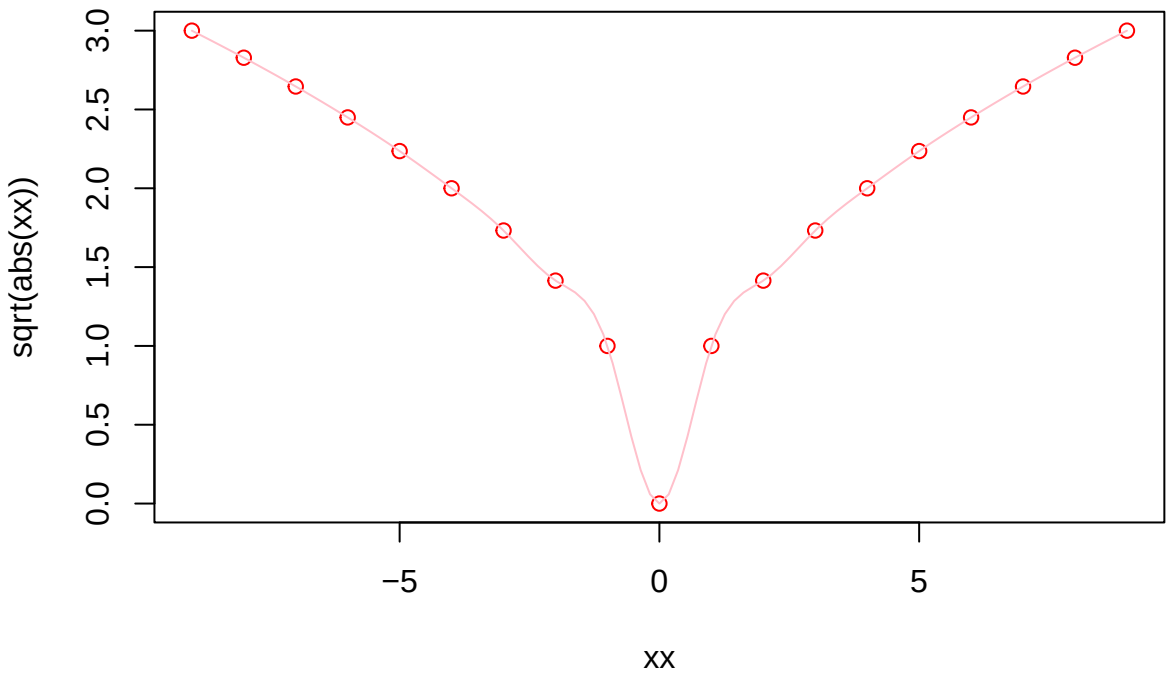
R Exam

Ehsan Eslmai Shafigh

2023-06-14

```
example(sqrt)
```

```
##
## sqrt> require(stats) # for spline
##
## sqrt> require(graphics)
##
## sqrt> xx <- -9:9
##
## sqrt> plot(xx, sqrt(abs(xx)),  col = "red")
```




```
##
## sqrt> lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

```
?plot
```

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

```
objects()
```

```
## [1] "xx"
```

```
search()
```

```
## [1] ".GlobalEnv"        "package:stats"     "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"         "package:base"
```

```
str(mean)
```

```
## function (x, ...)
```

```
str(sqrt)
```

```
## function (x)
```

```
119%/%12
```

```
## [1] 9
```

```
119%%12
```

```
## [1] 11
```

```
x = c(1 , 2 , 3)
y = c(2 , 3 )

x * y
```

```
## Warning in x * y: longer object length is not a multiple of shorter object
## length
```

```
## [1] 2 6 6
```

```
seq(1 , 7 , 2)
```

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

```r
# we want to remove the smallest and largest values of a vector

v = c(10,13,5,3,7,1)

v = sort(v)
v = v[-1]
v = v[-length(v)]

v
```

```
## [1]  3  5  7 10
```

```r
x = 1:10

sum(x < 5)
```

```
## [1] 4
```

```r
sum(x[x<5])
```

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

```r
rep(3,5)
```

```
## [1] 3 3 3 3 3
```

```r
rep(1:4 , 2)
```

```
## [1] 1 2 3 4 1 2 3 4
```

```r
rep(1:4 , each = 2)
```

```
## [1] 1 1 2 2 3 3 4 4
```

```r
rep(1:4 , each = 2 , times = 3)
```

```
##  [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

```r
rep(1:4 , 1:4)
```

```
##  [1] 1 2 2 3 3 3 4 4 4 4
```

```r
s = 1:3

class(s)
```

```
## [1] "integer"
```

```
typeof(s)
```

```
## [1] "integer"
```

```
c = c(1 , 5 , 10)

class(c)
```

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

```
typeof(c)
```

```
## [1] "double"
```

```
# we want to get the all odd , positive numbers between -5 and 5

ns = -5:5

ns.positive.index = ns > 0
ns.odd.index = ns %% 2 == 1

ns[ns.positive.index & ns.odd.index]
```

```
## [1] 1 3 5
```

```
y = c(5 , NA , 3)

is.na(y)
```

```
## [1] FALSE  TRUE FALSE
```

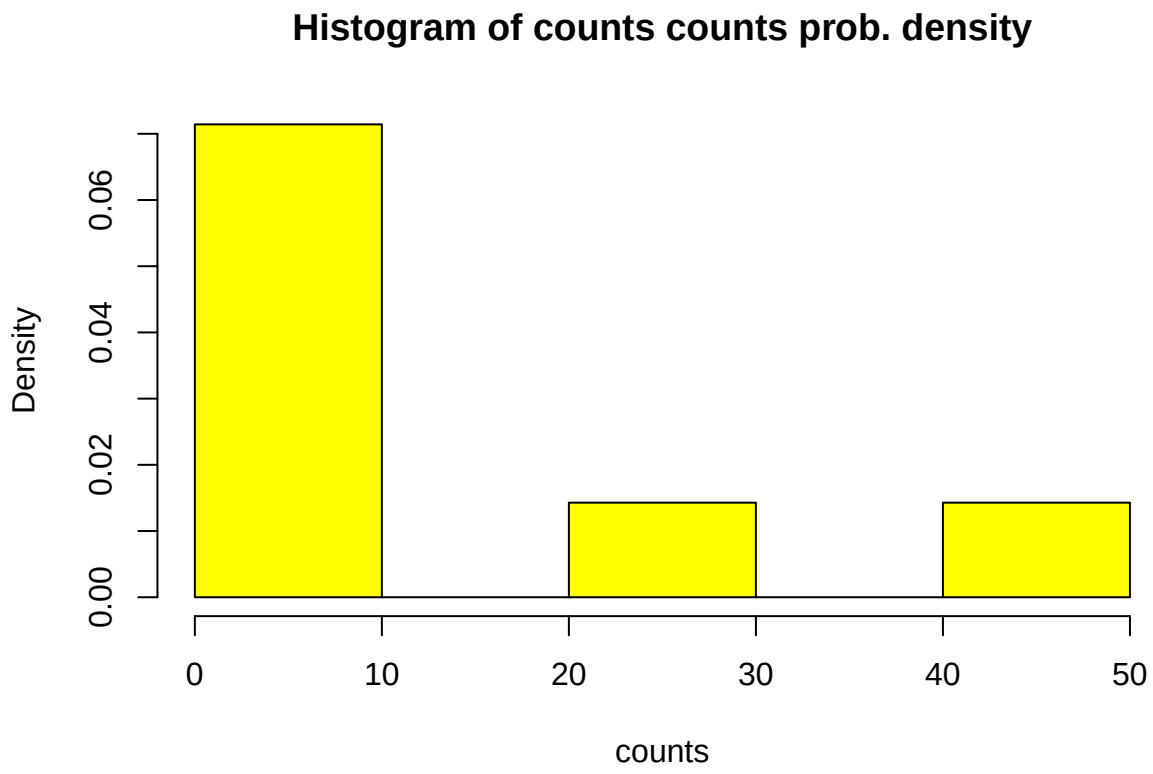
```
# this way we can remove the NA values

y[! is.na(y)]
```

```
## [1] 5 3
```

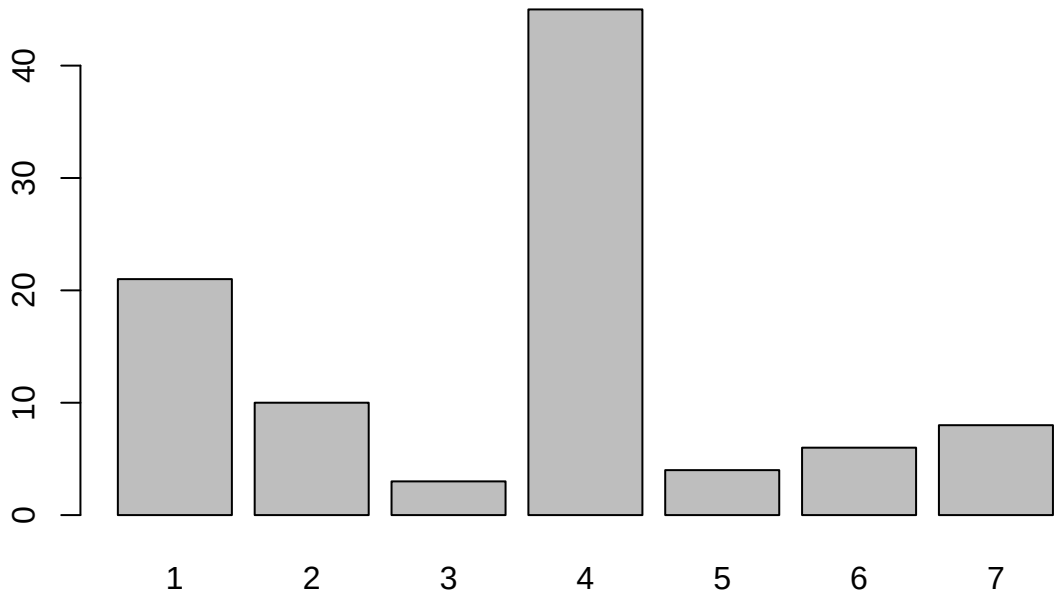
```
gender = c("male" , "female", "female", "female","male" , "female")
gen_factor = factor(gender)

str(gen_factor)
```

```
##  Factor w/ 2 levels "female","male": 2 1 1 1 2 1
```

```
# naming

counts = c(21 , 10 , 3 , 45 , 4 , 6 , 8)
names(counts) = c(1:length(counts))

hist(counts , col = "yellow" , freq = F , main = paste("Histogram of" , "counts" , xlab = "counts" , yla
```




```
barplot(counts)
```




```
?hist()
```

```
?barplot()
```

```
v1 = c(1:20)
dim(v1) = c(5 , 4)

class(v1)
```

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

```
str(v1)
```

```
##  int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
```

```
v1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

```r
v1 = c(1:20)
dim(v1) = c(5 , 2 , 2)
class(v1)
```

```
## [1] "array"
```

```r
str(v1)
```

```
##  int [1:5, 1:2, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
```

```r
v1
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
##
## , , 2
##
##      [,1] [,2]
## [1,]   11   16
## [2,]   12   17
## [3,]   13   18
## [4,]   14   19
## [5,]   15   20
```

```r
m1 = matrix(1:12 , nrow = 3 , byrow = T)
m1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

```r
m1[3 , ]
```

```
## [1]  9 10 11 12
```

```r
A = matrix(c(1 , -2 , 4 , 3) , nrow = 2)
b = c(12 , 4)

x = solve(A , b)

x
```

```
## [1] 1.818182 2.545455
```

```
A%*%x
```

```
##      [,1]
## [1,]   12
## [2,]    4
```

```
ar = array(1:24 , dim = c(3 , 2 , 4) )

ar
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
##
## , , 2
##
##      [,1] [,2]
## [1,]    7   10
## [2,]    8   11
## [3,]    9   12
##
## , , 3
##
##      [,1] [,2]
## [1,]   13   16
## [2,]   14   17
## [3,]   15   18
##
## , , 4
##
##      [,1] [,2]
## [1,]   19   22
## [2,]   20   23
## [3,]   21   24
```

```
ar[, , 3]
```

```
##      [,1] [,2]
## [1,]   13   16
## [2,]   14   17
## [3,]   15   18
```

```
?array
```

```
u = sqrt(2)

identical(u*u , 2)
```

```
## [1] FALSE
```

```r
all.equal(u*u , 2)
```

```
## [1] TRUE
```

```r
# we want to write a code which compares two numbers and tell us which one is bigger than the other

x = 10
y = 15

#first method:

if (x>y) {
  cat("x is bigger\n")
}else if (y>x) {
  cat("y is bigger\n")
  }else {
    cat("x and y are equal")
  }
```

```
## y is bigger
```

```r
# second method

sign = sign(x - y)

msg = switch(as.character(sign) , "1" = "x is bigger" , "-1" = "y is bigger" , "0" = "x and y are equal

msg
```

```
## [1] "y is bigger"
```

```r
# print the index number along with the values of a vector

v = c("a" , "b" , "c")

for (i in seq_along(v)) cat(i , v[i] , "\n")
```

```
## 1 a
## 2 b
## 3 c
```

```r
# write a function that returns the factorial of a given number

facto = function (n) {

  m = n
  tmp = n

  while (m > 1) {

    m = m - 1
```

```
    tmp = m * tmp
  }

  return (tmp)
}


facto(3)
```

```
## [1] 6
```

```
cumprod(1:3)[length(1:3)]
```

```
## [1] 6
```

```
factorial(3)
```

```
## [1] 6
```

```
# draw 5 random numbers from poisson dist. and take the log, then replace the infinit results with nan

y = log(rpois(5 , 1.5))

y
```

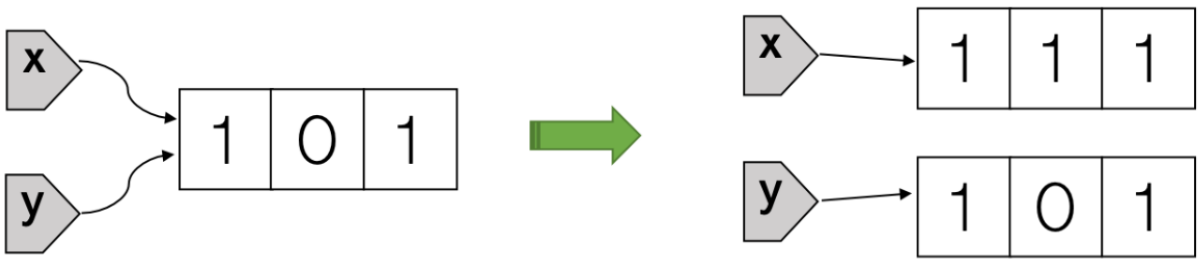
```
## [1]     -Inf     -Inf 1.098612 0.000000 0.000000
```

```
y = ifelse(y<0 , NA , y)

y
```

```
## [1]       NA       NA 1.098612 0.000000 0.000000
```

```
# unlike python list which are mutable, R vectors are immutable, meaning that if we assign two variable

x = c(1 , 5 ,7)
y = x

y
```

```
## [1] 1 5 7
```



```
x[1] = 2

y
```

```
## [1] 1 5 7
```

```
# lists:

l1 = list(c(1 , 3 ,5) , c(T , F , T) , "list is a good thing")

str(l1)
```

```
## List of 3
##  $ : num [1:3] 1 3 5
##  $ : logi [1:3] TRUE FALSE TRUE
##  $ : chr "list is a good thing"
```

```
#extracting from lists

l1[1]
```

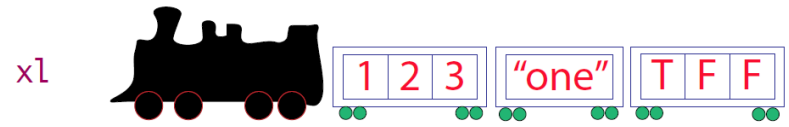
```
## [[1]]
## [1] 1 3 5
```

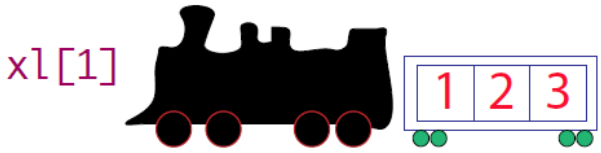
```
l1[[1]]
```

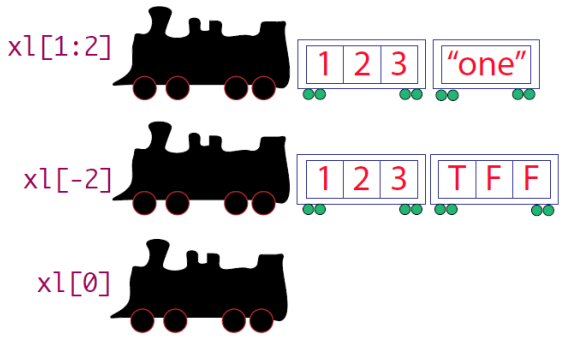
```
## [1] 1 3 5
```

```
# if you put one bracket you get another list, if you put two brackets you get the content of that list

l1[[1]][2]
```

```
## [1] 3
```








```
# we can also give index to the items whithin a list

l2 = list(num = c(1,2,3) , text = "R is bullshit")

str(l2)
```

```
## List of 2
##  $ num : num [1:3] 1 2 3
##  $ text: chr "R is bullshit"
```

```
l2$text
```

```
## [1] "R is bullshit"
```

```
l3 = l2

l2[1] = list(3:1)

l2
```

```
## $num
## [1] 3 2 1
##
## $text
## [1] "R is bullshit"
```

```
l3
```

```
## $num
## [1] 1 2 3
##
## $text
## [1] "R is bullshit"
```

```
# data frames are built on top of lists. They they are lists which have fixed lengths.

letter = letters[1:5]
numbers = 1:5

df1 = data.frame(letter , numbers)

str(df1)
```

```
## 'data.frame':    5 obs. of  2 variables:
##  $ letter : chr  "a" "b" "c" "d" ...
##  $ numbers: int  1 2 3 4 5
```

```
#accessing

df1[[1]]
```

```
## [1] "a" "b" "c" "d" "e"
```

```
df1$letter
```

```
## [1] "a" "b" "c" "d" "e"
```

```
# dataframes can be also accessed the matrix way

#df[1,2]
```

```
# write a code which selects 2 rows randomly from df1

df1[sample(1:5 , 2) ,]
```

```
##   letter numbers
## 4      d       4
## 1      a       1
```

```
# select only the columns which are numeric

filter = sapply(df1 , is.numeric)

df1[,filter]
```

```
## [1] 1 2 3 4 5
```

```
# drop the last two rows from df1

df2 = df1

len = length(df1[[1]])

df2 = df1[-((len-1) : len),]

df1
```

```
##   letter numbers
## 1      a       1
## 2      b       2
## 3      c       3
## 4      d       4
## 5      e       5
```

```
df2
```

```
##   letter numbers
## 1      a       1
## 2      b       2
## 3      c       3
```

```
#subsetting

# select every other row in df1

filter = c(T , F)

df1[filter , ]
```

```
##   letter numbers
## 1      a       1
## 3      c       3
## 5      e       5
```

```
# zero!

x = c(1 , 2 , 3)

x[0]
```

```
## numeric(0)
```

```
# named vectors can be also accessed with thier names

x = setNames(x , letters[1:3])

x["a"]
```

```
## a
## 1
```