IntroR.R

Wow

Mon Aug 06 18:05:06 2018

```
# Introduction to R for Data Mining

#Assign operators
x <- 2
y = 3
assign("z",4)

#Vectors
x <- c(9,4,2,5,8) #c() is combine function
b <- c(x,0,x)
c <- 2*x + y + 1

#Arithmetic operators
1/x
```

```
## [1] 0.1111111 0.2500000 0.5000000 0.2000000 0.1250000
```

```
x+1
```

```
## [1] 10  5  3  6  9
```

```
mean(x)    #find average of all values in vector x
```

```
## [1] 5.6
```

```
length(x)  #find length of vector x
```

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

```
#Regular sequence
x1 <- 1:10         # create a vector starting from 1 to 10 (incrementing by
1)
x2 <- 10:1         # create a vector starting from 10 to 1 (decrementing by
1)
x3 <- seq(-5,5,by=0.2)    #incrementing by 0.2
x4 <- seq(5,-5,by=-0.2)   #decrementing by -0.2
x5 <- seq(length=10, from=-5, by=0.2)
x6 <- rep(2, times=5)
x6 <- rep(2, 5)  #for short of the above command
x7 <- rep(x, 5)  #put 5 copies of x end-to-end
x8 <- rep(x, each=5) #repeat each element of x 5 times before moving to the
next
```

```r
#Logical vectors
tf <- x >= 5
as.numeric(tf)
```

```
## [1] 1 0 0 1 1
```

```r
#Missing values
m <- c(1:3, NA) #create a vector of size 4 with missing value
mm <- is.na(m)  #find missing values

#Other values
n <- 0/0   #create a NaN (not a number)
n
```

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

```r
i <- 2^5000 #cretate infinity
i
```

```
## [1] Inf
```

```r
#Character vectors
s <- c("Michael","Nancy","Vicky")
s
```

```
## [1] "Michael" "Nancy"   "Vicky"
```

```r
paste("Hello", s)  #pasting strings together
```

```
## [1] "Hello Michael" "Hello Nancy"   "Hello Vicky"
```

```r
labs <- paste(c("X","Y"), 1:10, sep="") #c("X","Y") is repeat 5 times to
match 1:10
labs
```

```
##  [1] "X1"  "Y2"  "X3"  "Y4"  "X5"  "Y6"  "X7"  "Y8"  "X9"  "Y10"
```

```r
#Selecting and modifying subsets
x <- c(8,6,4,2,0)
x[1]  #select the first element
```

```
## [1] 8
```

```r
x[-1]  #remove the first element
```

```
## [1] 6 4 2 0
```

```r
x[2:4]  #select elements 2 to 4
```

```
## [1] 6 4 2
```

```r
x[-(2:4)]  #remove elements 2 to 4
```

```
## [1] 8 0
```

```r
x[x>4]  #select elements that are more than 4
```

```
## [1] 8 6
```

```r
x[2:4] <- 1:3  #replace elements
x
```

```
## [1] 8 1 2 3 0
```

```r
#names()
fruit <- c(5,10,1,20)
names(fruit) <- c("orange","banana","apple","peach")
fruit
```

```
## orange banana  apple  peach
##      5     10      1     20
```

```r
lunch <- fruit[c("apple","orange")]
lunch
```

```
##  apple orange
##      1      5
```

```r
#mode
z <- 0:9
mode(z)
```

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

```r
zz <- as.character(z)  #coercion
mode(zz)
```

```
## [1] "character"
```

```r
zzz <- as.numeric(zz)
mode(zzz)
```

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

```r
#length
length(z)
```

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

```r
e <- numeric() #make e an empty vector structure of mode numeric
e
```

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

```r
length(e)
```

```
## [1] 0
```

```r
e[5] <- 12  #implicitly change length of e
e
```

```
## [1] NA NA NA NA 12
```

```r
length(e) <- 7  #changing the length of e explicitly (vector can be extended its length by missing value)
e
```

```
## [1] NA NA NA NA 12 NA NA
```

```r
aa <- 11:20
aa
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
aa <- aa[2*1:5]  #make it an object of length 5 consisting of just the former components with even index
length(aa)
```

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

```r
length(aa) <- 3  #retain just the first 3 values
aa
```

```
## [1] 12 14 16
```

```r
#attribute
z <- 1:4
attributes(z)
```

```
## NULL
```

```r
class(z)
```

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

```r
z
```

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

```r
attr(z, "dim") <- c(2,2)
attributes(z)
```

```
## $dim
## [1] 2 2
```

```r
class(z)
```

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

```r
z
```

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

#Ordered and unordered factors
state <- c("tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa",
           "qld", "vic", "nsw", "vic", "qld", "qld", "sa", "tas",
           "sa", "nt", "wa", "vic", "qld", "nsw", "nsw", "wa",
           "sa", "act", "nsw", "vic", "vic", "act")
statef <- factor(state)
statef

##  [1] tas sa  qld nsw nsw nt  wa  wa  qld vic nsw vic qld qld sa  tas sa
## [18] nt  wa  vic qld nsw nsw wa  sa  act nsw vic vic act
## Levels: act nsw nt qld sa tas vic wa

levels(statef)

## [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"

#tapply
incomes <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56,
             61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46,
             59, 46, 58, 43)
incmeans <- tapply(incomes, statef, mean)  #calculate the sample mean income
for each level of statef
incmeans

##      act      nsw       nt      qld       sa      tas      vic       wa
## 44.50000 57.33333 55.50000 53.60000 55.00000 60.50000 56.00000 52.25000

#ordered
stateo <- ordered(state)
stateo

##  [1] tas sa  qld nsw nsw nt  wa  wa  qld vic nsw vic qld qld sa  tas sa
## [18] nt  wa  vic qld nsw nsw wa  sa  act nsw vic vic act
## Levels: act < nsw < nt < qld < sa < tas < vic < wa

levels(stateo)

## [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"

#Array
z <- 1:12
dim(z) <- c(2,3,2)  #set dimensions to vector z; z becomes the array
z

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

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

```
zz <- array(1:12, dim=c(2,3,2))

#Array indexing
zz[2,3,2]
```

```
## [1] 12
```

```
zz[2,3,]
```

```
## [1]  6 12
```

```
zz[2,,]
```

```
##      [,1] [,2]
## [1,]    2    8
## [2,]    4   10
## [3,]    6   12
```

```
zz[2,2:3,1]
```

```
## [1] 4 6
```

```
#Matrix
x <- matrix(1:20,4,5)  #4 = number of rows, 5 = number of columns
x
```

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

```
#Index matrix
i <- matrix(c(1:3,3:1),3,2)  #Generate a 3 by 2 index matrix
i
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    2
## [3,]    3    1
```

```
x[i]  #Extract those elements
```

```
## [1] 9 6 3
```

```r
x[i] <- 0  #Replace those elements by 0
x
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    0   13   17
## [2,]    2    0   10   14   18
## [3,]    0    7   11   15   19
## [4,]    4    8   12   16   20
```

```r
#Matrix facilities
dim(x)
```

```
## [1] 4 5
```

```r
nrow(x)
```

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

```r
ncol(x)
```

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

```r
length(x)
```

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

```r
t(x)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    0    4
## [2,]    5    0    7    8
## [3,]    0   10   11   12
## [4,]   13   14   15   16
## [5,]   17   18   19   20
```

```r
#Name rows and columns
y <- matrix(1:6,2,3)
y
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
rownames(y) <- c("Michael","Peter")
y
```

```
##         [,1] [,2] [,3]
## Michael    1    3    5
## Peter      2    4    6
```

```r
colnames(y) <- c("Age","Weight","Height")
y
```

```
##         Age Weight Height
## Michael   1      3      5
## Peter     2      4      6
```

```
dimnames(y)
```

```
## [[1]]
## [1] "Michael" "Peter"
##
## [[2]]
## [1] "Age"    "Weight" "Height"
```

```
#Forming partitioned matrics
m1 <- matrix(1,2,2)
m2 <- matrix(2,2,2)
cm <- cbind(m1,m2)
cm
```

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

```
rm <- rbind(m1,m2)
rm
```

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

```
crm <- rbind(cm,cbind(m2,m1))
crm
```

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

```
#Lists
lst <- list(name="Fred",wife="Mary",no.children="3",child.ages=c(4,8,9))
lst
```

```
## $name
## [1] "Fred"
##
## $wife
## [1] "Mary"
##
## $no.children
## [1] "3"
```

```
##
## $child.ages
## [1] 4 8 9

lst[[2]]

## [1] "Mary"

lst$wife

## [1] "Mary"

lst[[4]]

## [1] 4 8 9

lst[[4]][1]

## [1] 4

lst$child.ages[1]

## [1] 4

#Data frames
df <- data.frame(name=c("Michael","Mark","Maggie"), children=c(2,0,2))
df

##      name children
## 1 Michael        2
## 2    Mark        0
## 3  Maggie        2

df$name

## [1] Michael Mark    Maggie
## Levels: Maggie Mark Michael

df[1,]

##      name children
## 1 Michael        2

df[,1]

## [1] Michael Mark    Maggie
## Levels: Maggie Mark Michael

#Read data from files
write.table(df, file="df.dat", sep=",")
df2 <- read.table("df.dat", sep=",")
df2
```

```
##      name children
## 1 Michael        2
## 2    Mark        0
## 3  Maggie        2
```

```
write.csv(df, file="df.csv")
df3 <- read.csv("df.csv")
df3
```

```
##   X    name children
## 1 1 Michael        2
## 2 2    Mark        0
## 3 3  Maggie        2
```

```
#Load data sets
data(iris)  #load the iris data set
iris[1,]    #shows the first row
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```
head(iris)  #shows the first few rows
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
data()
```

```
#If
x <- 12
if (x > 10) {
  cat("x is >10")
} else {
  cat("x is <=10")
}
```

```
## x is >10
```

```
x <- c(12,16,3)
if(all(x>10)) cat("All values in x is >10")
if(any(x>10)) cat("There is at least one value >10")
```

```
## There is at least one value >10
```

```
#For
x <- 0
for (i in 1:5){
```

```
  x <- x+i
}
x
```

```
## [1] 15
```

```
sum(1:5)  #equivalent to the for loop above
```

```
## [1] 15
```

```
#Repeat
x <- 0
c <- 1
repeat{
  x <- x+c
  c <- c+1
  if (c == 6) break
}
x
```

```
## [1] 15
```

```
c
```

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

```
#While
x <- 0
c <- 1
while(c < 6){
  x <- x+c
  c <- c+1
}
x
```

```
## [1] 15
```

```
c
```

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

```
#Function
inc <- function(x) {x <- x+1}
inc
```

```
## function(x) {x <- x+1}
```

```
mode(inc)
```

```
## [1] "function"
```

```
x1 <- inc(5)  #call function inc
x1
```

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

```
x2 <- inc(1:10)
x2
```

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

```
inc <- function(x, b=1) {x <- x+b}  #setting defaults
x1 <- inc(5)
x1
```

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

```
x2 <- inc(1:10,10)  #change b from 1 to 10
x2
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```
#lapply, sapply, apply
lt <- list(1:3,6,7:3)
lapply(lt, FUN=function(x) {rev(x)}) #apply reverse function to all element
```

```
## [[1]]
## [1] 3 2 1
##
## [[2]]
## [1] 6
##
## [[3]]
## [1] 3 4 5 6 7
```

```
sapply(lt, length)
```

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

```
m <- matrix(1:9,3)  #create a matrix
m
```

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

```
apply(m, MARGIN=1, sum)  #MARGIN=1 means row operation
```

```
## [1] 12 15 18
```

```
apply(m, MARGIN=2, sum)  #MARGIN=1 means column operation
```

```
## [1]  6 15 24
```

```
rowSums(m)
```

```
## [1] 12 15 18
```

```r
colSums(m)
```

```
## [1]  6 15 24
```

```r
#Getting help
help(solve)  #get help about solve
```

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

```r
?solve  #same as above
help.start()  #launch a Web browser that allows the help pages to be browsed
with hyperlinks
```

```
## If nothing happens, you should open
## 'http://127.0.0.1:14264/doc/html/index.html' yourself
```

```r
??solve  #allows searching for help in various ways about solve
```