R Programlamaya Giriş

Muhammed Fatih TÜZEN

04 01 2022

İÇİNDEKİLER

1 Giriş

1.1 Çalışma Dizini

Çalışma Dizini, üzerinde çalıştığınız veri kümeleri vb. gibi tüm gerekli dosya ve belgelerinizi içeren yerdir. Çalışma dizininizi ayarlamanın iki yolu vardır. İlk yol **getwd ve setwd** işlevlerini kullanmaktır. Diğer yol ise RStudio üzerinden **Session>Set Working Directory** youluyla yapılabilir.

```
getwd() #Çalışma dizini öğrenilir.
```

```
## [1] "E:/R/R_Course/R"
```

```
# Çalışma dizini istenilen şekilde ayarlanabilir.
#setwd("C:/Users/mfati") # windows
# setwd("~/Users/mfati") # linux
```

dir veya **list.files** komutları ile dizinde yer alan dosyalar öğrenilebilir.

```
dir("datasets")
```

```
## [1] "counties.rds"        "data.RData"          "mtcars.rds"
## [4] "mtcars_csv.csv"      "mtcars_csv2.csv"     "mtcars_excel.xlsx"
## [7] "mtcars_txt.txt"      "my_data.RData"       "my_work_space.RData"
```

```
list.files("datasets")
```

```
## [1] "counties.rds"        "data.RData"          "mtcars.rds"
## [4] "mtcars_csv.csv"      "mtcars_csv2.csv"     "mtcars_excel.xlsx"
## [7] "mtcars_txt.txt"      "my_data.RData"       "my_work_space.RData"
```

- **file.exists** kullanılarak klasörün var olup olmadığı sorgulanabilir.
- **dir.create** komutu ile yeni bir klasör oluşturmak mümkündür.

```
file.exists("R-Programlamaya-Giriş.pdf")
```

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

```
file.exists("newfolder")
```

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

```
dir.create("newfolder")
```

```
## Warning in dir.create("newfolder"): 'newfolder' zaten var
```

```
file.exists("newfolder")
```

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

1.2 Yardım

Herhangi bir fonksiyonla ilgili yardım almak için "?" veya help () komutları kullanılır.

```
# mean ve median fonksiyonları hakkında yardım aşağıdaki şekillerde alınabilir
?mean
```

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

```
help(median)
```

1.3 Yardımcı Bilgiler

R komutlarında *Büyük-küçük harf duyarlılığı (case sensitive)* vardır.

```
a <- 5
print(a)
```

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

```
A <- 6
print(A)
```

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

Noktalı virgül (;) işareti ile aynı satırda birden fazla kod çalıştırılabilir hale getirilir.

```
x <- 1 ; y <- 2 ; z <- 3
x; y; z
```

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

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

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

Komutlar arası açıklamaları ve yorumları **#(hashtag)** ile yazabiliriz. Hastagli satırlar, kod olarak algılanıp çalıştırılmaz. Bu kısımlara yazılan kodlar ile ilgili hatırlatıcı bilgiler (comment) yazılabilir.

```
# x <- 1:5

# 6 ile başyalan ve  10 ile  biten tamsayıları c vektörüne atayalım
c <- 6:10
c
```

```
## [1]  6  7  8  9 10
```

- **ls()** çalışma alanındaki nesne ve fonksiyonları listeler.
- **rm(a)** çalışma alanından **a** nesnesini siler.
- **rm(list=ls())** bütün çalışma alanını temizler.
- **q()** R'dan çıkış yapmayı sağlar.
- **install.packages("package")** paket yüklemeyi sağlar.
- **library("pacakge")** yüklü olan paketi getirir.
- **installed.packages()** yüklü olan paketleri listeler
- **options(digits=10)** sayılarda ondalık kısmın basamak sayısını ifade eder.

2 Temel Operatörler

2.1 Atama Operatörü

Bir değişkene, tabloya veya objeye değer atarken **'<-'** veya **'='** operatörü kullanılır. '<-' atama operatöründe ok hangi yöndeyse o tarafa atama yapılır. Genellikle '<-' operatörü kullanılmaktadır. Çünkü '=' operatörü filtrelemelerde veya işlemlerdeki '==' ile karıştırılabilmektedir. Ayrıca fonksiyonlar içinde de kullanılabildiği için kod karmaşasına sebebiyet verebilir. Her iki operatör de aynı işlevi görmektedir.

```
# a'ya 20 değerini atayalım
a <- 20

# tabloyu ya da değeri görüntülemek için nesnenin kendisi de direkt yazılabilir.
# ya da print fonksiyonu kullanılabilir.

print(a)
```

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

```
# b'ye 12 değerini atayalım
b <- 12
print(b)
```

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

```
# a ve b değerlerinden üretilen bir c değeri üretelim.

c <- 2 * a + 3 * b
print(c)
```

```
## [1] 76
```

c() ile vektör oluştutulabilir. c "combine" (birleştirmek) kelimesinin ilk harfini ifade eder. Bir değişkene birden fazla değer atamak istediğimizde kullanılır.

```
# d adında bir vektör oluşturalım ve değerler atayalım.
d <- c(4,7,13)
d
```

```
## [1]  4  7 13
```

Bir metini değişkene atamak istersek de aşağıdaki gibi metin " " işareti içine yazılmalıdır.

```
metin <- "Merhaba Arkadaşlar"
print(metin)
```

```
## [1] "Merhaba Arkadaşlar"
```

2.2 Matematiksel Operatörler

R ve R Studio, güçlü bir hesap makinesi olarak kabul edilebilir.

```
3+5
```

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

```
7*8
```

```
## [1] 56
```

```
88/2
```

```
## [1] 44
```

```
3*(12+(15/3-2))
```

```
## [1] 45
```

```
9^2 # karesini alır
```

```
## [1] 81
```

```
a <-  3
b <-  a^2
print(b)
```

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

```
log(15) #ln15 yani doğal logaritma
```

```
## [1] 2.70805
```

```r
log10(1000) # 10 tabanına göre hesaplama
```

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

```r
exp(12) #exponential power of the number. e (2.718) üzeri 12
```

```
## [1] 162754.8
```

```r
factorial(6) # faktöriyel hesaplama yapar
```

```
## [1] 720
```

```r
sqrt(81) # karekör alma
```

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

```r
abs(-3) # mutlak değer
```

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

```r
sign(-5) # işaret bulma
```

```
## [1] -1
```

```r
sin(45) # sinüs
```

```
## [1] 0.8509035
```

```r
cos(90) # cosinüs
```

```
## [1] -0.4480736
```

```r
pi # pi sayısı
```

```
## [1] 3.141593
```

```
tan(pi) # tanjant
```

```
## [1] -1.224647e-16
```

2.3 Mantıksal Operatörler

Mantıksal sorgulamalar, koşullarda ve filtrelerde kullanılmaktadır. Verilen koşul veya filtre sağlandığında **TRUE**, sağlanmadığında ise **FALSE** değerleri elde edilmektedir. Bu mantıksal operatörler ayrıca komutlar içindeki özellikleri aktifleştirmek ve pasifleştirmek için de kullanılmaktadır.

Mantıksal operatörler aşağıdaki şekilde kullanılmaktadır:

- eşittir : ==
- eşit değildir : **!=**
- küçüktür : <
- küçük eşittir : <=
- büyüktür : >
- büyük eşittir : >=
- x değil : **!x**
- x ve y : **x&y**
- x veya y: **x|y**

```
3 > 5
```

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

```
# & (ve) operatörü
# iki durumda TRUE ise sonuç TRUE döner.
3 < 5 & 8 > 7
```

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

```
# bir durum FALSE diğer durum TRUE ise sonuç FALSE döner.
3 < 5 & 6 > 7
```

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

```
# iki durumda FALSE ise sonuç FALSE döner.
6 < 5 & 6 > 7
```

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

```
# | (veya) operatörü
# Her iki durumdan birisi TRUE ise TRUE döner
(5==4) | (3!=4)
```

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

3 Veri Tipleri ve Yapıları

R'da kulllanılan 5 temel veri tipi vardır. Bu veri tipleri atomic vectörler olarak da bilinir. Atomic olması vektörlerin homojen olması anlamına gelmektedir. Yani vektör içerisinde aynı veri tipinden değerler yer alabilir. Veri tipleri;

- numeric veya double (reel sayılar)
- integer (tamsayılar)
- complex (karmaşık sayılar)
- character (metinsel ifadeler)
- logical, TRUE ve FALSE (mantıksal)

typeof() veya class() fonksiyonları ile veri tipi öğrenilebilir.

```
# numeric

a <- 3.5
class(a)
```

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

```
typeof(a) # typeof numeriklerin tipini double olarak gösterir.
```

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

```
is.numeric(a) # verinin tipinin numerik olup olmadığı sorgulanır.
```

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

```
# integer

b <- 5
class(b)
```

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

```
is.integer(b)
```

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

```
c <- 6L # integer olması için sayının sağına L yazılır.
class(c)
```

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

```
is.integer(c)
```

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

```
class(as.integer(b)) # as. ile baslayan fonksiyonlar dönüşüm için kullanılır.
```

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

```
# complex

z <- 4 + 2i
class(z)
```

```
## [1] "complex"
```

```
# character

d <- "R Programlama"
class(d)
```

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

```r
e <- "5.5"
class(e)
```

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

```r
class(as.numeric(e))
```

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

```r
# logical

x <- TRUE ; y <- FALSE
class(c(x,y))
```

```
## [1] "logical"
```

```r
as.integer(c(x,y)) # TRUE ve FALSE numeric olarak 1 ve 0 değerine karşılık gelir.
```

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

3.1 Vektörler

- R'daki en temel nesneler vektörlerdir.
- Vektörler homojen yapıya sahiptir yani bütün elemanları aynı veri tipinde olmalıdır.
- Vektörler tek boyutludur.
- Bir vektör oluşturmak için kullanabilecek en temel fonksiyon **c()**'dir.

```r
v <- c(1,4,7,2,5,8,3,6,9)

v[1] # 1. elemanını seçer
```

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

```r
v[3] # 3. elemanını seçer
```

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

```r
v[c(3,7)] # 3. ve 7. elemani secer
```

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

```r
v[1:6] # 1. elemandan 6. elemana kadar secer
```

```
## [1] 1 4 7 2 5 8
```

```r
v[-2] # 2. elemani haric tutarak secer
```

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

```r
length(v) # vektörün uzunluğunu gösterir
```

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

```r
v2 <- c(v,12) # vektöre eleman ekleme
v2
```

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

```r
# : ile başlangıç ve bitiş değerleri belli olan vektörler yaratılabilir.

v3 <- 1:10
v3
```

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

```r
v4 <- 11:20
v4
```

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

```r
# Vektörler ile matematiksel işlemler yapılabilir.

v3 + v4
```

```
##  [1] 12 14 16 18 20 22 24 26 28 30
```

```
v3 / v4
```

```
##  [1] 0.09090909 0.16666667 0.23076923 0.28571429 0.33333333 0.37500000
##  [7] 0.41176471 0.44444444 0.47368421 0.50000000
```

```
2 * v3 - v4
```

```
##  [1] -9 -8 -7 -6 -5 -4 -3 -2 -1  0
```

```
# Vektörler ile ilgili kullanılabilecek bazı fonksiyonlar

# seq ()
#aritmetik bir diziden meydana gelen bir vektör oluşturmak için kullanılır.
seq(from = 5, to = 50, by =5) # 5 ile başlayan 50 ile biten 5şer artan vektör
```

```
##  [1]  5 10 15 20 25 30 35 40 45 50
```

```
seq(from = 5, to = 50, length = 7) # 5 ile başlayan 50 ile 7 elemanlı vektör
```

```
## [1]  5.0 12.5 20.0 27.5 35.0 42.5 50.0
```

```
seq(5,1,-1) # 5 ile baslayıp 1'e kadar 1'er azaltarak vektor olusturma
```

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

```
# rep()
# tekrarlı sayılar içeren vektörler oluşturulur.
rep(10,8) # 8 tane 10 değeri olan vektör
```

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

```
rep(c(1,2,3),4) # 1,2,3 vekrünün 4 defa tekrarlanması
```

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

```
rep(c(1,2,3), each = 4) # each argünmanı ile sıralı ve tekrarlı vektör
```

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

```r
# rev()
v5 <- c(3,5,6,1,NA,12,NA,8,9) # R'da NA boş gözlemi ifade eder.
rev(v5) # vektörü tersine çevirir
```

```
## [1]  9  8 NA 12 NA  1  6  5  3
```

```r
# rank()
rank(v5) # elemanların sıra numarasını verir
```

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

```r
rank(v5, na.last = TRUE) # NA leri son sıraya atar.
```

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

```r
rank(v5, na.last = FALSE) # NA leri en başa koyar.
```

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

```r
rank(v5,na.last = NA) # NA değerlere yer verilmez
```

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

```r
rank(v5, na.last = "keep") # NA değerler oldukları gibi görünürler.
```

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

```r
# all()
all(v5>5) # vektördeki tüm elemanların şartı sağlayıp sağlamadıkları test edilir.
```

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

```r
all(v5>0) # vektörde NA varsa sonuç NA döner
```

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

```
all(v5>0, na.rm = TRUE) # NA gözlemler hariç tutularak sonuç üretir.
```

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

```
# any()
any(v5>6) # vektördeki en az bir elemanın şartı sağlayıp sağlamadığı test edilir.
```

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

```
any(v5==9)
```

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

```
# unique()
v6 <- rep(1:5,3)
v6
```

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

```
unique(v6) # tekrarlı gözlemler temizlenir
```

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

```
# duplicated()
duplicated(v6) # tekrarlı gözlemlerin varlığını kontrol eder
```

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

```
v6[duplicated(v6)] # tekrarlı gözlemleri listeler
```

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

```
# sort()
sort(v5) # küçükten büyüğe sıralama yapar.
```

```
## [1]  1  3  5  6  8  9 12
```

```
sort(v5,decreasing = TRUE) # azalan sırada sıralama yapar.
```

```
## [1] 12  9  8  6  5  3  1
```

```
# diff()
diff(v5) # vektörde ardışık elemanlar arasındaki farkı bulur.
```

```
## [1]  2  1 -5 NA NA NA NA  1
```

```
diff(na.omit(v5)) # na.omit vektördeki NA gözlemleri temizler
```

```
## [1]  2  1 -5 11 -4  1
```

```
# is.na()
is.na(v5) # vektördeki elamanların NA olup olmadığını test eder.
```

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

```
is.nan(v5) # NaN aynı zamanda bir NA'dir.
```

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

```
# which
which(v5==12) # 6 sayısının posizyonunu gösterir
```

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

```
which.max(v5) # vektördeki maximum elemanın posizyonunu gösterir
```

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

```
which.min(v5) # vektördeki minimum elemanın posizyonunu gösterir
```

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

```r
v5[which.min(v5)] # vektördeki minimum elemanı gösterir
```

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

```r
# Temel İstatistiksel Fonksiyonlar
mean(v5) # NA varsa sonuç NA döner
```

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

```r
mean(v5, na.rm = TRUE) # aritmetik ortalama
```

```
## [1] 6.285714
```

```r
median(v5,na.rm = TRUE) # medyan (ortanca)
```

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

```r
sum(v5,na.rm = TRUE) # vektör toplamını verir
```

```
## [1] 44
```

```r
min(v5,na.rm = TRUE) # vektörün minimum değeri
```

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

```r
max(v5,na.rm = TRUE) # vektörün maximum değeri
```

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

```r
sd(v5,na.rm = TRUE) # standart sapma
```

```
## [1] 3.728909
```

```r
var(v5,na.rm = TRUE) # varyans
```

```
## [1] 13.90476
```

3.2 Matrisler

- Matrisler, iki boyutlu yani satır ve sütunları olan atomik vektörlerdir.
- **matrix()** fonksiyonu ile tanımlanmaktadır.
- Vektörlerin birleştirlmesi ile de matrisler oluşturulabilir. **rbind**satır bazlı alt alta birleştirme, **cbind** ise sütun bazlı yanyana birleştirme yapar. Burada vektörlerin aynı boyutlarda olmasına dikkat edilmesi gerekir.

```
v1 <- c(3,4,6,8,5)
v2 <- c(4,8,4,7,1)
v3 <- c(2,2,5,4,6)
v4 <- c(4,7,5,2,5)

matris <- cbind(v1, v2, v3, v4)
matris
```

```
##      v1 v2 v3 v4
## [1,]  3  4  2  4
## [2,]  4  8  2  7
## [3,]  6  4  5  5
## [4,]  8  7  4  2
## [5,]  5  1  6  5
```

```
is.matrix(matris)
```

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

```
dim(matris)
```

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

```
matrix(nrow = 3, ncol = 3, 1:9)
```

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

```r
matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE) # byrow satırlara göre oluşturur.
```

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

```r
mat <- seq(3, 21, by = 2)
mat
```

```
##  [1]  3  5  7  9 11 13 15 17 19 21
```

```r
dim(mat) <- c(5,2)
mat
```

```
##      [,1] [,2]
## [1,]    3   13
## [2,]    5   15
## [3,]    7   17
## [4,]    9   19
## [5,]   11   21
```

```r
matrix(c(1,2,3,11,22,33), nrow = 2, ncol = 3, byrow = TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]   11   22   33
```

```r
# normal dağılımdan 0 ortamalı, 1 standart sapmalı 16 sayı üret
MA <- rnorm(16, 0, 1)
MA <- matrix(MA, nrow = 4, ncol = 4)

# normal dağılımdan 90 ortamalı, 10 standart sapmalı 16 sayı üret
MB <- rnorm(16, 90, 10)
MB <- matrix(MB, nrow = 4, ncol = 4)

m <- rbind(MA, MB)
m
```

```
##             [,1]        [,2]        [,3]         [,4]
## [1,]   0.8771499  0.01025206   0.8656115   0.03545045
## [2,]   0.6079894  0.43932790  -1.5522166   1.69968170
## [3,]   2.0249507 -1.27834411  -1.9663289   0.10415966
## [4,]  -0.5884568  1.35752958   0.2610880  -0.32178935
## [5,]  85.8638476 80.35106896  99.3217468  74.20318464
## [6,]  93.6194196 96.12590917 102.1571912  96.88575146
## [7,]  98.7166736 94.12963977  89.2320013  76.32134050
## [8,] 103.1641619 75.02276561  94.3385921 103.91367826
```

```
# satır ve sütun isimlendirme
colnames(m) <- LETTERS[1:4]
rownames(m) <- tail(LETTERS,8)
m
```

```
##             A           B           C            D
## S   0.8771499  0.01025206   0.8656115   0.03545045
## T   0.6079894  0.43932790  -1.5522166   1.69968170
## U   2.0249507 -1.27834411  -1.9663289   0.10415966
## V  -0.5884568  1.35752958   0.2610880  -0.32178935
## W  85.8638476 80.35106896  99.3217468  74.20318464
## X  93.6194196 96.12590917 102.1571912  96.88575146
## Y  98.7166736 94.12963977  89.2320013  76.32134050
## Z 103.1641619 75.02276561  94.3385921 103.91367826
```

```
#Matris Elemanlarina Erismek
m[1,1] # 1. satır, 1.sütundak, eleman
```

```
## [1] 0.8771499
```

```
m[4,2] # 4. satır, 2.sütundak, eleman
```

```
## [1] 1.35753
```

```
m[,2] # 2. sütun elemanları
```

```
##           S           T           U           V           W           X
##  0.01025206  0.43932790 -1.27834411  1.35752958 80.35106896 96.12590917
##           Y           Z
## 94.12963977 75.02276561
```

```
m[-3,] # 3. satır hariç tüm elemanlar
```

```
##             A           B           C            D
## S   0.8771499  0.01025206   0.8656115   0.03545045
## T   0.6079894  0.43932790  -1.5522166   1.69968170
## V  -0.5884568  1.35752958   0.2610880  -0.32178935
## W  85.8638476 80.35106896  99.3217468  74.20318464
## X  93.6194196 96.12590917 102.1571912  96.88575146
## Y  98.7166736 94.12963977  89.2320013  76.32134050
## Z 103.1641619 75.02276561  94.3385921 103.91367826
```

```
# köşegen matris oluşturma
diag(2,nrow=3)
```

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

```
diag(1,5) # 5*5 birim matris
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```
# transpose
t(m)
```

```
##             S          T          U          V        W         X        Y
## A 0.87714986  0.6079894  2.0249507 -0.5884568 85.86385  93.61942 98.71667
## B 0.01025206  0.4393279 -1.2783441  1.3575296 80.35107  96.12591 94.12964
## C 0.86561153 -1.5522166 -1.9663289  0.2610880 99.32175 102.15719 89.23200
## D 0.03545045  1.6996817  0.1041597 -0.3217894 74.20318  96.88575 76.32134
##           Z
## A 103.16416
## B  75.02277
## C  94.33859
## D 103.91368
```

```r
# matris ile işlemler

m1 <- matrix(1:4,nrow=2)
m2 <- matrix(5:8,nrow=2)

m1;m2
```

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

```
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```r
m1 + m2 # matris elemanları birebir toplanır
```

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

```r
m1 / m2 # matris elemanları birebir toplanır
```

```
##           [,1]      [,2]
## [1,] 0.2000000 0.4285714
## [2,] 0.3333333 0.5000000
```

```r
m1 * m2 # matris elemanları birebir çarpılır
```

```
##      [,1] [,2]
## [1,]    5   21
## [2,]   12   32
```

```r
m1 %*% m2 # matris çarpımı
```

```
##      [,1] [,2]
## [1,]   23   31
## [2,]   34   46
```

```
solve(m2) # matrisin tersi
```

```
##      [,1] [,2]
## [1,]   -4  3.5
## [2,]    3 -2.5
```

```
rowSums(m1) # satır toplamları
```

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

```
rowMeans(m1) # satır ortalaması
```

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

```
colSums(m1) # sütun toplamları
```

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

```
colMeans(m1) # sütun ortalaması
```

```
## [1] 1.5 3.5
```

3.3 Listeler

- Listeler, birbirinden farklı veri tiplerine sahip vektörler, matrisler vb farklı objeleri birarada tutabilen yapılardır.
- **list()** ile liste oluşturulur.

```
x <- c(3,5,7)
y <- letters[1:10]
z <- c(rep(TRUE,3),rep(FALSE,4))

list <- list(x,y,z)
list
```

```
## [[1]]
## [1] 3 5 7
##
## [[2]]
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
##
## [[3]]
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
class(list) # listenin sınıfını verir
```

```
## [1] "list"
```

```
str(list) # listenin yapısını verir
```

```
## List of 3
##  $ : num [1:3] 3 5 7
##  $ : chr [1:10] "a" "b" "c" "d" ...
##  $ : logi [1:7] TRUE TRUE TRUE FALSE FALSE FALSE ...
```

```
names(list) <- c("numeric","karakter","mantıksal") # liste isimlendirme
list
```

```
## $numeric
## [1] 3 5 7
##
## $karakter
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
##
## $mantıksal
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
list$numeric
```

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

```
list$karakter
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
list$mantıksal
```

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

```
list[[2]]
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
list$numeric2 <- c(4:10) # listeye eleman ekleme
list
```

```
## $numeric
## [1] 3 5 7
##
## $karakter
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
##
## $mantıksal
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
##
## $numeric2
## [1]  4  5  6  7  8  9 10
```

```
list$numeric <- NULL # listeden eleman silme
list
```

```
## $karakter
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
##
## $mantıksal
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
##
## $numeric2
## [1]  4  5  6  7  8  9 10
```

```
unlist(list) # listeyi vektöre çevirir.
```

```
##  karakter1  karakter2  karakter3  karakter4  karakter5  karakter6  karakter7
##        "a"        "b"        "c"        "d"        "e"        "f"        "g"
##  karakter8  karakter9 karakter10 mantıksal1 mantıksal2 mantıksal3 mantıksal4
##        "h"        "i"        "j"     "TRUE"     "TRUE"     "TRUE"    "FALSE"
## mantıksal5 mantıksal6 mantıksal7  numeric21  numeric22  numeric23  numeric24
##    "FALSE"    "FALSE"    "FALSE"        "4"        "5"        "6"        "7"
##  numeric25  numeric26  numeric27
##        "8"        "9"       "10"
```

3.4 Dataframe

Veri çerçevesi, her sütunun bir değişkenin değerlerini ve her satırın her sütundan bir değer kümesini içerdiği bir tablo veya iki boyutlu dizi benzeri bir yapıdır. Bir veri çerçevesinin özellikleri şunlardır:

- Sütun adları boş olmamalıdır.
- Satır adları benzersiz olmalıdır.
- Bir veri çerçevesinde saklanan veriler sayısal, faktör veya karakter tipinde olabilir.
- Her sütun aynı sayıda veri öğesi içermelidir.

data.frame() fonksiyonunu uygulayarak bir veri çerçevesi oluşturabiliriz.

```
# data.frame oluşturma
set.seed(12345)

data <- data.frame(
  row_num = 1:20,
  col1 = rnorm(20),
  col2 = runif(20), # uniform dağılımdam 20 gözlem üret
  col3 = rbinom(20,size=5,prob = 0.5), # binom dağılımdam 20 gözlem üret
  col4 = sample(c("TRUE","FALSE"),20,replace = TRUE),
  col5 = sample(c(rep(c("E","K"),8),rep(NA,4))),
  stringsAsFactors = TRUE # karakter olanlar faktör olarak değerlendirilsin
)

class(data)
```

```
## [1] "data.frame"
```

```
head(data) # ilk 6 gözlemi gösterir
```

```
##   row_num       col1      col2 col3  col4 col5
## 1       1  0.5855288 0.7821933    3 FALSE    E
## 2       2  0.7094660 0.4291988    2  TRUE    E
## 3       3 -0.1093033 0.9272740    5  TRUE    E
## 4       4 -0.4534972 0.7732432    3 FALSE    K
## 5       5  0.6058875 0.2596812    5  TRUE    E
## 6       6 -1.8179560 0.3212247    2  TRUE <NA>
```

```
tail(data) # son 6 gözlemi gösterir
```

```
##    row_num       col1       col2 col3  col4 col5
## 15      15 -0.7505320 0.73249612    1 FALSE    K
## 16      16  0.8168998 0.49924102    3 FALSE    K
## 17      17 -0.8863575 0.72977197    4 FALSE    K
## 18      18 -0.3315776 0.08033604    3  TRUE <NA>
## 19      19  1.1207127 0.43553048    3 FALSE    K
## 20      20  0.2987237 0.23658045    1 FALSE    E
```

```
tail(data,10) # son 10 gözlemi gösterir
```

```
##    row_num       col1       col2 col3  col4 col5
## 11      11 -0.1162478 0.96447029    3  TRUE    K
## 12      12  1.8173120 0.82730287    3  TRUE    E
## 13      13  0.3706279 0.31502824    2 FALSE <NA>
## 14      14  0.5202165 0.21302545    2  TRUE    K
## 15      15 -0.7505320 0.73249612    1 FALSE    K
## 16      16  0.8168998 0.49924102    3 FALSE    K
## 17      17 -0.8863575 0.72977197    4 FALSE    K
## 18      18 -0.3315776 0.08033604    3  TRUE <NA>
## 19      19  1.1207127 0.43553048    3 FALSE    K
## 20      20  0.2987237 0.23658045    1 FALSE    E
```

```
str(data) # tablonun yapısını gösterir
```

```
## 'data.frame':    20 obs. of  6 variables:
##  $ row_num: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ col1   : num  0.586 0.709 -0.109 -0.453 0.606 ...
##  $ col2   : num  0.782 0.429 0.927 0.773 0.26 ...
##  $ col3   : int  3 2 5 3 5 2 4 1 3 4 ...
##  $ col4   : Factor w/ 2 levels "FALSE","TRUE": 1 2 2 1 2 2 2 1 2 2 ...
##  $ col5   : Factor w/ 2 levels "E","K": 1 1 1 2 1 NA 1 NA 2 1 ...
```

```
summary(data) # tablonun özet istatistiklerini gösterir
```

```
##     row_num           col1                col2              col3         col4
##  Min.   : 1.00   Min.   :-1.81796   Min.   :0.04346   Min.   :1.00   FALSE: 9
##  1st Qu.: 5.75   1st Qu.:-0.36206   1st Qu.:0.23069   1st Qu.:2.00   TRUE :11
##  Median :10.50   Median : 0.09471   Median :0.43236   Median :3.00
##  Mean   :10.50   Mean   : 0.07652   Mean   :0.46554   Mean   :2.85
##  3rd Qu.:15.25   3rd Qu.: 0.61194   3rd Qu.:0.74268   3rd Qu.:3.25
##  Max.   :20.00   Max.   : 1.81731   Max.   :0.96447   Max.   :5.00
##    col5
##  E   :8
##  K   :8
##  NA's:4
##
##
##
```

```r
# veri çerçevesinden belirli sütun/ları seçmek için $ veya [] kullanılır.
head(data$col1)
```

```
## [1]  0.5855288  0.7094660 -0.1093033 -0.4534972  0.6058875 -1.8179560
```

```r
head(data[,"col1"])
```

```
## [1]  0.5855288  0.7094660 -0.1093033 -0.4534972  0.6058875 -1.8179560
```

```r
# veri çerçevesinden belirli satır/ları seçmek için [] kullanılır.
data[1:2,]
```

```
##   row_num      col1      col2 col3  col4 col5
## 1       1 0.5855288 0.7821933    3 FALSE    E
## 2       2 0.7094660 0.4291988    2  TRUE    E
```

```r
# 3. and 5. satır ile 2. ve 4. kolon
data[c(3,5),c(2,4)]
```

```
##         col1 col3
## 3 -0.1093033    5
## 5  0.6058875    5
```

```r
# koşula göre veriler seçilebilir
data$row_num > 12 # TRUE veya FALSE döner
```

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

```r
data[data$row_num > 12,] # koşula göre tablonu değerleri döner
```

```
##    row_num       col1       col2 col3  col4 col5
## 13      13  0.3706279 0.31502824    2 FALSE <NA>
## 14      14  0.5202165 0.21302545    2  TRUE    K
## 15      15 -0.7505320 0.73249612    1 FALSE    K
## 16      16  0.8168998 0.49924102    3 FALSE    K
## 17      17 -0.8863575 0.72977197    4 FALSE    K
## 18      18 -0.3315776 0.08033604    3  TRUE <NA>
## 19      19  1.1207127 0.43553048    3 FALSE    K
## 20      20  0.2987237 0.23658045    1 FALSE    E
```

```
# subset ile tablo filtrelenebilir.
subset(data,
       row_num >= 10 & col4 == 'TRUE',
       select = c(row_num, col1, col2,col4))
```

```
##    row_num       col1       col2 col4
## 10      10 -0.9193220 0.62554280 TRUE
## 11      11 -0.1162478 0.96447029 TRUE
## 12      12  1.8173120 0.82730287 TRUE
## 14      14  0.5202165 0.21302545 TRUE
## 18      18 -0.3315776 0.08033604 TRUE
```

```
# names veya colnames ile sütun isimleri elde edilir.
names(data)
```

```
## [1] "row_num" "col1"    "col2"    "col3"    "col4"    "col5"
```

```
colnames(data)
```

```
## [1] "row_num" "col1"    "col2"    "col3"    "col4"    "col5"
```

```
# vektör ile sütun seçimi
cols <- c("col1","col2","col5")
head(data[cols])
```

```
##         col1      col2 col5
## 1  0.5855288 0.7821933    E
## 2  0.7094660 0.4291988    E
## 3 -0.1093033 0.9272740    E
## 4 -0.4534972 0.7732432    K
## 5  0.6058875 0.2596812    E
## 6 -1.8179560 0.3212247 <NA>
```

```
# sütun silme
data$col1 <- NULL
head(data)
```

```
##   row_num      col2 col3  col4 col5
## 1       1 0.7821933    3 FALSE    E
## 2       2 0.4291988    2  TRUE    E
## 3       3 0.9272740    5  TRUE    E
## 4       4 0.7732432    3 FALSE    K
## 5       5 0.2596812    5  TRUE    E
## 6       6 0.3212247    2  TRUE <NA>
```

```r
# sütun ekleme
data$col1 <- rnorm(20)
head(data)
```

```
##   row_num      col2 col3  col4 col5       col1
## 1       1 0.7821933    3 FALSE    E  0.4768080
## 2       2 0.4291988    2  TRUE    E  0.8424486
## 3       3 0.9272740    5  TRUE    E -0.8903234
## 4       4 0.7732432    3 FALSE    K  0.7529609
## 5       5 0.2596812    5  TRUE    E  0.4452159
## 6       6 0.3212247    2  TRUE <NA>  0.4211062
```

```r
# sütunları sıralama
head(data[c("row_num","col1","col2","col3","col4","col5")])
```

```
##   row_num       col1      col2 col3  col4 col5
## 1       1  0.4768080 0.7821933    3 FALSE    E
## 2       2  0.8424486 0.4291988    2  TRUE    E
## 3       3 -0.8903234 0.9272740    5  TRUE    E
## 4       4  0.7529609 0.7732432    3 FALSE    K
## 5       5  0.4452159 0.2596812    5  TRUE    E
## 6       6  0.4211062 0.3212247    2  TRUE <NA>
```

```r
# sıralama
head(data[order(data$col3),]) # artan
```

```
##    row_num       col2 col3  col4 col5         col1
## 8        8 0.04345645    1 FALSE <NA> -0.896320181
## 15      15 0.73249612    1 FALSE    K  0.148543198
## 20      20 0.23658045    1 FALSE    E  0.240173186
## 2        2 0.42919882    2  TRUE    E  0.842448636
## 6        6 0.32122467    2  TRUE <NA>  0.421106220
## 13      13 0.31502824    2 FALSE <NA> -0.008925433
```

```r
head(data[order(-data$row_num),]) # azalan
```

```
##    row_num       col2 col3  col4 col5       col1
## 20      20 0.23658045    1 FALSE    E  0.2401732
## 19      19 0.43553048    3 FALSE    K  0.2583817
## 18      18 0.08033604    3  TRUE <NA> -0.1712566
## 17      17 0.72977197    4 FALSE    K  0.7884411
## 16      16 0.49924102    3 FALSE    K -0.3798679
## 15      15 0.73249612    1 FALSE    K  0.1485432
```

```
head(data[order(data$col3,-data$row_num),])
```

```
##    row_num       col2 col3  col4 col5         col1
## 20      20 0.23658045    1 FALSE    E  0.240173186
## 15      15 0.73249612    1 FALSE    K  0.148543198
## 8        8 0.04345645    1 FALSE <NA> -0.896320181
## 14      14 0.21302545    2  TRUE    K -0.326216850
## 13      13 0.31502824    2 FALSE <NA> -0.008925433
## 6        6 0.32122467    2  TRUE <NA>  0.421106220
```

```
# kayıp gözlemler (missing values)
tail(is.na(data))
```

```
##       row_num  col2  col3  col4  col5  col1
## [15,]   FALSE FALSE FALSE FALSE FALSE FALSE
## [16,]   FALSE FALSE FALSE FALSE FALSE FALSE
## [17,]   FALSE FALSE FALSE FALSE FALSE FALSE
## [18,]   FALSE FALSE FALSE FALSE  TRUE FALSE
## [19,]   FALSE FALSE FALSE FALSE FALSE FALSE
## [20,]   FALSE FALSE FALSE FALSE FALSE FALSE
```

```
tail(is.na(data$col5))
```

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

```
data[is.na(data$col5),]
```

```
##    row_num       col2 col3  col4 col5         col1
## 6        6 0.32122467    2  TRUE <NA>  0.421106220
## 8        8 0.04345645    1 FALSE <NA> -0.896320181
## 13      13 0.31502824    2 FALSE <NA> -0.008925433
## 18      18 0.08033604    3  TRUE <NA> -0.171256569
```

```
data[!is.na(data$col5),]
```

```
##    row_num       col2 col3  col4 col5       col1
## 1        1 0.78219328    3 FALSE    E  0.4768080
## 2        2 0.42919882    2  TRUE    E  0.8424486
## 3        3 0.92727397    5  TRUE    E -0.8903234
## 4        4 0.77324322    3 FALSE    K  0.7529609
## 5        5 0.25968125    5  TRUE    E  0.4452159
```

```
## 7        7 0.06019516    4  TRUE    E  1.1495922
## 9        9 0.05505382    3  TRUE    K  0.8696714
## 10      10 0.62554280    4  TRUE    E  0.5059117
## 11      11 0.96447029    3  TRUE    K  0.3317020
## 12      12 0.82730287    3  TRUE    E  1.7399997
## 14      14 0.21302545    2  TRUE    K -0.3262169
## 15      15 0.73249612    1 FALSE    K  0.1485432
## 16      16 0.49924102    3 FALSE    K -0.3798679
## 17      17 0.72977197    4 FALSE    K  0.7884411
## 19      19 0.43553048    3 FALSE    K  0.2583817
## 20      20 0.23658045    1 FALSE    E  0.2401732
```

```
rowSums(is.na(data)) # satılardaki toplam kayıp gözlem sayısı
```

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

```
colSums(is.na(data)) # sütunlardaki toplam kayıp gözlem sayısı
```

```
## row_num    col2    col3    col4    col5    col1
##       0       0       0       0       4       0
```

```
sum(is.na(data)) # tablodaki toplam kayıp gözlem sayısı
```

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

```
complete.cases(data) # satırlarda eksik gözlemlerin durumu
```

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

```
data[complete.cases(data),]
```

```
##    row_num       col2 col3  col4 col5       col1
## 1        1 0.78219328    3 FALSE    E  0.4768080
## 2        2 0.42919882    2  TRUE    E  0.8424486
## 3        3 0.92727397    5  TRUE    E -0.8903234
## 4        4 0.77324322    3 FALSE    K  0.7529609
## 5        5 0.25968125    5  TRUE    E  0.4452159
## 7        7 0.06019516    4  TRUE    E  1.1495922
## 9        9 0.05505382    3  TRUE    K  0.8696714
## 10      10 0.62554280    4  TRUE    E  0.5059117
```

```
## 11      11 0.96447029    3  TRUE    K  0.3317020
## 12      12 0.82730287    3  TRUE    E  1.7399997
## 14      14 0.21302545    2  TRUE    K -0.3262169
## 15      15 0.73249612    1 FALSE    K  0.1485432
## 16      16 0.49924102    3 FALSE    K -0.3798679
## 17      17 0.72977197    4 FALSE    K  0.7884411
## 19      19 0.43553048    3 FALSE    K  0.2583817
## 20      20 0.23658045    1 FALSE    E  0.2401732
```

```
data[!complete.cases(data),]
```

```
##    row_num       col2 col3  col4 col5         col1
## 6        6 0.32122467    2  TRUE <NA>  0.421106220
## 8        8 0.04345645    1 FALSE <NA> -0.896320181
## 13      13 0.31502824    2 FALSE <NA> -0.008925433
## 18      18 0.08033604    3  TRUE <NA> -0.171256569
```

```
na.omit(data) # NA olan satırları siler.
```

```
##    row_num       col2 col3  col4 col5       col1
## 1        1 0.78219328    3 FALSE    E  0.4768080
## 2        2 0.42919882    2  TRUE    E  0.8424486
## 3        3 0.92727397    5  TRUE    E -0.8903234
## 4        4 0.77324322    3 FALSE    K  0.7529609
## 5        5 0.25968125    5  TRUE    E  0.4452159
## 7        7 0.06019516    4  TRUE    E  1.1495922
## 9        9 0.05505382    3  TRUE    K  0.8696714
## 10      10 0.62554280    4  TRUE    E  0.5059117
## 11      11 0.96447029    3  TRUE    K  0.3317020
## 12      12 0.82730287    3  TRUE    E  1.7399997
## 14      14 0.21302545    2  TRUE    K -0.3262169
## 15      15 0.73249612    1 FALSE    K  0.1485432
## 16      16 0.49924102    3 FALSE    K -0.3798679
## 17      17 0.72977197    4 FALSE    K  0.7884411
## 19      19 0.43553048    3 FALSE    K  0.2583817
## 20      20 0.23658045    1 FALSE    E  0.2401732
```

3.5 Faktörler

- Faktörler, verileri kategorilere ayırmak ve düzeyler halinde depolamak için kullanılan veri nesneleridir. Hem karakter hem de tam sayıları depolayabilirler.
- "Erkek,"Kadın" ve Doğru, Yanlış vb. gibi istatistiksel modelleme için veri analizinde faydalıdırlar.

- Faktörler, girdi olarak bir vektör alınarak faktör () işlevi kullanılarak oluşturulur.

```
data <- c(rep("erkek",5),rep("kadın",7))
print(data)
```

```
##  [1] "erkek" "erkek" "erkek" "erkek" "erkek" "kadın" "kadın" "kadın" "kadın"
## [10] "kadın" "kadın" "kadın"
```

```
is.factor(data)
```

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

```
# veriyi faktöre çevirme
factor_data <- factor(data)

print(factor_data)
```

```
##  [1] erkek erkek erkek erkek erkek kadın kadın kadın kadın kadın kadın kadın
## Levels: erkek kadın
```

```
print(is.factor(factor_data))
```

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

```
as.numeric(factor_data)
```

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

```
# data frame için vektörler oluşturalım
boy <- c(132,151,162,139,166,147,122)
kilo <- c(48,49,66,53,67,52,40)
cinsiyet <- c("erkek","erkek","kadın","kadın","erkek","kadın","erkek")

# data frame
df <- data.frame(boy,kilo,cinsiyet)
str(df)
```

```
## 'data.frame':    7 obs. of  3 variables:
##  $ boy     : num  132 151 162 139 166 147 122
##  $ kilo    : num  48 49 66 53 67 52 40
##  $ cinsiyet: chr  "erkek" "erkek" "kadın" "kadın" ...
```

```
df$cinsiyet <- factor(cinsiyet)
str(df)
```

```
## 'data.frame':    7 obs. of  3 variables:
##  $ boy     : num  132 151 162 139 166 147 122
##  $ kilo    : num  48 49 66 53 67 52 40
##  $ cinsiyet: Factor w/ 2 levels "erkek","kadın": 1 1 2 2 1 2 1
```

```
print(is.factor(df$cinsiyet))
```

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

```
# cinsiyet kolononun seviyeleri
print(df$cinsiyet)
```

```
## [1] erkek erkek kadın kadın erkek kadın erkek
## Levels: erkek kadın
```

```
# seviyelerin sırası değiştirilebilir.

df2 <- c(rep("düşük",4),rep("orta",5),rep("yüksek",2))

factor_df2 <- factor(df2)
print(factor_df2)
```

```
##  [1] düşük düşük düşük düşük orta  orta  orta  orta  orta  yüksek
## [11] yüksek
## Levels: düşük orta yüksek
```

```
order_df2 <- factor(factor_df2,levels = c("yüksek","orta","düşük"))
print(order_df2)
```

```
##  [1] düşük düşük düşük düşük orta  orta  orta  orta  orta  yüksek
## [11] yüksek
## Levels: yüksek orta düşük
```

```
# ordered=TRUE ile seviyelerin sıralı olduğu ifade edilir
order_df2 <- factor(factor_df2,levels = c("yüksek","orta","düşük"),ordered = TRUE)
print(order_df2)
```

```
##  [1] düşük düşük düşük düşük orta  orta  orta  orta  orta  yüksek
## [11] yüksek
## Levels: yüksek < orta < düşük
```

```
# Faktör seviyesi üretme

# gl() fonksiyonunu kullanarak faktör seviyeleri üretebiliriz.
# Girdi olarak kaç seviye ve her seviyeden kaç tane sayı oalcağı belirtilir.

faktor <- gl(n=3, k=4, labels = c("level1", "level2","level3"),ordered = TRUE)
print(faktor)
```

```
##  [1] level1 level1 level1 level1 level2 level2 level2 level2 level3 level3
## [11] level3 level3
## Levels: level1 < level2 < level3
```

4 Fonksiyonlar

Fonksiyonlar çoğu programlama dillerinin çok önemli bir özelliğidir. Yalnızca mevcut fonksiyonları kullanmak yerine, belirli işleri yapmak için kendimize ait fonksiyonlar yazabiliriz. Ama neden fonksiyon yazmalıyız?

- Tekrarlardan kaçınmanızı sağlar.
- Yeniden kullanımı kolaylaştırır.
- Karmaşık komut dosyalarından kaçınmanıza yardımcı olur.
- Hata ayıklamayı kolaylaştırır.

Bir fonksiyonun temel kod yapısı aşağıdak gibidir:

```
function_name <- function(arg_1, arg_2, ...) {
   Function body
}
```

```
# kare alma fonksiyonu
f_kare <- function(x) {
   x^2
 }

f_kare(15)
```

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

```
f_kare(x=20)
```

```
## [1] 400
```

```
# standart sapma fonksiyonu

# Standart sapmanın hesaplanması
# sqrt(sum((x - mean(x))^2) / (length(x) - 1))

set.seed(123) # Pseudo-randomization
x1 <- rnorm(1000, 0, 1.0)
x2 <- rnorm(1000, 0, 1.5)
x3 <- rnorm(1000, 0, 5.0)

# her serinin ayrı ayrı standart sapmasının hesaplanması
sd1 <- sqrt(sum((x1 - mean(x1))^2) / (length(x1) - 1))
sd2 <- sqrt(sum((x2 - mean(x2))^2) / (length(x2) - 1))
sd3 <- sqrt(sum((x3 - mean(x1))^2) / (length(x3) - 1))
c(sd1 = sd1, sd2 = sd2, sd3 = sd3)
```

```
##      sd1      sd2      sd3
## 0.991695 1.514511 4.893180
```

```
# fonksiyonu oluşturalım
f_sd <- function(x) {
  result <- sqrt(sum((x - mean(x))^2) / (length(x) - 1))
  return(result)
}

sd1 <- f_sd(x1)
sd2 <- f_sd(x2)
sd3 <- f_sd(x3)
c(sd1 = sd1, sd2 = sd2, sd3 = sd3)
```

```
##      sd1      sd2      sd3
## 0.991695 1.514511 4.891787
```

```
# standartlaştırma fonksiyonu
f_std <- function(x) {
m <- mean(x)
```

```
s <- sd(x)
(x - m) / s
}

x4 <- rnorm(10,5,10)
x4
```

```
##  [1]  3.496925  1.722429 -9.481653 -1.972846 30.984902  4.625850 14.134919
##  [8]  3.154735 11.098243  4.472732
```

```
f_std(x4)
```

```
##  [1] -0.2517201 -0.4155359 -1.4498610 -0.7566719  2.2858821 -0.1475014
##  [7]  0.7303455 -0.2833100  0.4500093 -0.1616367
```

5 Kontrol İfadeleri

Kontrol ifadeleri ve döngüler R içerisinde sıklıkla kullanılan yapılardır. Belirli şartlara bağlı olan ya da tekrarlı işlemler için oldukça faydalıdırlar. R programlama dilinde en çok kullanılan **if-else, for, while, next, break** gibi kontrol döngüleridir.

5.1 if-else

Bu kombinasyon R'de en sık kullanılan kontrol yapılarındandır. Bu yapıda, bir koşulu test edebilir ve doğru veya yanlış olmasına bağlı olarak ona göre hareket edebilirsiniz. if-else kombinasyonlarında aşağıdaki yapılar kullanılmaktadır.

```
if (condition){
#do something if condition is true
}


if (condition){
#do something if condition is true
}
else{
#do someting if condition is not true
}

if (condition){
```

```
#do something if condition is true

} else if (condition2) {

#do someting if condition2 is true

} else {

#do something if neither condition 1 nor condition 2 is true

}
```

```
x <- 8

if (x < 10) {
    print("x 10'dan küçüktür")
} else {
    print("x 10'dan büyüktür ya da 10'a eşittir")
}
```

```
## [1] "x 10'dan küçüktür"
```

```
# ifelse
# ifelse(condition, do_if_true, do_if_false)
df <- data.frame(value = 1:9)
df$group <- ifelse(df$value <= 3,1,ifelse(df$value > 3 & df$value <= 6,2,3))
df
```

```
##   value group
## 1     1     1
## 2     2     1
## 3     3     1
## 4     4     2
## 5     5     2
## 6     6     2
## 7     7     3
## 8     8     3
## 9     9     3
```

5.2 for-while-repeat-next-break

- **for** döngüleri bir tekrarlayıcı değişken alır ve ona bir diziden veya vektörden ardışık değerler atar. En yaygın olarak bir nesnenin öğeleri üzerinde tekrarlayan işlem yapmak için kullanılır.

- **while** döngüleri bir şartı test ederek başlar. Eğer denenecek şart doğru ise istenilen komutlar yerine getirilir. Döngü şartın doğru olmadığı ana kadar devam eder.
- **repeat** sonsuz bir döngü oluşturur. Döngüden çıkmak için **break** kullanılır.
- **next** ifadesi ile bir döngüdeki belirli tekrarlar atlanabilir.

```
for (i in 1:5) {
    print(i)
}
```

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

```
v <- LETTERS[1:4]
for ( i in v) {
   print(i)
}
```

```
## [1] "A"
## [1] "B"
## [1] "C"
## [1] "D"
```

```
# dataframe içerisinde for
for (i in 1:nrow(df)){

  df[i,"multiply"] <- df[i,"value"] * df[i,"group"]
}

# i yerine farklı ifade de kullanılabilir
(x <- data.frame(age=c(28, 35, 13, 13),
                 height=c(1.62, 1.53, 1.83, 1.71),
                 weight=c(65, 59, 72, 83)))
```

```
##   age height weight
## 1  28   1.62     65
## 2  35   1.53     59
## 3  13   1.83     72
## 4  13   1.71     83
```

```r
for (var in colnames(x)) {
    m <- mean(x[, var])
    print(paste("Average", var, "is", m))
}
```

```
## [1] "Average age is 22.25"
## [1] "Average height is 1.6725"
## [1] "Average weight is 69.75"
```

```r
# while

x <- 0

while (x^2 < 20) {
  print(x)      # Print x
  x <- x + 1    # x'i bir artır
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

```r
# repeat

x <- 0

repeat {
    if (x^2 > 20) break     # bu koşul sağlandığında döngüyü bitir
    print(x)
    x <- x + 1              # x'i bir artır
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

```r
# next

for(i in 1:7) {
if (i==4) next # i=4 olduğunda atla
print(1:i)
}
```

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

```r
(s <- seq(1,10,1))
```

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

```r
for (i in s) {
    if (i%%2 == 1) { # mod
        next
    } else {
        print(i)
    }
}
```

```
## [1] 2
## [1] 4
## [1] 6
## [1] 8
## [1] 10
```

```r
# döngü içinde döngü

(mat <- matrix(nrow=4, ncol=4))
```

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

```
nr <-  nrow(mat)
nc <-  ncol(mat)

# matrisin içini dolduralım
for(i in 1:nr) {
  for (j in 1:nc) {
    mat[i, j] = i * j
  }
}

mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    2    4    6    8
## [3,]    3    6    9   12
## [4,]    4    8   12   16
```

6 Tarih ve Zaman İşlemleri

Tarihler, Date sınıfı tarafından temsil edilir ve **as.Date()** işlevi kullanılarak bir karakter dizesinden oluşturulabilir. Bu, R'de bir Date nesnesi elde etmenin yaygın bir yoludur. Date sınıfı varsayılan olarak tarihleri 1 Ocak 1970'den bu yana geçen günlerin sayısı olarak temsil eder. **as.Date()** işlevinin kullanılması bir karakter dizesinden Date nesneleri oluşturmamıza olanak tanır. Varsayılan biçim "YYYY/m/d" veya "YYYY-m-d" şeklindedir.

```
Sys.Date()
```

```
## [1] "2021-12-24"
```

```
class(Sys.Date())
```

```
## [1] "Date"
```

```
myDate <- as.Date("2022-01-04")

class(myDate)
```

```
## [1] "Date"
```

```r
# format argümanı ile tarih formatı tanımlanabilir
as.Date("12/31/2021", format = "%m/%d/%Y")
```

```
## [1] "2021-12-31"
```

```r
# year
format(myDate, "%Y")
```

```
## [1] "2022"
```

```r
as.numeric(format(myDate, "%Y"))
```

```
## [1] 2022
```

```r
# weekday
weekdays(myDate)
```

```
## [1] "Salı"
```

```r
# month
months(myDate)
```

```
## [1] "Ocak"
```

```r
# quarters
quarters(myDate)
```

```
## [1] "Q1"
```

```r
# create date sequence
date_week <- seq(from = as.Date("2021-10-1"),
    to = as.Date("2021/12/31"),
    by = "1 week")

date_week
```

```
##  [1] "2021-10-01" "2021-10-08" "2021-10-15" "2021-10-22" "2021-10-29"
##  [6] "2021-11-05" "2021-11-12" "2021-11-19" "2021-11-26" "2021-12-03"
## [11] "2021-12-10" "2021-12-17" "2021-12-24" "2021-12-31"
```

```
date_day <- seq(from = as.Date("2021-12-15"),
    to = as.Date("2021/12/31"),
    by = "day")

date_day
```

```
##  [1] "2021-12-15" "2021-12-16" "2021-12-17" "2021-12-18" "2021-12-19"
##  [6] "2021-12-20" "2021-12-21" "2021-12-22" "2021-12-23" "2021-12-24"
## [11] "2021-12-25" "2021-12-26" "2021-12-27" "2021-12-28" "2021-12-29"
## [16] "2021-12-30" "2021-12-31"
```

```
date_month <- seq(from = as.Date("2021-1-15"),
    to = as.Date("2021/12/31"),
    by = "month")

date_month
```

```
##  [1] "2021-01-15" "2021-02-15" "2021-03-15" "2021-04-15" "2021-05-15"
##  [6] "2021-06-15" "2021-07-15" "2021-08-15" "2021-09-15" "2021-10-15"
## [11] "2021-11-15" "2021-12-15"
```

Temel R **POSIXt** sınıfları, saat dilimlerini kontrol ederek tarih ve saatlere izin verir. R'de kullanılabilen iki POSIXt alt sınıfı vardır: **POSIXct ve POSIXlt.** POSIXct sınıfı, GMT (UTC – evrensel saat, koordineli) 1970-01-01 gece yarısından bu yana işaretli saniye sayısı olarak tarih-saat değerlerini temsil eder. POSIXlt sınıfı, tarih-saat değerlerini, saniye (sn), dakika (dk), saat (saat), ayın günü (mday), ay (mon), yıl (yıl), gün için öğeleri içeren adlandırılmış bir liste olarak temsil eder.

tarih-saatleri temsil eden en yaygın format kodları seti, `strptime()` işlevinin yardım dosyasında listelenmiştir (konsolunuza help(`strptime`) yazın).

```
Sys.time()
```

```
## [1] "2021-12-24 22:14:19 +03"
```

```
class(Sys.time())
```

```
## [1] "POSIXct" "POSIXt"
```

```
myDateTime <- "2021-12-11 22:10:35"
myDateTime
```

```
## [1] "2021-12-11 22:10:35"
```

```
class(myDateTime)
```

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

```
as.POSIXct(myDateTime)
```

```
## [1] "2021-12-11 22:10:35 +03"
```

```
class(as.POSIXct(myDateTime))
```

```
## [1] "POSIXct" "POSIXt"
```

```
Sys.timezone()
```

```
## [1] "Europe/Istanbul"
```

```
as.POSIXct("30-12-2021 23:25", format = "%d-%m-%Y %H:%M")
```

```
## [1] "2021-12-30 23:25:00 +03"
```

```
myDateTime.POSIXlt <- as.POSIXlt(myDateTime)

# seconds
myDateTime.POSIXlt$sec
```

```
## [1] 35
```

```
# minutes
myDateTime.POSIXlt$min
```

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

```
# hours
myDateTime.POSIXlt$hour
```

```
## [1] 22
```

```
# POSIXt nesneleri tarih formatına dönüştürülebilir.
as.Date(myDateTime.POSIXlt)
```

```
## [1] "2021-12-11"
```

Lubridate paketi, R'de tarih ve saatlerle çalışmayı kolaylaştıran çeşitli işlevler sağlar. Lubridate paketi, `ymd()`, `ymd_hms()`,`dmy()`, `dmy_hms()`,`mdy()`gibi işlevler sağlayarak tarih-zamanların ayrıştırılmasını kolay ve hızlı hale getirir.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# convert a number into a data object
ymd(20211215) # year-month-date
```

```
## [1] "2021-12-15"
```

```
ymd_hm(202112121533) # year-month-date-hour-minute
```

```
## [1] "2021-12-12 15:33:00 UTC"
```

```
mdy("Aralık 13, 2021") # month date year
```

```
## [1] "2021-12-13"
```

```
mdy("12 18, 2021") # month date year
```

```
## [1] "2021-12-18"
```

```
dmy(241221) # day-month-year
```

```
## [1] "2021-12-24"
```

```r
dmy(24122021) # day-month-year
```

```
## [1] "2021-12-24"
```

```r
today <- Sys.time()
today
```

```
## [1] "2021-12-24 22:14:20 +03"
```

```r
year(today) # year
```

```
## [1] 2021
```

```r
month(today) # month
```

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

```r
month(today, label = TRUE) # labeled month
```

```
## [1] Ara
## 12 Levels: Oca < Şub < Mar < Nis < May < Haz < Tem < Ağu < Eyl < ... < Ara
```

```r
month(today,label = TRUE, abbr = FALSE) # labeled month
```

```
## [1] Aralık
## 12 Levels: Ocak < Şubat < Mart < Nisan < Mayıs < Haziran < ... < Aralık
```

```r
week(today) # week
```

```
## [1] 52
```

```r
mday(today) # day
```

```
## [1] 24
```

```
wday(today) # weekday
```

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

```
wday(today, label = TRUE) # labeled weekday
```

```
## [1] Cum
## Levels: Paz < Pzt < Sal < Çar < Per < Cum < Cmt
```

```
wday(today, label = TRUE, abbr = FALSE) # labeled weekday
```

```
## [1] Cuma
## 7 Levels: Pazar < Pazartesi < Salı < Çarşamba < Perşembe < ... < Cumartesi
```

```
yday(today) # day of the year
```

```
## [1] 358
```

```
hour(today) # hour
```

```
## [1] 22
```

```
minute(today) # minute
```

```
## [1] 14
```

```
second(today) # second
```

```
## [1] 20.04883
```

Yukarıda listelenen çeşitli işlevlere ek olarak, **zoo** paketindeki **as.yearmon()** ve **as.yearqtr()** işlevleri, düzenli aralıklarla aylık ve üç aylık verilerle çalışırken uygundur.

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
as.yearmon(today)
```

```
## [1] "Ara 2021"
```

```
format(as.yearmon(today), "%B %Y")
```

```
## [1] "Aralık 2021"
```

```
format(as.yearmon(today), "%Y-%m")
```

```
## [1] "2021-12"
```

```
as.yearqtr(today)
```

```
## [1] "2021 Q4"
```

```
# dataframe içerisinde tarih kullanmak
df <-
  data.frame(date = c(
    "2010-02-01",
    "20110522",
    "2009/04/30",
    "2012 11 05",
    "11-9-2015"
  ))

df$date2 <- as.Date(parse_date_time(df$date, c("ymd", "mdy")))
df
```

```
##         date      date2
## 1 2010-02-01 2010-02-01
## 2   20110522 2011-05-22
## 3 2009/04/30 2009-04-30
## 4 2012 11 05 2012-11-05
## 5  11-9-2015 2015-11-09
```

7 Metin İşlemleri

R'de bir çift tek tırnak veya çift tırnak içine yazılan herhangi bir değer, bir karakter olarak kabul edilir. Karakter yapısına sahip olan verilerin analizi özellikle metin madenciliği konusunda kullanışlıdır. Karakter nesneleri üzerinde çalışmak için kullanılabilecek birçok fonksiyon vardır.

```
# as.character
as.character(3.14)
```

```
## [1] "3.14"
```

```
class(as.character(3.14))
```

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

```
# paste and paste0 karakter verilerini birleştirir

first <- "Fatih"
last <- "Tüzen"
paste(first,last) # default olarak arada boşluk bırakır
```

```
## [1] "Fatih Tüzen"
```

```
paste0(first,last) # default olarak arada boşluk yoktur
```

```
## [1] "FatihTüzen"
```

```
paste("R","Python","SPSS",sep = "-")
```

```
## [1] "R-Python-SPSS"
```

```
# grep fonksiyonu metin vektörünün içinde belirli bir deseni arar

x <- c("R programı","program","istatistik","programlama dili","bilgisayar","matematik")
grep("program",x)
```

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

```r
grep("^ist",x) # ist ile başlayan ifdelerin olduğu yerler
```

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

```r
grep("tik$",x) # tik ile biten ifdelerin olduğu yerler
```

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

```r
# grepl TRUE-FALSE olarak sonuç döndürür
grepl("tik$",x) # tik ile biten ifdelerin olduğu yerler
```

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

```r
x[grep("tik$",x)] # tik ile biten ifdelerin olduğu yerler
```

```
## [1] "istatistik" "matematik"
```

```r
x[grepl("tik$",x)] # tik ile biten ifdelerin olduğu yerler
```

```
## [1] "istatistik" "matematik"
```

```r
# nchar karakter uzunluğunu verir
nchar(x)
```

```
## [1] 10  7 10 16 10  9
```

```r
nchar("R Programlama") # boşluklar da sayılır!
```

```
## [1] 13
```

```r
# tolower ve toupper
toupper("program") # karakteri büyük harf yapar
```

```
## [1] "PROGRAM"
```

```r
tolower(c("SPSS","R","PYTHON")) # karakteri küçük harf yapar
```

```
## [1] "spss"   "r"      "python"
```

```r
# substr ve substring ile karakter parçalama yapılır
substr("123456789",start = 3, stop = 6)
```

```
## [1] "3456"
```

```r
substring("123456789", first =3, last = 6)
```

```
## [1] "3456"
```

```r
x <- "R Programlama"
substr(x,nchar(x)-3,nchar(x)) # son 4 karakteri getir
```

```
## [1] "lama"
```

```r
# strsplit karakteri bölme işini yapar
strsplit("Ankara;İstanbul;İzmir",split = ";")
```

```
## [[1]]
## [1] "Ankara"   "İstanbul" "İzmir"
```

8 Apply Ailesi

Apply() ailesi, matrislerden, dizilerden, listelerden ve veri çerçevelerinden tekrarlayan bir şekilde veri dilimlerini işlemek için fonksiyonlarla doldurulur. Bu fonksiyonlar sayesinde döngü yapılarının kullanılmasından kaçınır. Bir girdi listesi, matris veya dizi üzerinde hareket ederler ve bir veya birkaç isteğe bağlı argümanla adlandırılmış bir fonksiyon uygularlar.

- `apply()`: bir dizinin ya da matrisin satır ya da sütunlarına fonksiyon uygular.
- `lapply()`: liste üzerindeki her elemana fonksiyon uygular.
- `sapply()`: lapply fonksiyonu ile aynıdır ancak çıktısı matris ya da veri çerçevesidir.
- `mapply()`: lapply fonksiyonunun çoklu versiyonudur.
- `tapply()`: faktör ya da grup düzeyinde fonkyion uygular.

```r
# apply
x <-matrix(rnorm(30), nrow=5, ncol=6)
x
```

```
##            [,1]       [,2]       [,3]        [,4]       [,5]       [,6]
## [1,]  1.3639220  1.4634076  0.7255986 -0.47217008 -0.7614269  0.5745030
## [2,] -0.5036334  0.1243783 -0.4592384 -1.63237193 -1.3288423 -1.5952915
## [3,] -1.7090602  1.4537408  1.6847592 -2.17835531 -0.6020307 -0.6240689
## [4,]  0.8985497  0.3502270  0.1465840  0.05920865 -1.5505253  1.0472161
## [5,] -0.2377340 -0.3490255 -2.0298571  0.64786064  0.7030018 -0.1680592
```

```r
apply(x, 2 ,sum) # sütunlar üzerinde işlem yapar
```

```
## [1] -0.18795597  3.04272817  0.06784633 -3.57582803 -3.53982344 -0.76570055
```

```r
apply(x, 1 ,sum) # satırlar üzerinde işlem yapar
```

```
## [1]  2.8938342 -5.3949993 -1.9750150  0.9512601 -1.4338134
```

```r
apply(x, 2 ,sd)
```

```
## [1] 1.2136371 0.8159707 1.3889468 1.1724690 0.8803840 1.0334550
```

```r
apply(x, 1 ,mean)
```

```
## [1]  0.4823057 -0.8991666 -0.3291692  0.1585433 -0.2389689
```

```r
mat <- matrix(c(1:12),nrow=4)
mat
```

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

```r
apply(mat,2,function(x) x^2) # gözlemlerin karesi alınır
```

```
##      [,1] [,2] [,3]
## [1,]    1   25   81
## [2,]    4   36  100
## [3,]    9   49  121
## [4,]   16   64  144
```

```r
apply(mat,2, quantile,probs=c(0.25,0.5,0.75)) # extra argüman eklenebilir
```

```
##     [,1] [,2]  [,3]
## 25% 1.75 5.75  9.75
## 50% 2.50 6.50 10.50
## 75% 3.25 7.25 11.25
```

```r
# lapply

a <-matrix(1:9, 3,3)
b <-matrix(4:15, 4,3)
c <-matrix(8:10, 3,2)
mylist<-list(a,b,c)
mylist
```

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

```r
lapply(mylist,mean)
```

```
## [[1]]
## [1] 5
##
## [[2]]
## [1] 9.5
##
## [[3]]
## [1] 9
```

```r
lapply(mylist,sum)
```

```
## [[1]]
## [1] 45
##
## [[2]]
## [1] 114
##
## [[3]]
## [1] 54
```

```r
lapply(mylist, function(x) x[,1]) # listedeki her matrisin ilk kolonunu çıkar
```

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

```r
mylist2 <- list(a = 1:4, b = rnorm(10), c = rnorm(20, 1), d = rnorm(100, 5))
mylist2
```

```
## $a
## [1] 1 2 3 4
##
## $b
##  [1]  0.009515892  0.417240224  0.626834197  1.206243139  0.772565369
##  [6] -1.377567064 -0.362426925  0.302298496 -0.109079876 -2.179165281
##
## $c
##  [1]  0.2418853  2.0145512  1.1580472 -0.4725604  1.2159262  0.8412925
##  [7]  1.6718539  3.1065586 -0.5159001  0.4949365  0.8612371 -1.1362050
## [13]  0.9687800  0.4068310  3.2356028 -1.9179762  2.4882212  2.0080247
## [19]  1.7350916  1.1468120
##
## $d
##   [1] 4.289200 6.105631 4.114253 5.694762 5.402639 6.076238 4.403454 4.419012
##   [9] 5.302077 5.305685 6.373998 5.485399 5.144840 5.842620 4.456392 6.092972
##  [17] 3.977458 5.338147 3.293154 5.246449 3.432037 4.768515 3.242545 4.360380
```

```
##  [25] 4.223833 5.554775 4.417878 4.231405 6.221516 6.669170 6.093480 3.508757
##  [33] 6.276653 3.771462 4.928049 5.734458 4.786113 4.849607 5.125382 5.427858
##  [41] 5.411351 3.273639 4.824352 5.286839 5.590740 4.682640 4.297348 3.632412
##  [49] 4.271193 4.878475 4.364794 5.594703 5.367501 3.391262 5.317332 5.541705
##  [57] 4.738673 4.973504 4.567762 4.706613 3.992960 5.425208 3.775183 4.409535
##  [65] 4.086362 4.129082 4.631444 4.514747 4.300038 4.746566 3.415102 5.111038
##  [73] 5.631846 3.716901 3.586899 7.231997 5.239507 5.053193 4.801493 5.492648
##  [81] 5.037241 4.454088 4.567524 4.655515 4.680331 4.248301 6.074102 5.093822
##  [89] 4.226933 5.591249 5.350228 4.862420 5.400882 4.093621 4.801160 4.000845
##  [97] 3.816190 4.668601 5.168025 4.086994
```

```
lapply(mylist2, mean)
```

```
## $a
## [1] 2.5
##
## $b
## [1] -0.06935418
##
## $c
## [1] 0.9776505
##
## $d
## [1] 4.828709
```

```
# sapply

head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
lapply(cars,sum)
```

```
## $speed
## [1] 770
##
## $dist
## [1] 2149
```

```r
sapply(cars,sum)
```

```
## speed  dist
##   770  2149
```

```r
sapply(cars,median)
```

```
## speed  dist
##    15    36
```

```r
sapply(cars,mean)
```

```
## speed  dist
## 15.40 42.98
```

```r
# mapply

l1 <- list(a=c(1:5),b=c(6:10))
l2 <- list(c=c(11:15),d=c(16:20))

mapply(sum,l1$a,l1$b,l2$c,l2$d) # gözlemlerin toplamı
```

```
## [1] 34 38 42 46 50
```

```r
mapply(prod,l1$a,l1$b,l2$c,l2$d) # gözlemlerin çarpımı
```

```
## [1]  1056  2856  5616  9576 15000
```

```r
# tapply

df <- data.frame(x =round(runif(15,min=1,max=10)),
                 group=sample(c(1:3),15,replace = TRUE))
df
```

```
##     x group
## 1   5     2
## 2   9     1
## 3   1     3
## 4   1     1
## 5   4     1
```

```
## 6   4     2
## 7   9     2
## 8   6     2
## 9   2     2
## 10  2     2
## 11  2     3
## 12  9     3
## 13  5     3
## 14 10     2
## 15  7     1
```

```
tapply(df$x,df$group, FUN = mean)
```

```
##        1        2        3
## 5.250000 5.428571 4.250000
```

```
tapply(df$x,df$group, FUN = sum)
```

```
##  1  2  3
## 21 38 17
```

```
tapply(df$x,df$group, FUN = length)
```

```
## 1 2 3
## 4 7 4
```

```
tapply(df$x,df$group, FUN = range)
```

```
## $`1`
## [1] 1 9
##
## $`2`
## [1]  2 10
##
## $`3`
## [1] 1 9
```

9 Verilerin İçe ve Dışa Aktarılması

Temel anlamda R içerisinde excel ortamından (virgül ya da noktalı virgül ile ayrılmış) veri aktarımı (import) için `read.table, read.csv, read.csv2` fonksiyonları kullanılmaktadır. Excel'den veri aktarımı için `readxl` veya `openxlsx`paketi kullanılabilir. Verilerin dışa aktarılması için ise `write.csv, write.table` fonksiyonları kullanılabilir.

```
# delimiter/separator , ise
mtcars_csv <- read.csv("datasets/mtcars_csv.csv")
str(mtcars_csv)
```

```
## 'data.frame':    32 obs. of  12 variables:
##  $ car : chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : int  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : int  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : int  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : int  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: int  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: int  4 4 1 1 2 1 4 2 2 4 ...
```

```
# stringsAsFactors karakter kolonları faktöre çevirir
mtcars_csv <- read.csv("datasets/mtcars_csv.csv",
                       stringsAsFactors = TRUE)
str(mtcars_csv)
```

```
## 'data.frame':    32 obs. of  12 variables:
##  $ car : Factor w/ 32 levels "AMC Javelin",..: 18 19 5 13 14 31 7 21 20 22 ...
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : int  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : int  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : int  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : int  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: int  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: int  4 4 1 1 2 1 4 2 2 4 ...
```

```r
# delimiter/separator ; ise

mtcars_csv2 <- read.csv2("datasets/mtcars_csv2.csv")
str(mtcars_csv2)
```

```
## 'data.frame':    32 obs. of  12 variables:
##  $ car : chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ mpg : chr  "21" "21" "22.8" "21.4" ...
##  $ cyl : int  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: chr  "160" "160" "108" "258" ...
##  $ hp  : int  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: chr  "3.9" "3.9" "3.85" "3.08" ...
##  $ wt  : chr  "2.62" "2.875" "2.32" "3.215" ...
##  $ qsec: chr  "16.46" "17.02" "18.61" "19.44" ...
##  $ vs  : int  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : int  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: int  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: int  4 4 1 1 2 1 4 2 2 4 ...
```

```r
# read.table

mtcars_csv <- read.table("datasets/mtcars_csv.csv",
                         sep = ",",
                         header = TRUE)

mtcars_csv2 <- read.table("datasets/mtcars_csv2.csv",
                          sep = ";",
                          header = TRUE)

# txt uzantılı dosyalar

mtcars_txt <- read.table("datasets/mtcars_txt.txt",
                          sep = ";",
                          header = TRUE)

# excel dosyaları için
library(readxl)
mtcars_excel <- read_excel("datasets/mtcars_excel.xlsx",
                           sheet = "mtcars")
str(mtcars_excel)
```

```
## tibble [32 x 12] (S3: tbl_df/tbl/data.frame)
##  $ car : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
```

```
##  $ mpg : num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num [1:32] 6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num [1:32] 160 160 108 258 360 ...
##  $ hp  : num [1:32] 110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num [1:32] 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num [1:32] 2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num [1:32] 16.5 17 18.6 19.4 17 ...
##  $ vs  : num [1:32] 0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num [1:32] 1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num [1:32] 4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num [1:32] 4 4 1 1 2 1 4 2 2 4 ...
```

```
mtcars_excel2 <- read_excel("datasets/mtcars_excel.xlsx",
                            sheet = "mtcars2")
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
```

```
str(mtcars_excel2) # tablo 2. satırdan başlıyor o yüzden tablo başlıkları hatalı
```

```
## tibble [33 x 5] (S3: tbl_df/tbl/data.frame)
##  $ mtcars verisi: chr [1:33] "car" "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" ...
##  $ ...2         : chr [1:33] "mpg" "21" "21" "22.8" ...
##  $ ...3         : chr [1:33] "cyl" "6" "6" "4" ...
##  $ ...4         : chr [1:33] "disp" "160" "160" "108" ...
##  $ ...5         : chr [1:33] "hp" "110" "110" "93" ...
```

```
# istenilen satırı atlayarak istenilen sheet adı için,
mtcars_excel2 <- read_excel("datasets/mtcars_excel.xlsx",
                            sheet = "mtcars2",
                            skip = 1)
str(mtcars_excel2)
```

```
## tibble [32 x 5] (S3: tbl_df/tbl/data.frame)
##  $ car : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ mpg : num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num [1:32] 6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num [1:32] 160 160 108 258 360 ...
##  $ hp  : num [1:32] 110 110 93 110 175 105 245 62 95 123 ...
```

```
# export

write.csv(mtcars_csv,"write_mtcars.csv",
          row.names = FALSE)

write.table(mtcars_csv,"write_mtcars.csv",
            row.names = FALSE,
            sep = ";")

openxlsx::write.xlsx(mtcars_csv,"write_mtcars.xlsx")
```

R içerisinde yer alan hazır veri setlerine erişim için `data()` fonksiyonu kulanılır. Bu fonksyion R ortamınızdaki aktif kutuphanelerin icindeki veri setlerini listeler. Tüm veri setlerine erişim için `data(package = .packages(all.available = TRUE))` kodu çalıştırılır.

Dosyaları Excel gibi yazılımlarla açmak istiyorsanız, txt, csv veya Excel dosya formatlarında veri yazmak en iyi çözümdür. Ancak bu çözüm, sütun veri türleri (sayısal, karakter veya faktör) gibi veri yapılarını korumaz. Bunun için veriler R data formatında yazılmalıdır. Bu amaçla R ortamındaki verilerinizi tekil olarak saklamak ya da içeri aktarmak için **`saveRDS()`** ve **`readRDS()`** fonksiyonları kullanılır. Bu fonksiyonlar ile **`rds`**(serialized *R* data) uzantılı R formatındaki dosyalar kullanılır.

```
# Tek bir dosyayı rds formatında saklamak
saveRDS(mtcars, "datasets/mtcars.rds")

# rds uzantılı dosyayı yüklemek
my_data <- readRDS("datasets/mtcars.rds")
```

Eğer birden fazla dosya aynı anda saklanmak ya da içeri aktarılmak isteniyorsa **`save`** ve **`load`** fonksiyonları kullanılır. Bu fonksiyonlar ile **`RData`** uzantılı dosyalar elde edilir. Ayrıca **`rda`**(**`Rdata`**) uzantılı dosyalar da bir veya daha fazla farklı R nesnesi alabilir.

```
# RData formatında tek dosya saklamak
save(my_data, file = "datasets/my_data.RData")

# RData formatında birden fazla dosya saklamak
save(my_data, mtcars_csv, file = "datasets/data.RData")

# RData formatındaki verileri yüklemek
load("datasets/data.RData")
```

RStudio'yu kapattığınızda, çalışma alanınızı kaydetmek isteyip istemediğinizi sorar. Evet derseniz, RStudio'yu bir sonraki başlatışınızda o çalışma alanı yüklenecektir. Bu

kaydedilen dosya da .RData olarak adlandırılacaktır. Çalışma alanınızı kaydetmek için dosya adını belirtmek de mümkündür. Bunun için **save.image()** fonksiyonu kullanılır.

```
# workspace saklamak
save.image(file = "datasets/my_work_space.RData")

# workspace yüklemek
load("datasets/my_work_space.RData")
```

Ayrıca aşağıdaki programlar ile üretilmiş veriler için **haven** ya da **foreign** paketleri kullanılabilir.

- **SAS** (sas7bdat, sas7bcat)
- **SPSS** (sav, por)
- **STATA** (dta)