# R Temelleri 2

## Vektörler ve Türler

R'da bir vektör ilişkili şeylerin (sıralı) bir kümesidir.

Bunu Excel'deki bir sütun olarak düşünebilirsiniz.

```
x <- c(4, 7, 9)
x
```

```
## [1] 4 7 9
```

```
y <- c('a', 'b', 'c')
y
```

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

`4`, `7` ve `9` ilişkilidir çünkü tümü sayıdırlar.

Benzer biçimde `a`, `b` ve `c` de harftirler.

İlişkili olmayan şeyleri birleştirmeye çalıştığımızda ne olur?

```
x <- c(1, TRUE, "üç")
x
```

```
## [1] "1"    "TRUE" "üç"
```

Tırnak işaretlerine dikkat edin!

R `1` ve `TRUE`'yu metin gösterimine dönüştürdü.

Bunun nedeni `1` ve `TRUE`'nun `"üç"`'den farklı bir türe sahip olmasıdır.

---

R'da karşımıza çıkabilecek 4 temel değişken türü vardır. Bunlar hiyerarşik sırayla şu şekidedir.

1. `logical` (Mantıksal): `TRUE` (Doğru) veye `FALSE` (Yanlış)
2. `integer` (Tam Sayı): `0L`, `-1L`, `1L`, vb. Ondalık kısmı olmayan (reel) sayılar.
3. `numeric` (Sayısal): `pi`, `0.34`, `1.4043`, vb. Reel sayılar.
4. `character` (Karakter): `"bağzı şeyler"`, `"bağzı bağzı şeyler"`, vb.

---

Vektörler bu listedeki en yüksek sayıya sahip türe dönüştürülür.

Yukarıdaki örnekte `x`'in içeriğinde `"üç"` vardı.

Dolayısıyla tüm vektör karakter vektörüne dönüştürüldü.

## Vektör Aritmetiği ve Fonksiyonlar

Vektörler bir çok hesaplamayı bir kerede yapmaya olanak sağlar.

Bir sayı listesini toplamak, tüm listeyi 3'e bölmek, vb.

```
x <- c(1, 2, 3)
x + 4
```

```
## [1] 5 6 7
```

```
x/3
```

```
## [1] 0.3333 0.6667 1.0000
```

```
-x
```

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

```
x^3
```

```
## [1]  1  8 27
```

```
y <- c(3, 2, 1)
x - y
```

```
## [1] -2  0  2
```

```
x * y
```

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

```
x/y
```

```
## [1] 0.3333 1.0000 3.0000
```

```
x > 2
```

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

```
x >= 2
```

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

Matematikteki gibi fonksiyon bir girdiyi bir çıktıya eşleştirir.

Matematik derslerinde olduğu gibi fonksiyonlar `()` kullanırlar.

Örneğin daha önce vektör yaratan **c**oncatenate (birleştirme) fonksiyonunu `c` gördük.

Vektör girdisine istediğimiz kadar fonksiyon uygulayabiliriz.

Örneğin:

```
x <- c(1, 2, 3)
y <- c(-1, 2, 4)

# sum: vektör elemanlarını topla
sum(x)
```

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

```
# sum komutuyla sayısal vektör elemanlarını
# topladığımız gibi prod komutuyla elemanları
# çarpabiliriz:
prod(x)
```

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

```
# sqrt: karekök
sqrt(x)
```

```
## [1] 1.000 1.414 1.732
```

```
# abs: mutlak değer
abs(y)
```

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

```
# exp: üstel exp(x) is e^x
exp(y)
```

```
## [1]  0.3679  7.3891 54.5982
```

```
# log: doğal logaritma (e tabanı)
log(x)
```

```
## [1] 0.0000 0.6931 1.0986
```

```
# sin ve cos fonksiyonları derece
# değil radyan ile çalışır
sin(x) + cos(y)
```

```
## [1]  1.3818  0.4932 -0.5125
```

```
# max: maksimum
max(y)
```

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

```
# min: minimum
min(y)
```

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

```
# range: aralık
range(y)
```

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

```
# mean: ortalama
mean(x)
```

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

```
# median: medyan
median(x)
```

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

R'da sürekli yapacağımız bir diğer şey de, düzenli aralıklarla sayı dizileri kullanmaktır.

Bunlar R'da `:` veya `seq` ile oluşturulur.:

```
x <- 1:10
x
```

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

```
y <- 10:1
y
```

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

```
# Bazen aralıkların 1'den farklı olması istenir
z <- seq(0, 1, by = .25)
z
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
# Bazen de aralıklar değil eleman sayısı önemlidir
w <- seq(0, 1, length.out = 10)
w
```

```
##  [1] 0.0000 0.1111 0.2222 0.3333 0.4444 0.5556 0.6667 0.7778 0.8889 1.0000
```

Matematik / aritmetik fonksiyonlarına ek olarak, her zaman kullanmanız muhtemel olan temel programlama fonksiyonlarının bazıları şunlardır:

```
x <- 99:32

# length: x'te kaç eleman vardır?
length(x)
```

```
## [1] 68
```

```
y <- c("hey you!", "out there in the cold")

# y'nin türü nedir?
class(y)
```

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

```
# rep: tekrarla
rep(y, 4)
```

```
## [1] "hey you!"              "out there in the cold" "hey you!"
## [4] "out there in the cold" "hey you!"              "out there in the cold"
## [7] "hey you!"              "out there in the cold"
```

```
# head/tail: bir objenin sadece
# başlangıç ve sonunu göster
# büyük objeler için yararlıdır
x <- 1:100000
head(x)
```

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

```
tail(x)
```

```
## [1]  99995  99996  99997  99998  99999 100000
```

## Vektörleri Altkümeleme: `[`

Genellikle bir vektörün sadece bir kısmını, genellikle bir koşulu karşılayan bir kısmını incelemek isteriz.

Bazen de ilk veya son birkaç elemanını incelemek isteriz.

Bunu yapmak için `[` işlevini kullanarak bu elemanları çekeriz:

```
x <- c(5, 4, 1)
x[1]
```

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

```
x[3]
```

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

```
x[1:2]
```

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

```
x[2:3]
```

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

`x[birsey]` notasyonunda, ***`birsey`*** *bir vektördür*!

Bu yüzden yukarıdaki örneklerin hepsi daha karmaşık alt grup türleri için kısa bir yoldur:

```
x <- 20:30
x
```

```
##  [1] 20 21 22 23 24 25 26 27 28 29 30
```

```
x[c(1, 3, 5)]
```

```
## [1] 20 22 24
```

```
x[c(5, 9)]
```

```
## [1] 24 28
```

```
x[seq(1, 10, by = 2)]
```

```
## [1] 20 22 24 26 28
```

`tamsayı` olmanın yanı sıra, `birsey` vektör uzunluğuna eşit *mantıksal vektör* de olabilir:

```
x <- c(5, 6, 7)
x[c(TRUE, TRUE, FALSE)]
```

```
## [1] 5 6
```

```
x[c(FALSE, TRUE, FALSE)]
```

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

```
x[c(FALSE, FALSE, TRUE)]
```

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

Yaygın olarak, yukarıdakilerle aynı olan ancak daha doğal olan bir yöntem izleriz:

```
x <- c(-1, 0, 1)
x
```

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

```
x > 0
```

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

```
x[x > 0]
```

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

```
x[x <= 0]
```

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

Ayrıca bir vektörün parçalarını *değiştirebiliriz*:

```r
x <- c(-1, 5, 10)
x[3] <- 4
x
```

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

```r
x[x < 0] <- 0
```

## İsimlendirilmiş Vektörler

Bilgiyi düzenlemeye yardımcı olmak için vektörlerimizi *isimlendirmek* genellikle yararlıdır.

---

Trump ailesinin üyelerinin yaşlarını takip ettiğimizi varsayalım:

```r
trump_ages <- c(70, 46, 38, 34, 32, 22, 9)
```

Her elemanın kimi temsil ettiğini takip edersek çok daha faydalı olur:

```r
trump_ages <- c(Donald = 70, Melania = 46, Donald_Jr = 38, Ivanka = 34,
                Eric = 32, Tiffany = 22, Barron = 9)
trump_ages
```

```
##    Donald   Melania Donald_Jr    Ivanka      Eric   Tiffany    Barron
##        70        46        38        34        32        22         9
```

İsim atamak için `names` işlevini de kullanabiliriz; örneğin isimler boşluk içeriyorsa bu daha kolaydır:

```r
names(trump_ages) <- c("Donald", "Melania", "Donald, Jr.", "Ivanka", "Eric", "Tiffany", "Barron")
trump_ages
```

```
##      Donald     Melania Donald, Jr.      Ivanka        Eric     Tiffany      Barron
##          70          46          38          34          32          22           9
```

Bu aynı zamanda alt küme için kodun okunmasını kolaylaştırır, çünkü isimleri kullanarak alt kümeye girebiliriz:

```r
trump_ages["Donald"]
```

```
## Donald
##     70
```

```r
trump_ages[c("Donald", "Barron")]
```

```
## Donald Barron
##     70      9
```

## Listeler

Yukarıda, R'ın vektörlerin farklı türlere sahip olmasını sevmediğini gördük: `c(TRUE, 1, "Frank")`, `c("TRUE", "1", "Frank")`olur.

Ancak farklı türlerdeki nesneleri depolamak veri analizinde kesinlikle esastır.

R, farklı türlerdeki verileri yan yana depolamak için vektörlerin yanı sıra diğer bir nesneye daha sahiptir: `liste`:

```r
x <- list(TRUE, 1, "Frank")
x
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] 1
##
## [[3]]
## [1] "Frank"
```

`c` ile karşılaştırıldığında çıktının ne kadar farklı göründüğüne dikkat edin!

Son bileşen dışında tırnak işaretleri silinmiştir.

Şimdilik `[[` ve `[` dağınıklığını görmezden gelebilirsiniz:

```
x <- list(c(1, 2), c("a", "b"), c(TRUE, FALSE), c(5L, 6L))
x
```

```
## [[1]]
## [1] 1 2
##
## [[2]]
## [1] "a" "b"
##
## [[3]]
## [1]  TRUE FALSE
##
## [[4]]
## [1] 5 6
```

```
y <- list(list(1, 2, 3), list(4:5), 6)
y
```

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

`x`, her biri 2 bileşenli bir vektör olan 4 bileşenden oluşan bir listedir.

Bu, R'ın farklı türlerdeki birçok değişkenle bir veri setini nasıl ele aldığına dair ilk ipucunu verir - özünde, R bir veri setini bir listede saklar!

`y` ise iç içe geçmiş bir listedir - bazı bileşenleri listeler içeren bir listedir.