R Programlama

Muhammed Fatih Tüzen

Table of contents

Önsöz

R programlama dili, veri bilimi dünyasında vazgeçilmez bir araç haline geldi. Bu kitap, veri manipülasyonundan görselleştirmeye, keşifçi veri analizinden temel istatistik konularına kadar geniş bir yelpazede R dilini kullanarak veri analizi becerilerinizi güçlendirmenize odaklanıyor.

Kitabımız, R programlama dilini temel seviyeden başlayarak adım adım öğrenmek isteyen herkes için tasarlandı. İlk bölümlerde R dilinin temellerini kavrayacak ve dplyr gibi güçlü paketler aracılığıyla veri manipülasyonunun inceliklerini keşfedeceksiniz. Veri analizinin görselleştirme aşamasında ggplot2 paketiyle nasıl etkileyici grafikler oluşturabileceğinizi adım adım öğrenecek ve veri setlerinizin hikayesini çarpıcı görsellerle anlatacaksınız.

Kitabımız, keşifçi veri analizi sürecinde size rehberlik ederken, veri işleme tekniklerini ve önemli istatistik kavramlarını pratik örneklerle ele alacak. Temel istatistik kolları üzerine odaklanarak, veri setlerinizdeki deseni anlamak ve çözümlemek için gerekli araçları edineceksiniz. Ayrıca doğrusal regresyon gibi önemli modelleme tekniklerini R dilinde nasıl uygulayabileceğinizi adım adım öğreneceksiniz.

Bu kitabın amacı, R programlama dilini veri analizi süreçlerinizde güvenle kullanmanıza yardımcı olmak ve veri odaklı kararlar almanızı desteklemektir. Bilgi birikiminizi genişletirken öğrendiklerinizi uygulamaya dökme şansına sahip olacaksınız. Umarım bu kitap, veri analizi yolculuğunuzda size rehberlik eder ve R dilini kullanarak veriyle olan etkileşiminizi daha da derinleştirir.

R Programlama Hakkında

R programlama, veri analizi, istatistiksel ve ekonometrik hesaplamalar, veri görselleştirme ve veri madenciliği gibi istatistiksel ve veri analitiği işlemleri için kullanılan bir programlama dilidir. İlk olarak 1990 yılında **R**oss Ihaka ve **R**obert Gentleman tarafından geliştirilmeye başlanmıştır ve o zamandan bu yana istatistiksel analiz alanında çok popüler bir araç haline gelmiştir. Yazılım ismini yazarların isimlerinin baş harflerinden almaktadır.

R Programı ile Neler Yapılabilir

R, açık kaynaklı bir programlama dili ve yazılım ortamıdır, bu da onu geniş bir kullanıcı topluluğu tarafından desteklenen ve geliştirilen bir platform yapar. R ile yapılabilecek başlıca işler şunlardır:

1. **Veri Analizi**: R, veri çerçeveleri ve veri setleri üzerinde işlem yapmak için bir dizi fonksiyon ve araç sunar. Veri temizleme, dönüştürme, özeti alma ve analiz etme işlemleri R ile kolayca gerçekleştirilebilir.
2. **Veri Görselleştirme**: R, ggplot2 gibi grafik paketleri ile verilerinizi görselleştirmenize olanak tanır. Çeşitli grafik türleri (çizgi grafikleri, sütun grafikleri, dağılım grafikleri vb.) oluşturabilirsiniz.
3. **İstatistiksel Analiz**: R, istatistiksel modelleri oluşturmak, hipotez testleri yapmak ve regresyon analizi gibi istatistiksel analizler gerçekleştirmek için zengin bir araç seti sunar. Ayrıca zaman serisi analizi ve kümeleme gibi konularda da kullanılır.
4. **Veri Madenciliği**: R, veri madenciliği uygulamaları için kullanılabilir. Makine öğrenimi algoritmaları uygulamak ve veri madenciliği projeleri geliştirmek için paketler içerir.
5. **Raporlama**: R Markdown kullanarak veri analizi ve sonuçlarını raporlama için kullanılır. Bu, anlamlı ve formatlı raporlar oluşturmanıza yardımcı olur.
6. **Paketler ve Genişletilebilirlik**: R, kullanıcıların işlevselliği genişletmek için paketler ekleyebileceği bir sistem sunar. CRAN (Comprehensive R Archive Network) gibi kaynaklar, binlerce paketi içeren bir depo sağlar.

> **Not**
>
> R programlama özellikle istatistik, veri bilimi ve akademik araştırmalar alanlarında çok kullanılır, ancak endüstriyel uygulamalarda da giderek daha fazla kullanılmaktadır. R'nin açık kaynaklı olması ve geniş bir kullanıcı topluluğuna sahip olması, bu dilin popülerliğini artırmıştır. R ile çalışmak için temel programlama bilgisine sahip olmak yararlı olacaktır, ancak öğrenmesi oldukça erişilebilir bir dildir ve çevrimiçi kaynaklar ve kurslar mevcuttur.

R Programlama ile ilgili Faydalı Kaynaklar

R programlamayı öğrenmek ve geliştirmek için bir dizi faydalı kaynak bulunmaktadır. R programlamaya başlamak veya ilerlemek için kullanabileceğiniz bazı kaynaklar:

1. **Resmi R Web Sitesi**: R'nin resmi web sitesi (**https://www.r-project.org/**) R programlamaya başlamak için temel kaynaktır. Burada R'nin indirilmesi, kurulumu ve temel belgelendirme bilgilerine erişebilirsiniz.
2. **RStudio**: R programlama için yaygın olarak kullanılan RStudio IDE'si (Entegre Geliştirme Ortamı), R kodlarını yazmak, çalıştırmak ve yönetmek için güçlü bir araçtır. RStudio'nun resmi web sitesi (**https://www.rstudio.com/**) RStudio'nun indirilmesi ve kullanımı hakkında bilgi sunar.
3. **R Dersleri ve Kurslar**: İnternette birçok ücretsiz R dersi ve kursu bulabilirsiniz. Coursera, edX, Udemy ve DataCamp gibi platformlar, R programlamayı öğrenmek için çeşitli kurslar sunmaktadır.
4. **R Belgeleri**: R'nin resmi belgeleme (**https://cran.r-project.org/manuals.html**) kaynakları, R dilinin temellerini ve paketlerini öğrenmek için çok faydalıdır. R'deki komutlar ve fonksiyonlar hakkında ayrıntılı bilgi içerirler.
5. **Kitaplar**: R programlamayı öğrenmek için yazılmış birçok kitap bulunmaktadır. Örnek olarak, "R Graphics Cookbook" (Hadley Wickham), "R for Data Science" (Hadley Wickham ve Garrett Grolemund), "Advanced R" (Hadley Wickham) gibi kitaplar önerilebilir.
6. **Stack Overflow**: Programlama sorunları ve hatalarıyla karşılaştığınızda, Stack Overflow gibi forumlarda R ile ilgili sorular sormak ve cevaplamak için topluluktan yardım alabilirsiniz.
7. **GitHub**: R ile ilgili açık kaynaklı projeleri incelemek ve kendi projelerinizi paylaşmak için GitHub gibi platformları kullanabilirsiniz. GitHub'da R kodlarını içeren birçok depo bulunmaktadır.

8. **Bloglar ve Videolar**: R ile ilgili bloglar ve YouTube kanalları, öğrenmek ve güncel kalmak için harika kaynaklardır. RStudio Blog (**https://blog.rstudio.com/**) ve YouTube'da R ile ilgili videoları bulabileceğiniz RStudio'nun resmi kanalı bunlara örnektir.

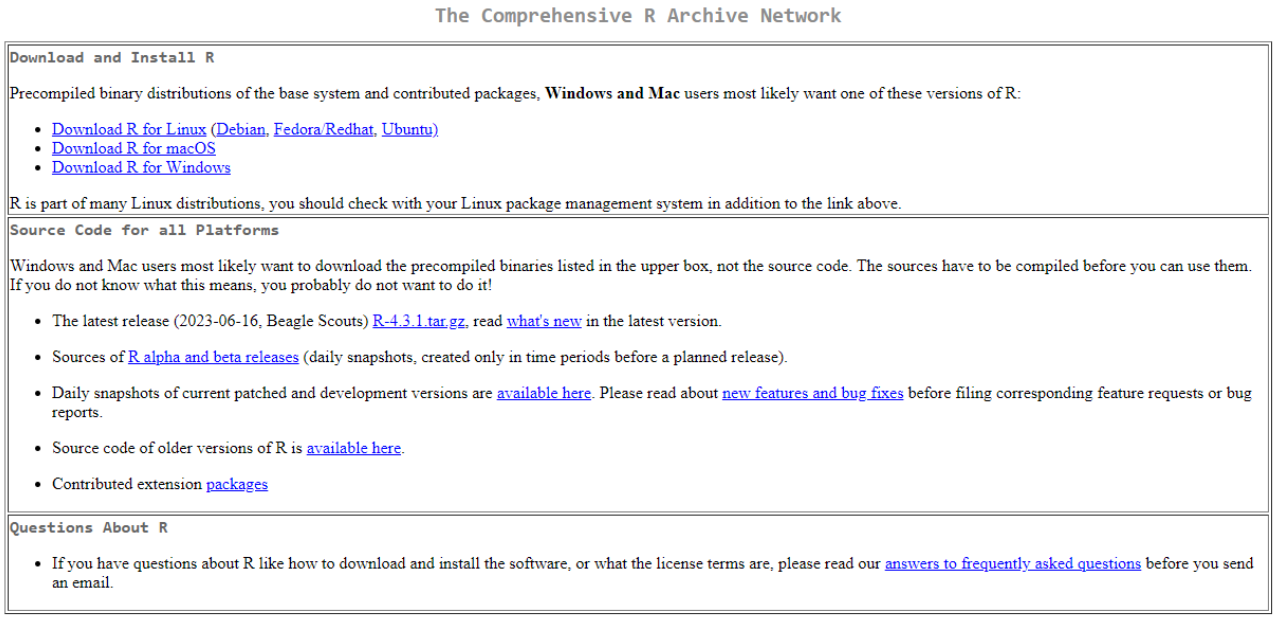
> Tavsiye
>
> R programlamayı öğrenmek ve geliştirmek için sürekli olarak yeni kaynaklar ve materyaller üretilmektedir. İhtiyacınıza ve seviyenize uygun kaynakları seçmek için zaman ayırın ve kendi hızınıza göre öğrenmeye devam edin.

R ve RStudio'nun Bilgisayara Kurulması

R'ın internet sitesinden işletim sisteminize uygun programı indirip kurabilirsiniz. Linux, Mac OS ve Windows işletim sistemleri için sürümleri mevcuttur.

Windows İşletim Sistemi İçin R Kurulumu

1. R programını indirmek için R resmi web sitesini ziyaret edin: **https://cran.r-project.org/**
2. Sayfanın üst kısmında **"Download R for Windows"** başlığını bulun ve tıklayın.


The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

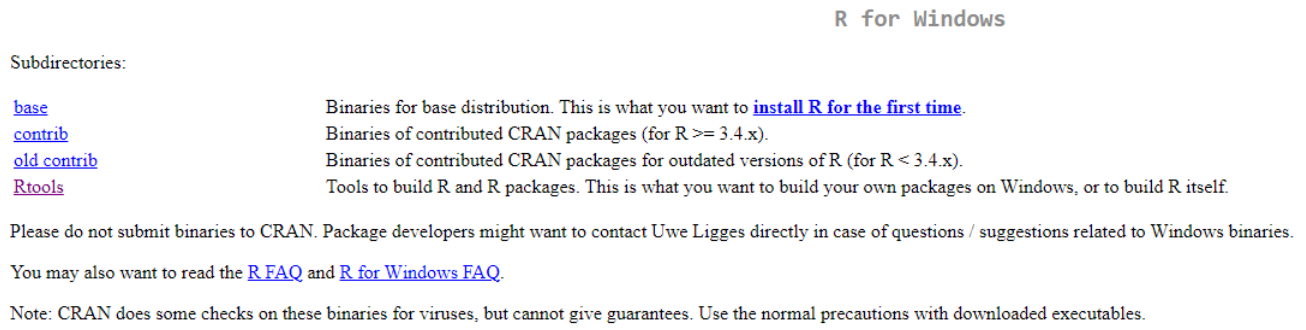
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2023-06-16, Beagle Scouts) R-4.3.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

3. İndirilen sayfada **"base"** sekmesine tıklayın.



R for Windows

Subdirectories:

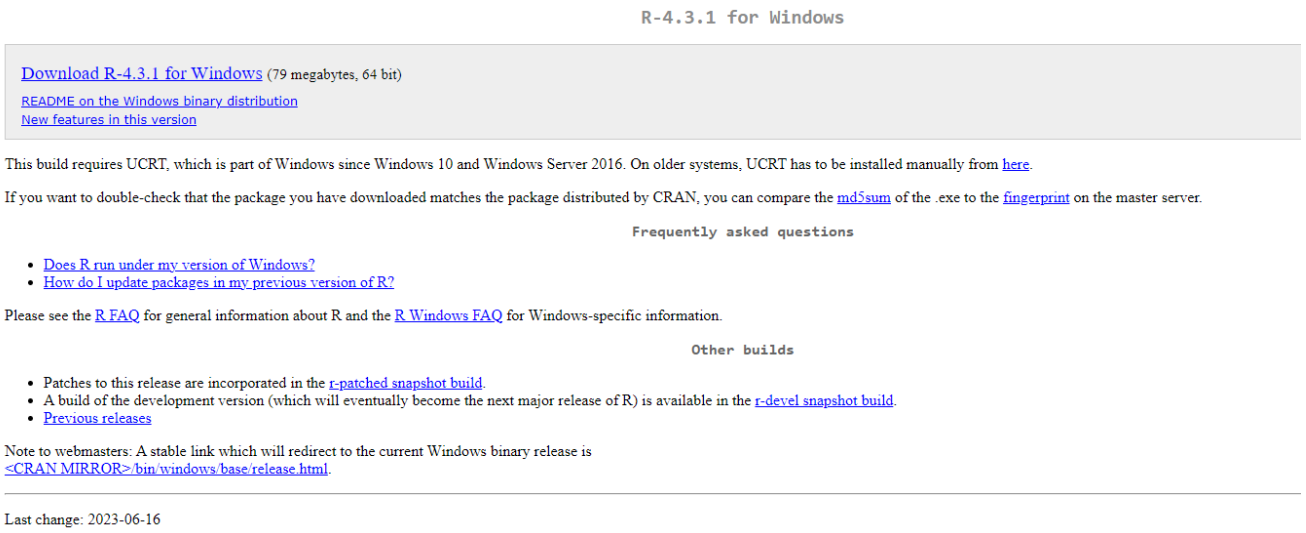
base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

4. Açılan sayfada "Download R 4.3.1 for Windows" linkine tıklayın ve dosyayı indirin.



R-4.3.1 for Windows

Download R-4.3.1 for Windows (79 megabytes, 64 bit)
README on the Windows binary distribution
New features in this version

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2023-06-16

> **Dikkat**
>
> Sayfayı ziyaret ettiğiniz tarihlerde farklı sürümlerin olabileceğine dikkat edin. Örneğin ileri bir tarihte bu sayfayı ziyaret ettiğinizde R programının yeni sürümü ile karşılabilirsiniz. O yüzden sürüm bilgisi değişkenlik gösterebilir.

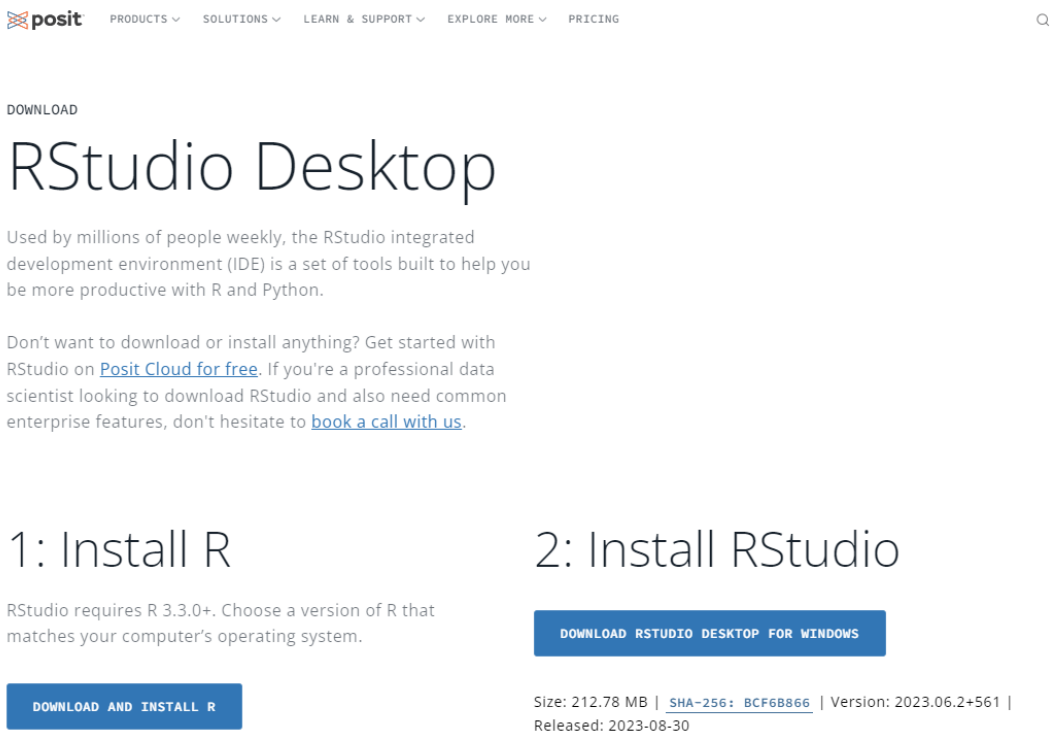
5. İndirilen dosyayı çift tıklayarak çalıştırın ve yükleyiciyi başlatın.
6. Yükleyici, R'nin temel sürümünü yüklemek için sizi yönlendirecektir. Varsayılan ayarları genellikle kabul edebilirsiniz.
7. Kurulum tamamlandığında, R'yi çalıştırmak için masaüstünüzde veya Başlat menüsünde **"R"** simgesini bulabilirsiniz.

Windows İşletim Sistemi İçin R Studio Kurulumu

R editörü grafiksel bir arayüz olmayıp eski tip bir yazılım konsoludur. **R Studio,** R programlama dili için geliştirilmiş entegre bir geliştirme ortamı (IDE) ve arayüzüdür. R Studio, R kodlarını daha verimli bir şekilde yazmanıza, çalıştırmanıza ve yönetmenize olanak tanıyan daha modern ve kullanışlı bir arayüz sunmaktadır. Ayrıca veri analizi, görselleştirme ve raporlama işlemleri için güçlü bir platform sunar. R Studio, açık kaynak bir projedir ve ücretsiz olarak kullanılabilir.

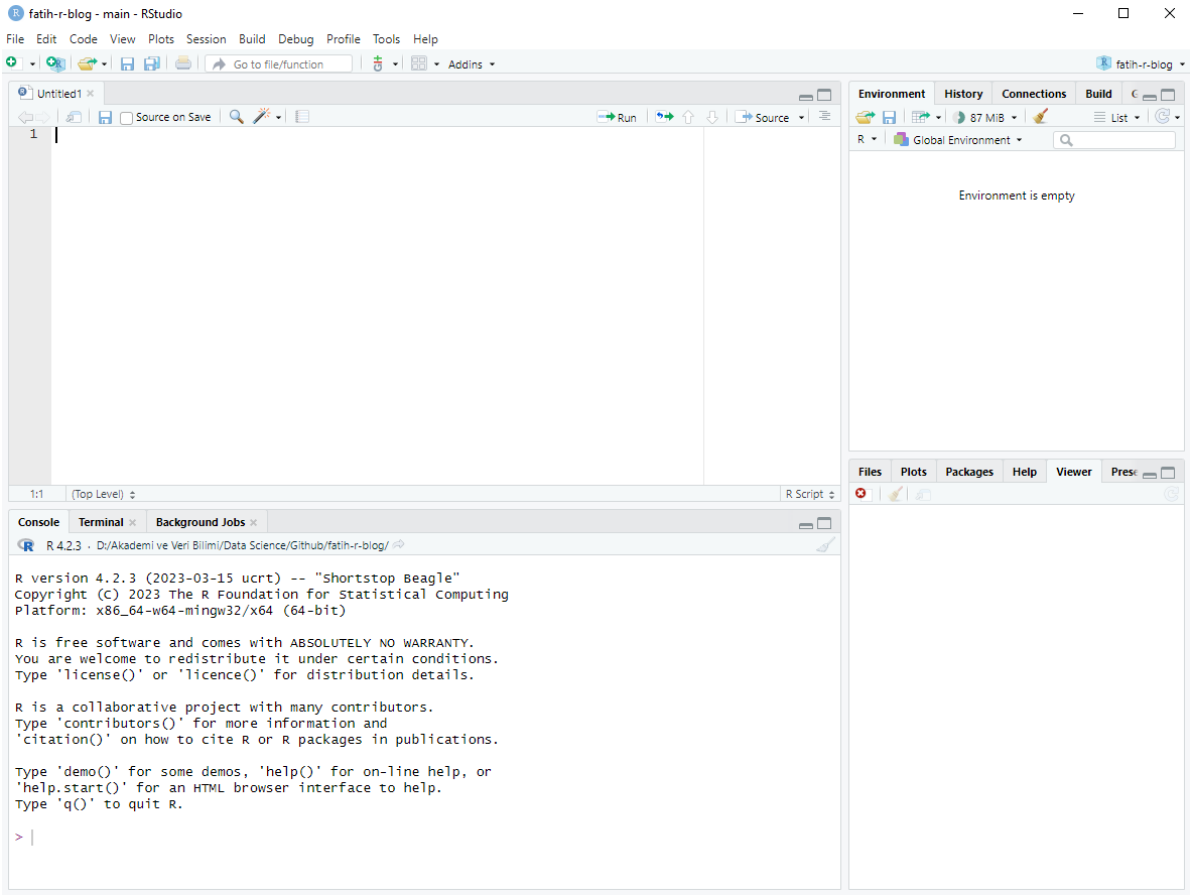
R Studio'nun kurulumu aşağıdaki adımlarla gerçekleştirilebilir:

1. R Studio'nun en son sürümünü indirmek için aşağıdaki bağlantıyı kullanın: **https://www.rstudio.com/products/rstudio/download/**
2. Sayfada **"Download RStudio Desktop for Windows"** kısmına tıklayın ve indirmeyi başlatın.



3. İndirilen dosyayı çift tıklayarak çalıştırın ve kurulumu başlatın. Kurulum sırasında varsayılan ayarları genellikle kabul edebilirsiniz.
4. Kurulum tamamlandığında, R Studio'yu başlatmak için masaüstünüzde veya Başlat menüsünde **"RStudio"** simgesini bulabilirsiniz.

R Studio Kişiselleştirme



RStudio, kullanıcıların ihtiyaçlarına göre kişiselleştirilebilen bir entegre geliştirme ortamı (IDE) sunar. RStudio'yu kişiselleştirmek için aşağıdaki yolları kullanabilirsiniz:

1. **R Studio Arayüzündeki Alanları Değiştirme**: Resimde görüldüğü gibi yeni bir R Script açıldığı takdirde arayüzde 4 farklı alan görülmektedir. Bu alanlar isteğe göre yer değiştirilebilmektedir. Bunun için **"Tools"** (Araçlar) menüsünden **"Global Options"** (Genel Ayarlar) sekmesi açılır. Buradan **"Pane Layout"** kısmından istenilen ayarlar yapılabilir.

2. **Temayı ve Editör Stilini Değiştirme**: RStudio'nun görünümünü değiştirmek için birçok tema ve editör stilini seçebilirsiniz. Bu, yazılım geliştirme ortamınızın daha hoş veya kullanışlı olmasını sağlar. **"Tools"** (Araçlar) menüsünden **"Global Options"** (Genel Ayarlar) sekmesini seçerek bu ayarları değiştirebilirsiniz.

3. **Klavye Kısayollarını Kişiselleştirme**: RStudio'da kullanılan klavye kısayollarını özelleştirebilirsiniz. "Tools" (Araçlar) menüsünden "Modify Keyboard Shortcuts"

(Klavye Kısayollarını Düzenle) seçeneğini kullanarak klavye kısayollarını tanımlayabilir veya değiştirebilirsiniz.

4. **Eklentileri ve Paketleri Kullanma**: RStudio, kullanıcıların işlevselliği genişletmek için eklentileri ve R paketlerini kullanmalarını sağlar. Bu paketler, kod otomatik tamamlama, kod görselleştirme, proje yönetimi gibi birçok işlemi kolaylaştırabilir. R Studio'nun sol üst köşesindeki **"Tools"** (Araçlar) menüsünden **"Install Packages"** (Paketleri Yükle) seçeneği ile yeni paketleri yükleyebilirsiniz.

5. **R Markdown Belgelerini Özelleştirme**: R Markdown belgeleri, raporlar ve belgeler oluşturmak için kullanılır. Bu belgeleri kişiselleştirebilirsiniz. R Markdown belgelerinin başlık, stil, tablo düzeni ve grafikler gibi birçok yönünü özelleştirebilirsiniz.

6. **Proje Ayarlarını Yapılandırma**: RStudio'da projeler kullanmak, projelerinizi daha düzenli ve etkili bir şekilde yönetmenize yardımcı olabilir. "File" (Dosya) menüsünden "New Project" (Yeni Proje) seçeneği ile yeni projeler oluşturabilir ve projelerinizi kişiselleştirebilirsiniz.

7. **Kod Tarayıcı ve Çalışma Ortamını Özelleştirme**: RStudio'nun sağ tarafında bulunan **"Environment"** (Çalışma Ortamı) ve **"Files"** (Dosyalar) sekmelerini özelleştirebilirsiniz. Bu sekmeleri dilediğiniz gibi düzenleyebilirsiniz.

8. **Addins Kullanma**: RStudio'nun "Addins" (Eklentiler) menüsü, kullanıcıların özel işlevleri ekleyebileceği bir bölümdür. Bu sayede belirli işlemleri hızlıca gerçekleştirebilirsiniz.

RStudio'nun bu kişiselleştirme seçenekleri, kullanıcıların kendi ihtiyaçlarına ve tercihlerine göre IDE'yi özelleştirmelerine olanak tanır. Bu şekilde, RStudio'yu daha verimli ve kişiselleştirilmiş bir şekilde kullanabilirsiniz. RStudio'nun ana bileşenleri ve temel özellikleri ise şunlardır:

1. **Script Editörü**: RStudio'nun sol üst kısmında yer alan bu bölüm, R kodlarını yazmak, düzenlemek ve çalıştırmak için kullanılır. Renk vurguları, otomatik tamamlama ve hata işaretleme gibi birçok yazılım geliştirme özelliği içerir.

2. **Environment (Çalışma Ortamı)** : Sağ üst köşede bulunan "Çalışma Ortamı" sekmesi, çalışan nesneleri ve değişkenleri görüntülemenizi sağlar. "Files" sekmesi ise projenizdeki dosyaları ve klasörleri görüntülemenize yardımcı olur.

3. **Console**: Alt sol köşede bulunan bu bölüm, R kodlarını anlık olarak çalıştırmanıza ve sonuçları görmesinize olanak tanır. R komutlarını doğrudan konsola yazabilir ve çalıştırabilirsiniz.

4. **Diğer Sekmeler** : RStudio, çeşitli grafikler ve görselleştirmeler oluşturmanıza olanak tanır. R koduyla çizilen grafikler, "**Plots**" sekmesinde görüntülenir. Bunu yanısıra "**Help"** kısmında fonksiyonlar ile ilgili bilgi alınabilir,"**Packages"** kısmından ise paket yükleme vb. işler yapılabilir.

Part I

R Programlamaya Giriş

R kodunun çalıştırılması oldukça basittir ve R Studio gibi entegre geliştirme ortamları (IDE'ler) kullanırken daha da kolaylaşır. R kodunu çalıştırmak için temel adımlar:

1. **R Studio'yu Açın**: İlk adım, R Studio veya başka bir R IDE'sini açmaktır.
2. **Yeni Bir script uluşturun veya mevcut bir script kullanın**:
   - R Studio'da, sol üst köşede bulunan "File" (Dosya) menüsünden "New Script"seçeneği ile yeni bir R scripti oluşturabilirsiniz.
   - Mevcut bir scripte gitmek istiyorsanız, "File" menüsünden "Open Script" seçeneğini kullanabilirsiniz.
3. **R Kodunu Scripte Yazın**: Oluşturduğunuz veya açtığınız R skriptinde, R kodlarını yazın veya yapıştırın. Örneğin, basit bir hesaplama yapmak için aşağıdaki kodu kullanabilirsiniz:

```
x <- 5
y <- 10
z <- x + y
z
```

```
[1] 15
```

4. **Kodu Çalıştırma**:
   - Çalıştırmak istediğiniz kodu seçin veya imleci çalıştırmak istediğiniz satıra getirin.
   - Çalıştırma işlemi için aşağıdaki yöntemlerden birini kullanabilirsiniz:
     - Klavyede varsayılan olarak "Ctrl+Enter" (Windows/Linux) veya "Command+Enter" (Mac) tuş kombinasyonunu kullanabilirsiniz.
     - R Studio'daki "Run" (Çalıştır) düğmesini veya "Run" (Çalıştır) menüsünü kullanabilirsiniz.
     - Çalıştırmak istediğiniz kodu seçtikten sonra sağ tıklarsanız, "Run" (Çalıştır) seçeneğini göreceksiniz.
5. **Sonuçları İnceleyin**: Çalıştırılan kodun sonuçları konsol penceresinde veya çıktı bölümünde görüntülenir. Örneğin, yukarıdaki örnekte "z" değişkeninin değeri olan "15" sonucunu göreceksiniz.

> ⚠ Dikkat
>
> Bir script üzerinden çalıştırılan R kodunun sonuçlarını sol alt kısımda yer alan Console bölümünde görebilirsiniz. Aynı şekilde kodu Console bölümüne yazıp Enter tuşuna

bastığınızda yine sonuç alabilirsiniz. Ancak script içerisinde yazılan kodları bir `.R` uzantılı dosya olarak saklama ve daha sonradan bu dosyaya ulaşma şansınız varken, Console ile çalıştırılan kodları bir `.R` dosyası olarak saklama şansınız yoktur. Console tarafındaki sonuçlar geçici olarak ekranda kalır ve R Studio'yu kapatıp açtığınızda tekrar yazdığınız ve çalıştırdığınız kodlara ulaşamayabilirsiniz.

İpucu

Console tarafına yansıyan kodların ve sonuçların farklı formatlarda saklama şansımız vardır. Bunun için `sink` fonksiyonunu araştırmanızı önerebilirim.

1 Temel Fonksiyonlar

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()
```

```
[1] "D:/Akademi ve Veri Bilimi/Data Science/Github/r-book-tr"
```

- `dir` veya `list.files` komutları ile dizinde yer alan dosyalar öğrenilebilir.
- `dir.create` komutu ile yeni bir klasör oluşturmak mümkündür.
- `file.exists` kullanılarak klasörün var olup olmadığı sorgulanabilir.

1.2 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.

```
#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("package")`** yüklü olan paketi getirir.
- **`installed.packages()`** yüklü olan paketleri listeler
- **`options(digits=10)`** sayılarda ondalık kısmın basamak sayısını ifade eder.
- **`help()`** fonksiyonu ya da **`?`** ile bir fonksiyon hakkında yardım alınabilir. Örneğin mean fonksiyonu ile ilgili yardım almak için scripte **`?mean`** ya da help(mean) yazmanız ve çalıştırmanız yeterlidir. Bunun yanı sıra R Studio penceresinin sağ alt kısmındaki help alanını kullanabilirsiniz.

1.3 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.

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

```
[1] "Merhaba Arkadaşlar"
```

1.4 Matematiksel Operatörler

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

```r
3+5
```

```
[1] 8
```

```r
7*8
```

```
[1] 56
```

```r
88/2
```

```
[1] 44
```

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

```
[1] 45
```

```r
9^2 # karesini alır
```

```
[1] 81
```

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

```
[1] 9
```

```r
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
```

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

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

1.5 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: $x|y$

```r
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
```

2 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
```

```r
b <- 5
class(b)
```

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

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

```
[1] FALSE
```

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

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

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

```
[1] TRUE
```

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

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

```r
# complex

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

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

```r
# 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
```

2.1 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
```

```r
v3 / v4
```

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

```r
2 * v3 - v4
```

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

```r
# 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
```

```r
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
```

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

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

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

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

```r
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
```

```r
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
```

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

```
[1] TRUE
```

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

```
[1] TRUE
```

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

```
[1] TRUE
```

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

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

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

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

```r
# 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
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
[1] 6
```

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

```
[1] 6
```

```r
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
```

2.2 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.

```r
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
```

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

```
[1] TRUE
```

```r
dim(matris)
```

```
[1] 5 4
```

```r
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,]  1.63300517   0.3295272   0.3579092 -0.42093148
[2,]  0.01508367  -0.9170607  -0.3107895 -0.01365056
[3,]  1.02474919   0.4910224   1.1133076 -0.24962662
[4,]  0.47180018   0.3558606  -0.3443041  1.34791720
[5,] 80.72442891  94.8813873  94.1792166 96.68723244
```

```
[6,] 80.85526712  90.3485519  93.1634012 86.30020670
[7,] 94.23016173  84.2688882  86.7271974 88.84577021
[8,] 94.62141182 101.7722473 100.5693835 89.25730795
```

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

```
           A           B           C           D
S  1.63300517   0.3295272   0.3579092 -0.42093148
T  0.01508367  -0.9170607  -0.3107895 -0.01365056
U  1.02474919   0.4910224   1.1133076 -0.24962662
V  0.47180018   0.3558606  -0.3443041  1.34791720
W 80.72442891  94.8813873  94.1792166 96.68723244
X 80.85526712  90.3485519  93.1634012 86.30020670
Y 94.23016173  84.2688882  86.7271974 88.84577021
Z 94.62141182 101.7722473 100.5693835 89.25730795
```

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

```
[1] 1.633005
```

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

```
[1] 0.3558606
```

```r
m[,2] # 2. sütun elemanları
```

```
         S          T          U          V          W          X
 0.3295272 -0.9170607  0.4910224  0.3558606 94.8813873 90.3485519
         Y          Z
84.2688882 101.7722473
```

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

```
           A           B           C           D
S  1.63300517   0.3295272   0.3579092 -0.42093148
T  0.01508367  -0.9170607  -0.3107895 -0.01365056
V  0.47180018   0.3558606  -0.3443041  1.34791720
W 80.72442891  94.8813873  94.1792166 96.68723244
X 80.85526712  90.3485519  93.1634012 86.30020670
Y 94.23016173  84.2688882  86.7271974 88.84577021
Z 94.62141182 101.7722473 100.5693835 89.25730795
```

```
# 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  1.6330052  0.01508367  1.0247492  0.4718002 80.72443 80.85527 94.23016
B  0.3295272 -0.91706072  0.4910224  0.3558606 94.88139 90.34855 84.26889
C  0.3579092 -0.31078952  1.1133076 -0.3443041 94.17922 93.16340 86.72720
D -0.4209315 -0.01365056 -0.2496266  1.3479172 96.68723 86.30021 88.84577
          Z
A  94.62141
B 101.77225
C 100.56938
D  89.25731
```

```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
```

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

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

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

```
[1] 4 6
```

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

```
[1] 2 3
```

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

```
[1] 3 7
```

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

```
[1] 1.5 3.5
```

2.3 Listeler

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

```r
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
```

```r
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"
```

2.4 dataframe

Veri çerçevesi (dataframe), 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.

```r
# 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 ...
```

```r
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
```

```
# 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
```

```
# 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
```

```
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
```

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

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

```r
colnames(data)
```

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

```r
# 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>
```

```r
# 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
```

```
# 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>
```

```
# 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
```

```
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
```

2.5 tibble

tibble, Hadley Wickham tarafından geliştirilen ve **dplyr** paketi ile sıkça kullanılan bir veri yapısıdır. **tibble**, **data.frame**'e benzerdir, ancak bazı önemli farklar vardır. **tibble**, daha düzenli ve okunabilir bir çıktı üretir ve bazı varsayılan davranışları **data.frame**'den farklıdır. Modern data.frame olarak tanımlanmaktadır.

```r
# tibble örneği
library(tibble)

ogrenciler_tibble <- tribble(
  ~Ad,     ~Yas, ~Cinsiyet,
  "Ali",   20,   "Erkek",
  "Ayşe",  22,   "Kadın",
  "Mehmet", 21,  "Erkek",
  "Zeynep", 23,  "Kadın"
)

# tibble'ı görüntüleme
print(ogrenciler_tibble)
```

```
# A tibble: 4 x 3
  Ad       Yas Cinsiyet
  <chr>  <dbl> <chr>
1 Ali       20 Erkek
2 Ayşe      22 Kadın
3 Mehmet    21 Erkek
4 Zeynep    23 Kadın
```

Yukarıdaki örnekte, "ogrenciler_tibble" adında bir **tibble** oluşturuldu. **tibble**, sütun adlarını ve içeriği daha okunabilir bir şekilde görüntüler ve sütunların başlık ve veri tipi (**~Ad**, **~Yas**, **~Cinsiyet**) gibi özelliklerini korur.

> **Not**
>
> Hem **dataframe** hem de **tibble** veri analizi ve işleme işlemlerinde kullanışlıdır. Hangi veri yapısını kullanacağınız, projenizin gereksinimlerine ve kişisel tercihinize bağlıdır. Özellikle veri analizi için **dplyr** gibi paketlerle çalışırken **tibble** tercih edilir.

2.6 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 **factor()** 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
```

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

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

```r
# 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" ...
```

```r
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
```

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

```
[1] TRUE
```

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

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

```r
# 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
```

```r
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
```

```r
# 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
```

```r
# 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
```

3 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))
```

```r
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
```

```r
# 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
```

```r
# 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
```

4 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.

4.1 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
```

4.2 Döngüler

- **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
```

```r
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
```

5 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] "2023-12-22"
```

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

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

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

class(myDate)
```

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

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

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

```
# 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"
```

```r
  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] "2023-12-22 14:08:07 +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"
```

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

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

```r
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] "2023-12-22 14:08:08 +03"
```

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

```
[1] 2023
```

```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] 51
```

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

```
[1] 22
```

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

```
[1] 6
```

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

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

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

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

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

```
[1] 356
```

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

```
[1] 14
```

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

```
[1] 8
```

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

```
[1] 8.052078
```

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 2023"
```

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

```
[1] "Aralık 2023"
```

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

```
[1] "2023-12"
```

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

```
[1] "2023 Q4"
```

```r
# 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
```

6 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"
```

```r
# 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"
```

7 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 fonksiyon uygular.

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

```
            [,1]       [,2]        [,3]        [,4]       [,5]       [,6]
[1,] -0.07947808 -1.3568243 -1.21094178 -0.30369633  0.1390995  0.9708739
[2,] -0.04101514  0.9724976 -1.36226823 -0.01186826 -0.2059228 -0.3819433
[3,]  0.44609981 -0.5751676 -0.05835586  0.77811630 -1.8717467 -0.1117794
[4,] -2.08459080  0.9191161  2.27791529  0.04936040  0.6511075  1.5212365
[5,] -0.05662142  0.3070069 -0.56410295 -0.20036113 -0.5275962 -0.3555195
```

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

```
[1] -1.8156056  0.2666287 -0.9177535  0.3115510 -1.8150588  1.6428683
```

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

```
[1] -1.840967 -1.030520 -1.392833  3.334145 -1.397194
```

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

```
[1] 0.9869706 1.0052799 1.4717520 0.4245602 0.9498157 0.8662740
```

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

```
[1] -0.3068278 -0.1717534 -0.2321389  0.5556908 -0.2328657
```

```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)
```

```r
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
```

```
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
```

```
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]  0.4466595 -0.5340010  1.7102845 -0.6205973 -0.1596570  1.2610486
 [7]  2.5453549  1.3945215  0.4488399  0.0109215

$c
 [1]  2.61477762  2.20893473  1.13679889  1.91021039  1.13628960  2.06096757
 [7] -0.91894907  1.78111844  1.38184741 -0.51538007  0.01835256  0.44811131
[13]  0.31244868  1.43474074  1.01713480  0.23069548  0.22301994  0.34112263
[19] -0.64446013  0.25022539

$d
  [1] 3.718755 5.373574 6.410670 5.396454 3.165064 5.450004 5.424223 5.065613
  [9] 4.973928 4.157502 3.063955 3.846201 5.003420 5.272545 4.250888 5.854513
 [17] 4.904233 4.997541 5.465176 6.191325 4.875607 5.715689 3.538458 3.885838
 [25] 5.745441 5.074683 4.073779 4.704401 4.253970 5.031568 3.917872 7.049186
 [33] 4.277012 4.571442 6.659513 6.057098 5.210465 4.566339 4.256626 5.526688
```

```
 [41] 5.591570 4.227644 6.533361 3.679753 5.032688 5.189173 4.784057 4.416259
 [49] 2.082980 4.362379 4.850522 6.400405 4.512348 3.567207 5.312086 6.007473
 [57] 2.773751 6.278426 5.704528 5.507890 3.936025 5.452980 3.311758 4.488489
 [65] 5.438833 4.672485 4.507835 5.880133 5.091001 4.237346 5.819003 4.488003
 [73] 4.855736 5.810382 5.165495 4.040738 6.999418 4.157152 4.949945 4.133546
 [81] 6.029311 4.564896 5.368076 5.170841 4.477861 5.050123 5.173035 6.794354
 [89] 4.458431 5.402393 3.763964 4.022414 3.585853 5.401459 6.573901 6.050615
 [97] 5.733294 6.015160 4.943448 5.864351
```

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

```
$a
[1] 2.5

$b
[1] 0.6503375

$c
[1] 0.8214003

$d
[1] 4.956718
```

```
# 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
```

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

```
speed  dist
  770  2149
```

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

```
speed  dist
   15    36
```

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

```
speed  dist
15.40 42.98
```

```
  # 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
```

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

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

```
  # tapply

  df <- data.frame(x =round(runif(15,min=1,max=10)),
```

```
                    group=sample(c(1:3),15,replace = TRUE))
df
```

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

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

```
       1        2        3
7.000000 6.166667 4.400000
```

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

```
 1  2  3
28 37 22
```

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

```
1 2 3
4 6 5
```

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

```
$`1`
[1]  4 10

$`2`
[1]  1 10

$`3`
[1] 3 6
```

8 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" ...
```

```r
# 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 ...
```

```r
# 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")
```

Part II

Veri Manipulasyonu

Veri manipülasyonu, veri çerçeveleri üzerinde verileri dönüştürmek, filtrelemek, birleştirmek veya yeniden düzenlemek gibi işlemleri içeren önemli bir veri bilimi becerisidir. R programlama dili, veri manipülasyonu için oldukça güçlü ve esnek bir araç sunar. Bu yazıda, R kullanarak veri manipülasyonunu nasıl yapabileceğinizi öğreneceğiz.

Veri manipülasyonu için R'da yaygın olarak kullanılan iki ana kavram, "veri çerçeveleri" ve "paketler"dir. Veri çerçeveleri, verileri tablo şeklinde düzenleyen ve işleyen veri yapılarıdır. R'da veri çerçeveleri, **`data.frame`** türünden nesnelerdir. Veri manipülasyonu için kullanabileceğiniz birçok paket vardır, ancak en yaygın kullanılanlar arasında **`dplyr`** ve **`tidyr`** bulunur. Bu paketler, veri manipülasyonunu kolaylaştırmak için bir dizi işlev içerir.

`dplyr`, RStudio'dan Hadley Wickham tarafından geliştirilmiş ve en yaygın veri işleme zorluklarını çözmenize yardımcı olan bir veri işleme dilbilgisidir. **`dplyr`** paketi, **`devtools`** paketi ve `install_github()` fonksiyonu kullanılarak **CRAN'dan** veya **GitHub'dan** kurulabilir. GitHub deposu genellikle paketteki en son güncellemeleri ve geliştirme sürümünü içerir.

CRAN sayfasından yüklemek için;

```
> install.packages("dplyr")
```

GitHub sayfasından yüklemek için;

```
> install_github("hadley/dplyr")
```

dplyr paketinde sıklıkla kullanılan fonksiyonlar şunlardır:

- **`select`** : veri çerçevesinden istenilen sütunları seçer.

- **filter** : mantıksal koşullara dayalı olarak bir veri çerçevesinden satırları filtreler.
- **arrange** : satıları sıralar.
- **rename** : sütun isimlerini yeniden isimlendirir.
- **mutate** : yeni değişkenler/sütunlar ekler veya mevcut değişkenleri dönüştürür.
- **summarise/ summarize** : veri çerçevesindeki farklı değişkenlerin özet istatistiklerini oluşturur
- **%>%** (pipe) operatörü birden çok eylemi ardışık düzende zincirleme şekilde birbirine bağlamak için kullanılır.

Veri manipülasyonu ile örnekler bazen küçük veri setleri oluşturulacaktır bazen de 2015 yılı ABD nüfus sayımına ilişkin **counties** veri seti kullanılacaktır. Bu veri setinde eyalet ve şehir detayında nüfus, gelir, ırk, coğrafi yapı, işgücü gibi değişkenler yer almaktadır.

```
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
counties <- readRDS("datasets/counties.rds")

# veri setinin yapısı hakkında bilgi sağlar
glimpse(counties)
```

```
Rows: 3,138
Columns: 40
$ census_id          <chr> "1001", "1003", "1005", "1007", "1009", "1011", "10~
$ state              <chr> "Alabama", "Alabama", "Alabama", "Alabama", "Alabam~
$ county             <chr> "Autauga", "Baldwin", "Barbour", "Bibb", "Blount", ~
$ region             <chr> "South", "South", "South", "South", "South", "South~
$ metro              <chr> "Metro", "Metro", "Nonmetro", "Metro", "Metro", "No~
```

```
$ population         <dbl> 55221, 195121, 26932, 22604, 57710, 10678, 20354, 1~
$ men                <dbl> 26745, 95314, 14497, 12073, 28512, 5660, 9502, 5627~
$ women              <dbl> 28476, 99807, 12435, 10531, 29198, 5018, 10852, 603~
$ hispanic           <dbl> 2.6, 4.5, 4.6, 2.2, 8.6, 4.4, 1.2, 3.5, 0.4, 1.5, 7~
$ white              <dbl> 75.8, 83.1, 46.2, 74.5, 87.9, 22.2, 53.3, 73.0, 57.~
$ black              <dbl> 18.5, 9.5, 46.7, 21.4, 1.5, 70.7, 43.8, 20.3, 40.3,~
$ native             <dbl> 0.4, 0.6, 0.2, 0.4, 0.3, 1.2, 0.1, 0.2, 0.2, 0.6, 0~
$ asian              <dbl> 1.0, 0.7, 0.4, 0.1, 0.1, 0.2, 0.4, 0.9, 0.8, 0.3, 0~
$ pacific            <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0~
$ citizens           <dbl> 40725, 147695, 20714, 17495, 42345, 8057, 15581, 88~
$ income             <dbl> 51281, 50254, 32964, 38678, 45813, 31938, 32229, 41~
$ income_err         <dbl> 2391, 1263, 2973, 3995, 3141, 5884, 1793, 925, 2949~
$ income_per_cap     <dbl> 24974, 27317, 16824, 18431, 20532, 17580, 18390, 21~
$ income_per_cap_err <dbl> 1080, 711, 798, 1618, 708, 2055, 714, 489, 1366, 15~
$ poverty            <dbl> 12.9, 13.4, 26.7, 16.8, 16.7, 24.6, 25.4, 20.5, 21.~
$ child_poverty      <dbl> 18.6, 19.2, 45.3, 27.9, 27.2, 38.4, 39.2, 31.6, 37.~
$ professional       <dbl> 33.2, 33.1, 26.8, 21.5, 28.5, 18.8, 27.5, 27.3, 23.~
$ service            <dbl> 17.0, 17.7, 16.1, 17.9, 14.1, 15.0, 16.6, 17.7, 14.~
$ office             <dbl> 24.2, 27.1, 23.1, 17.8, 23.9, 19.7, 21.9, 24.2, 26.~
$ construction       <dbl> 8.6, 10.8, 10.8, 19.0, 13.5, 20.1, 10.3, 10.5, 11.5~
$ production         <dbl> 17.1, 11.2, 23.1, 23.7, 19.9, 26.4, 23.7, 20.4, 24.~
$ drive              <dbl> 87.5, 84.7, 83.8, 83.2, 84.9, 74.9, 84.5, 85.3, 85.~
$ carpool            <dbl> 8.8, 8.8, 10.9, 13.5, 11.2, 14.9, 12.4, 9.4, 11.9, ~
$ transit            <dbl> 0.1, 0.1, 0.4, 0.5, 0.4, 0.7, 0.0, 0.2, 0.2, 0.2, 0~
$ walk               <dbl> 0.5, 1.0, 1.8, 0.6, 0.9, 5.0, 0.8, 1.2, 0.3, 0.6, 1~
$ other_transp       <dbl> 1.3, 1.4, 1.5, 1.5, 0.4, 1.7, 0.6, 1.2, 0.4, 0.7, 1~
$ work_at_home       <dbl> 1.8, 3.9, 1.6, 0.7, 2.3, 2.8, 1.7, 2.7, 2.1, 2.5, 1~
$ mean_commute       <dbl> 26.5, 26.4, 24.1, 28.8, 34.9, 27.5, 24.6, 24.1, 25.~
$ employed           <dbl> 23986, 85953, 8597, 8294, 22189, 3865, 7813, 47401,~
$ private_work       <dbl> 73.6, 81.5, 71.8, 76.8, 82.0, 79.5, 77.4, 74.1, 85.~
$ public_work        <dbl> 20.9, 12.3, 20.8, 16.1, 13.5, 15.1, 16.2, 20.8, 12.~
$ self_employed      <dbl> 5.5, 5.8, 7.3, 6.7, 4.2, 5.4, 6.2, 5.0, 2.8, 7.9, 4~
$ family_work        <dbl> 0.0, 0.4, 0.1, 0.4, 0.4, 0.0, 0.2, 0.1, 0.0, 0.5, 0~
$ unemployment       <dbl> 7.6, 7.5, 17.6, 8.3, 7.7, 18.0, 10.9, 12.3, 8.9, 7.~
$ land_area          <dbl> 594.44, 1589.78, 884.88, 622.58, 644.78, 622.81, 77~
```

9 select

Tabloyu (veri çerçevesi) seçmek ve dönüştürmek için R'da **dplyr** paketinde bulunan **select()** fonksiyonu oldukça kullanışlıdır. Bu fonksiyon, belirli sütunları seçmek veya sütun adlarını değiştirmek için kullanılır. **select()** fonksiyonunu kullanarak veri çerçevesinde sütunları seçme ve dönüştürme işlemlerinin nasıl yapıldığına dair aşağıda örnekler mevcuttur.

> **Not**
>
> **select()** fonksiyonu ayrıca sütunları seçerken veya döndürürken bazı özel işlevler de kullanmanıza olanak tanır. Örneğin, **starts_with()**, **ends_with()**, **contains()** gibi işlevleri kullanarak sütun adlarının belirli bir örüntüyü karşılayanları seçebilirsiniz. Bu fonksiyon, veri manipülasyonu işlemlerinde oldukça kullanışlıdır ve veri çerçevelerini istediğiniz şekilde özelleştirmenize yardımcı olur.

```
# library(dplyr)
# counties <- readRDS("datasets/counties.rds")

# belirli sütunları seçmek
counties %>% select(state, county, population, unemployment)
```

```
# A tibble: 3,138 x 4
   state   county   population unemployment
   <chr>   <chr>         <dbl>        <dbl>
 1 Alabama Autauga       55221          7.6
 2 Alabama Baldwin      195121          7.5
 3 Alabama Barbour       26932         17.6
 4 Alabama Bibb          22604          8.3
 5 Alabama Blount        57710          7.7
 6 Alabama Bullock       10678         18
 7 Alabama Butler        20354         10.9
 8 Alabama Calhoun      116648         12.3
 9 Alabama Chambers      34079          8.9
10 Alabama Cherokee      26008          7.9
# i 3,128 more rows
```

```r
# belli aralıkta bütün sütunların seçilmesi
counties %>% select(state, county, drive:work_at_home)
```

```
# A tibble: 3,138 x 8
   state   county   drive carpool transit  walk other_transp work_at_home
   <chr>   <chr>    <dbl>   <dbl>   <dbl> <dbl>        <dbl>        <dbl>
 1 Alabama Autauga   87.5     8.8     0.1   0.5          1.3          1.8
 2 Alabama Baldwin   84.7     8.8     0.1   1            1.4          3.9
 3 Alabama Barbour   83.8    10.9     0.4   1.8          1.5          1.6
 4 Alabama Bibb      83.2    13.5     0.5   0.6          1.5          0.7
 5 Alabama Blount    84.9    11.2     0.4   0.9          0.4          2.3
 6 Alabama Bullock   74.9    14.9     0.7   5            1.7          2.8
 7 Alabama Butler    84.5    12.4     0     0.8          0.6          1.7
 8 Alabama Calhoun   85.3     9.4     0.2   1.2          1.2          2.7
 9 Alabama Chambers  85.1    11.9     0.2   0.3          0.4          2.1
10 Alabama Cherokee  83.9    12.1     0.2   0.6          0.7          2.5
# i 3,128 more rows
```

```r
# belirli bir ifadeyi içeren sütunları seçmek
counties %>% select(state, county, contains("employed"))
```

```
# A tibble: 3,138 x 4
   state   county   employed self_employed
   <chr>   <chr>       <dbl>         <dbl>
 1 Alabama Autauga     23986           5.5
 2 Alabama Baldwin     85953           5.8
 3 Alabama Barbour      8597           7.3
 4 Alabama Bibb         8294           6.7
 5 Alabama Blount      22189           4.2
 6 Alabama Bullock      3865           5.4
 7 Alabama Butler       7813           6.2
 8 Alabama Calhoun     47401           5
 9 Alabama Chambers    13689           2.8
10 Alabama Cherokee    10155           7.9
# i 3,128 more rows
```

```r
# belirli bir ifade ile başyalan sütunları seçmek
counties %>% select(state, county, starts_with("income"))
```

```
# A tibble: 3,138 x 6
   state   county   income income_err income_per_cap income_per_cap_err
   <chr>   <chr>     <dbl>      <dbl>          <dbl>               <dbl>
 1 Alabama Autauga   51281       2391          24974                1080
 2 Alabama Baldwin   50254       1263          27317                 711
 3 Alabama Barbour   32964       2973          16824                 798
 4 Alabama Bibb      38678       3995          18431                1618
 5 Alabama Blount    45813       3141          20532                 708
 6 Alabama Bullock   31938       5884          17580                2055
 7 Alabama Butler    32229       1793          18390                 714
 8 Alabama Calhoun   41703        925          21374                 489
 9 Alabama Chambers  34177       2949          21071                1366
10 Alabama Cherokee  36296       1710          21811                1556
# i 3,128 more rows
```

```r
# belirli bir ifade ile biten sütunları seçmek
counties %>% select(state, county, ends_with("work"))
```

```
# A tibble: 3,138 x 5
   state   county   private_work public_work family_work
   <chr>   <chr>           <dbl>       <dbl>       <dbl>
 1 Alabama Autauga          73.6        20.9         0
 2 Alabama Baldwin          81.5        12.3         0.4
 3 Alabama Barbour          71.8        20.8         0.1
 4 Alabama Bibb             76.8        16.1         0.4
 5 Alabama Blount           82          13.5         0.4
 6 Alabama Bullock          79.5        15.1         0
 7 Alabama Butler           77.4        16.2         0.2
 8 Alabama Calhoun          74.1        20.8         0.1
 9 Alabama Chambers         85.1        12.1         0
10 Alabama Cherokee         73.1        18.5         0.5
# i 3,128 more rows
```

```r
# belirli sütunları hariç tutarak seçmek
counties %>% select(census_id:population,-c(men:land_area))
```

```
# A tibble: 3,138 x 6
   census_id state   county   region metro    population
   <chr>     <chr>   <chr>    <chr>  <chr>         <dbl>
 1 1001      Alabama Autauga  South  Metro         55221
```

```
 2 1003      Alabama Baldwin  South  Metro        195121
 3 1005      Alabama Barbour  South  Nonmetro      26932
 4 1007      Alabama Bibb     South  Metro         22604
 5 1009      Alabama Blount   South  Metro         57710
 6 1011      Alabama Bullock  South  Nonmetro      10678
 7 1013      Alabama Butler   South  Nonmetro      20354
 8 1015      Alabama Calhoun  South  Metro        116648
 9 1017      Alabama Chambers South  Nonmetro      34079
10 1019      Alabama Cherokee South  Nonmetro      26008
# i 3,128 more rows
```

```
  # belirli veri tipindeki sütunları seçmek
  counties %>% select(where(is.character))
```

```
# A tibble: 3,138 x 5
   census_id state   county   region metro
   <chr>     <chr>   <chr>    <chr>  <chr>
 1 1001      Alabama Autauga  South  Metro
 2 1003      Alabama Baldwin  South  Metro
 3 1005      Alabama Barbour  South  Nonmetro
 4 1007      Alabama Bibb     South  Metro
 5 1009      Alabama Blount   South  Metro
 6 1011      Alabama Bullock  South  Nonmetro
 7 1013      Alabama Butler   South  Nonmetro
 8 1015      Alabama Calhoun  South  Metro
 9 1017      Alabama Chambers South  Nonmetro
10 1019      Alabama Cherokee South  Nonmetro
# i 3,128 more rows
```

```
  # select ile kolon adı değiştirmek
  counties %>% select(census_id,pop = population)
```

```
# A tibble: 3,138 x 2
   census_id    pop
   <chr>      <dbl>
 1 1001       55221
 2 1003      195121
 3 1005       26932
 4 1007       22604
 5 1009       57710
```

```
 6 1011       10678
 7 1013       20354
 8 1015      116648
 9 1017       34079
10 1019       26008
# i 3,128 more rows
```