R Programlama

Muhammed Fatih Tüzen

İçindekiler

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.

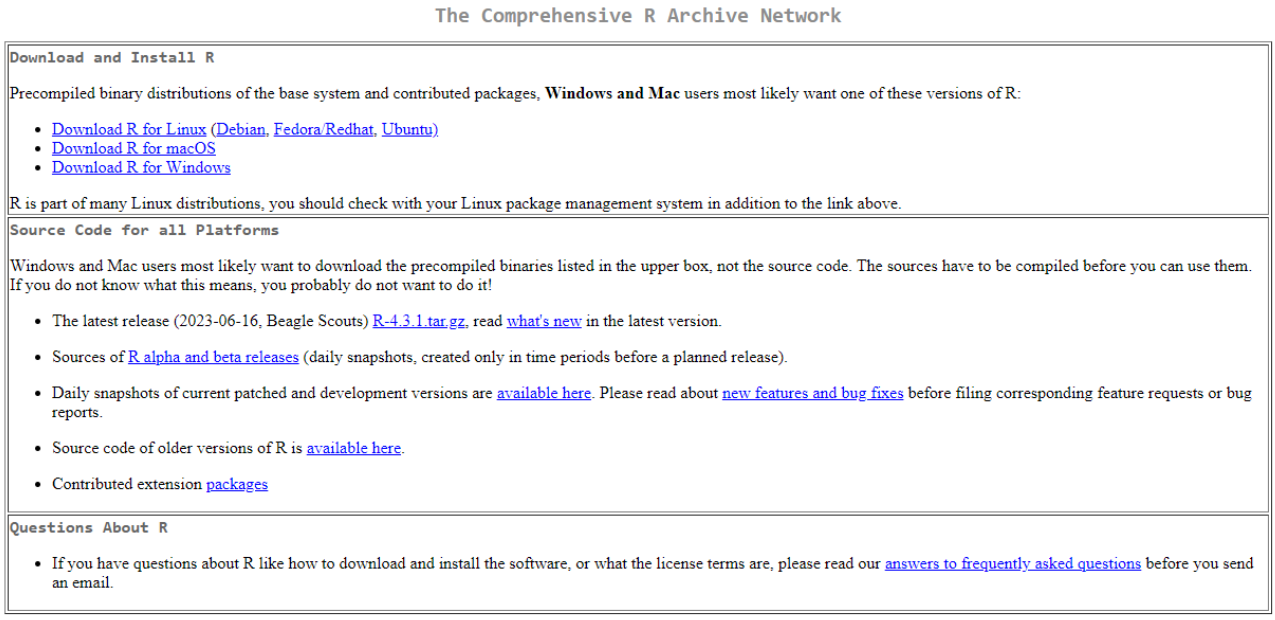
> Tavsiye
>
> R programlamayı öğrenmek ve geliştirmek için sürekli olarak yeni kaynaklar ve materyaller üretilmektedir. İhtiyacınıza ve seviyenize uygun kaynakları seçmek için zaman ayırın ve kendi hızınıza göre öğrenmeye devam edin.

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

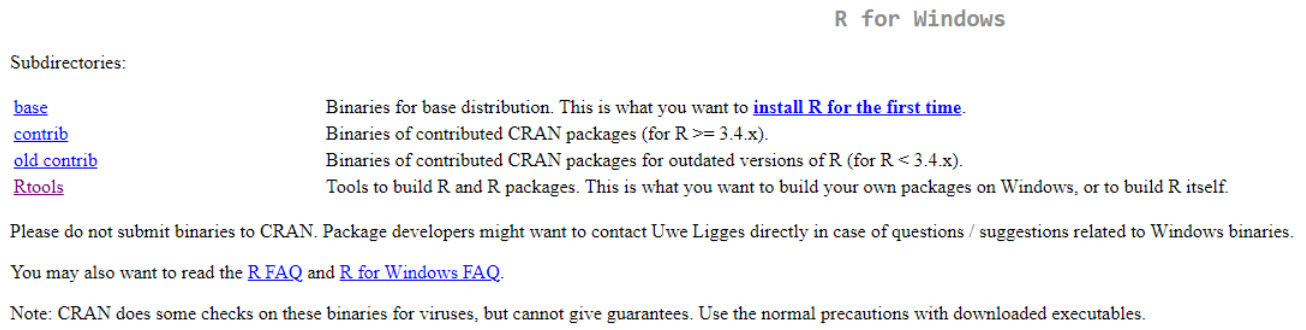
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- 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:

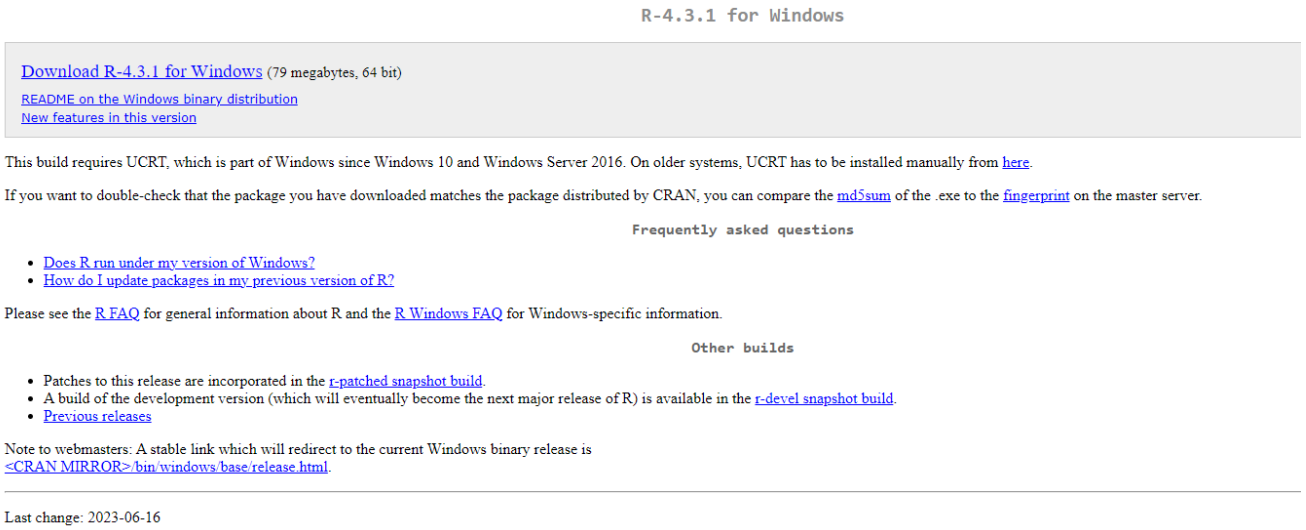
base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
Rtools	Tools to build R and R packages. This is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

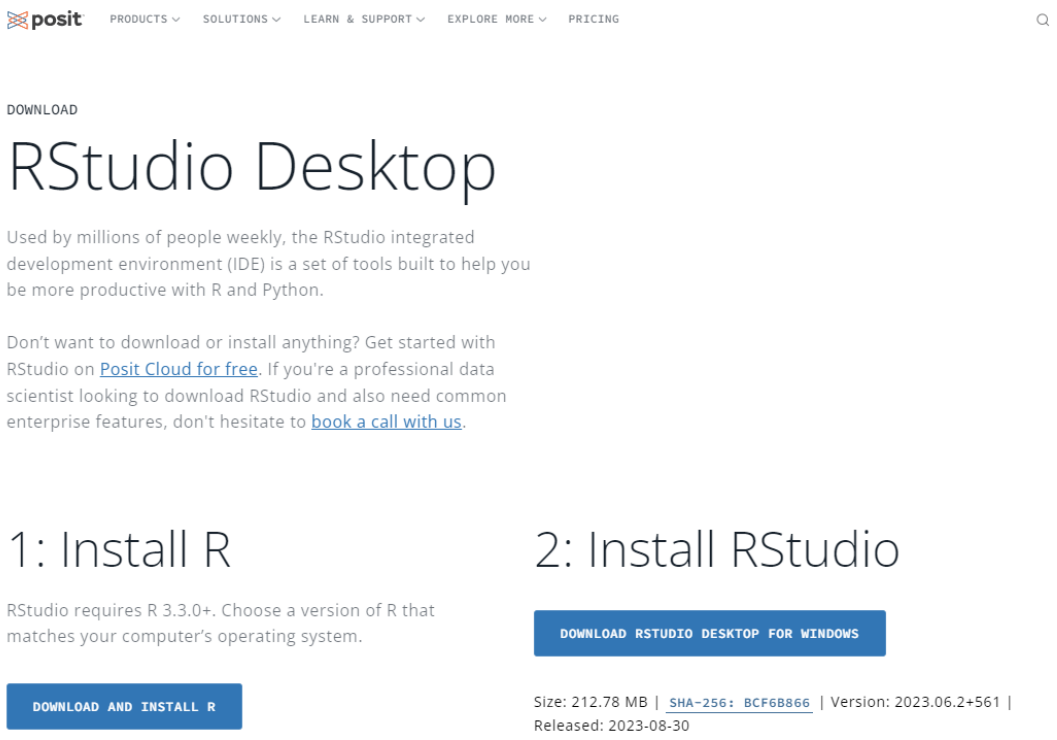
5. İndirilen dosyayı çift tıklayarak çalıştırın ve yükleyiciyi başlatın.
6. Yükleyici, R'nin temel sürümünü yüklemek için sizi yönlendirecektir. Varsayılan ayarları genellikle kabul edebilirsiniz.
7. Kurulum tamamlandığında, R'yi çalıştırmak için masaüstünüzde veya Başlat menüsünde **"R"** simgesini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

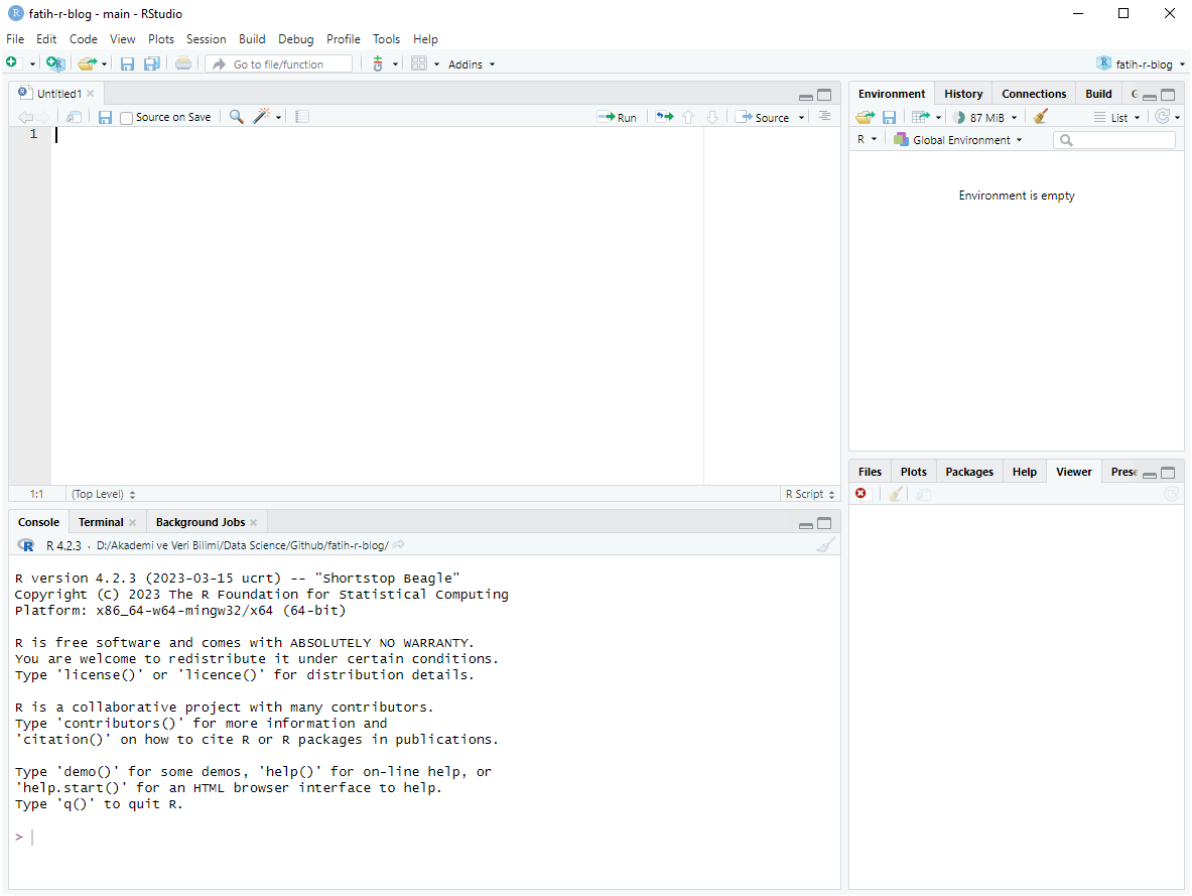
R Studio'nun kurulumu aşağıdaki adımlarla gerçekleştirilebilir:

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
```

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

```
[1] TRUE
```

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

```
[1] FALSE
```

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

```
[1] FALSE
```

```r
# | (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 birleştirilmesi 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,]  0.50704867  0.4712427  0.6558818 -2.11610814
[2,]  0.05134823  0.7892462 -0.1625571  0.01622168
[3,]  0.41028687 -1.5768597  1.1563391 -1.25373679
[4,] -1.39248168 -1.2943010  0.2438095  0.10115715
[5,] 88.39829181 79.0065809 95.4242607 94.73533117
```

```
[6,] 97.20877666 87.2953959 67.3123268 87.51028785
[7,] 82.40943578 82.0714739 93.3945146 82.27047621
[8,] 89.32196880 97.5144232 83.7317352 91.77888985
```

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

```
           A          B          C           D
S  0.50704867  0.4712427  0.6558818 -2.11610814
T  0.05134823  0.7892462 -0.1625571  0.01622168
U  0.41028687 -1.5768597  1.1563391 -1.25373679
V -1.39248168 -1.2943010  0.2438095  0.10115715
W 88.39829181 79.0065809 95.4242607 94.73533117
X 97.20877666 87.2953959 67.3123268 87.51028785
Y 82.40943578 82.0714739 93.3945146 82.27047621
Z 89.32196880 97.5144232 83.7317352 91.77888985
```

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

```
[1] 0.5070487
```

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

```
[1] -1.294301
```

```r
m[,2] # 2. sütun elemanları
```

```
        S         T         U         V         W         X         Y
 0.4712427 0.7892462 -1.5768597 -1.2943010 79.0065809 87.2953959 82.0714739
        Z
97.5144232
```

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

```
            A          B          C           D
S  0.50704867  0.4712427  0.6558818 -2.11610814
T  0.05134823  0.7892462 -0.1625571  0.01622168
V -1.39248168 -1.2943010  0.2438095  0.10115715
W 88.39829181 79.0065809 95.4242607 94.73533117
X 97.20877666 87.2953959 67.3123268 87.51028785
Y 82.40943578 82.0714739 93.3945146 82.27047621
Z 89.32196880 97.5144232 83.7317352 91.77888985
```

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

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

```r
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
```

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

```
           S           T          U          V        W        X        Y
A  0.5070487  0.05134823  0.4102869 -1.3924817 88.39829 97.20878 82.40944
B  0.4712427  0.78924623 -1.5768597 -1.2943010 79.00658 87.29540 82.07147
C  0.6558818 -0.16255714  1.1563391  0.2438095 95.42426 67.31233 93.39451
D -2.1161081  0.01622168 -1.2537368  0.1011572 94.73533 87.51029 82.27048
         Z
A 89.32197
B 97.51442
C 83.73174
D 91.77889
```

```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 birarada 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] "2023-12-25"
```

```
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] "2023-12-25 17:38:05 +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
```

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

```
[1] 22
```

```r
# 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.

```r
library(lubridate)
```

```
Attaching package: 'lubridate'

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

    date, intersect, setdiff, union
```

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

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

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

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

```r
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] "2023-12-25 17:38:05 +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] 52
```

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

```
[1] 25
```

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

```
[1] 2
```

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

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

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

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

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

```
[1] 359
```

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

```
[1] 17
```

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

```
[1] 38
```

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

```
[1] 5.418696
```

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,] -0.230100088 -0.70163999  0.6313825  0.08399443  0.83884104 -0.5844489
[2,]  0.008953817 -0.18261738 -0.3356496 -1.21736853  0.76982699 -1.0008397
[3,] -0.227024232 -0.85838773  0.6975943 -1.47520382 -0.84279209 -1.1574617
[4,] -0.663394437 -1.55844096  1.4997443 -0.32467163 -0.08505356  0.7144678
[5,]  0.385403307  0.08866406  0.6836027 -1.31249470  1.15790600  0.2425295
```

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

```
[1] -0.7261616 -3.2124220  3.1766742 -4.2457443  1.8387284 -1.7857529
```

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

```
[1]  0.0380290 -1.9576943 -3.8632752 -0.4173485  1.2456109
```

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

```
[1] 0.3833184 0.6393641 0.6511981 0.6870321 0.8182882 0.8084677
```

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

```
[1]  0.006338166 -0.326282391 -0.643879206 -0.069558083  0.207601812
```

```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
```

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

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

[[2]]
[1] 4 5 6 7

[[3]]
[1]  8  9 10
```

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

```
$a
[1] 1 2 3 4

$b
 [1] -0.4283601 -1.7379530 -2.7853723  1.0937715  0.8471561  0.4917568
 [7] -0.8388146 -0.6137568 -0.3567499 -0.9766411

$c
 [1]  0.39355834  0.08151052  1.51055065  0.27966401  1.75235918  0.98844990
 [7]  2.65510606  2.36991016  1.50739835  0.09163799  1.99027272  1.27243243
[13]  0.88459581  0.03233636  0.31072625  0.80059305  1.73921147 -0.32148160
[19]  2.32382295  2.13922038

$d
  [1] 3.737750 5.983020 5.280389 3.277922 4.275641 6.384722 5.392228 4.218388
  [9] 5.775689 3.498493 4.851033 3.124921 5.905383 3.630809 4.023036 5.898026
 [17] 4.133505 6.141493 4.205959 3.730453 6.234426 4.653007 6.754251 6.179029
 [25] 3.985514 5.986126 4.555791 4.575738 6.106523 3.976326 4.201788 4.550470
 [33] 6.854385 6.391990 4.285960 5.416807 5.497379 3.900122 4.295068 5.410636
```

```
 [41] 3.604568 6.241269 4.551405 4.459277 5.466613 4.506980 5.932207 4.583357
 [49] 4.704138 4.852910 4.675910 5.168269 4.138749 3.061964 5.094740 4.796135
 [57] 3.784765 5.824811 4.927007 4.755196 5.215905 4.101077 6.181052 3.507421
 [65] 6.203534 4.156585 6.450404 6.408682 4.687214 6.179298 4.243341 6.678980
 [73] 5.586735 5.147336 3.939492 4.262033 4.095497 4.061510 5.103087 4.863453
 [81] 5.753434 4.650764 6.059513 5.125056 5.226419 5.894673 6.113728 4.686735
 [89] 3.327157 4.975054 5.749537 5.006948 5.751254 5.169497 4.381528 3.919288
 [97] 5.832733 5.490146 7.514530 3.875821
```

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

```
$a
[1] 2.5

$b
[1] -0.5304963

$c
[1] 1.140094

$d
[1] 4.979869
```

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

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

```
       1        2        3
6.666667 4.400000 6.250000
```

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

```
 1  2  3
40 22 25
```

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

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

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

```
$`1`
[1] 3 9

$`2`
[1] 1 8

$`3`
[1] 4 9
```

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")
```

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 için 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)
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~
```

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.

```
# 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
```

```
# 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
```

```
# 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
```

```
# 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
```

```
# 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
```

```
# 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
```

```r
# 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
```

```r
# 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
```

arrange

dplyr paketinde bulunan **arrange()** fonksiyonu, veri çerçevesindeki satırları belirli bir sıraya göre düzenlemek için kullanılır. Bu sıralama işlemi, bir veya daha fazla sütunun değerlerine göre yapılabilir. **arrange()** fonksiyonu, veri analizi ve veri keşfi sırasında verilerinizi anlamak ve analiz etmek için önemli bir araçtır.

```
counties_selected <- counties %>%
select(state, county, population, unemployment)

# artan sıralama (ascending)
counties_selected %>%
arrange(population)
```

```
# A tibble: 3,138 x 4
   state      county    population unemployment
   <chr>      <chr>          <dbl>        <dbl>
 1 Hawaii     Kalawao           85          0
 2 Texas      King             267          5.1
 3 Nebraska   McPherson        433          0.9
 4 Montana    Petroleum        443          6.6
 5 Nebraska   Arthur           448          4
 6 Nebraska   Loup             548          0.7
 7 Nebraska   Blaine           551          0.7
 8 New Mexico Harding          565          6
 9 Texas      Kenedy           565          0
10 Colorado   San Juan         606         13.8
# i 3,128 more rows
```

```
# azalan sıralama (descending)
counties_selected %>%
arrange(desc(population))
```

```
# A tibble: 3,138 x 4
   state      county      population unemployment
   <chr>      <chr>            <dbl>        <dbl>
 1 California Los Angeles   10038388         10
 2 Illinois   Cook           5236393         10.7
 3 Texas      Harris         4356362          7.5
 4 Arizona    Maricopa       4018143          7.7
```

```
 5 California San Diego      3223096          8.7
 6 California Orange         3116069          7.6
 7 Florida    Miami-Dade     2639042         10
 8 New York   Kings          2595259         10
 9 Texas      Dallas         2485003          7.6
10 New York   Queens         2301139          8.6
# i 3,128 more rows
```

```
# birden fazla sütun seçerek sıralama
counties_selected %>%
arrange(state,desc(population))
```

```
# A tibble: 3,138 x 4
   state   county     population unemployment
   <chr>   <chr>           <dbl>        <dbl>
 1 Alabama Jefferson      659026          9.1
 2 Alabama Mobile         414251          9.8
 3 Alabama Madison        346438          8.5
 4 Alabama Montgomery     228138          8.8
 5 Alabama Shelby         203530          5.5
 6 Alabama Tuscaloosa     200458          7.6
 7 Alabama Baldwin        195121          7.5
 8 Alabama Lee            150982          7.3
 9 Alabama Morgan         119786          9.9
10 Alabama Calhoun        116648         12.3
# i 3,128 more rows
```

filter

dplyr paketindeki **filter()** fonksiyonu, veri çerçevesinde belirli bir koşulu karşılayan satırları seçmek için kullanılır. Bu fonksiyon, veri analizi sırasında verilerinizi filtrelemek ve istediğiniz verileri elde etmek için oldukça kullanışlıdır. **filter()** fonksiyonu, veri çerçevesindeki satırları seçerken belirli sütunlardaki değerlere dayalı koşulları uygulamanıza olanak tanır.

```
# sadece New York'u filtrele
counties_selected %>%
arrange(desc(population)) %>%
filter(state == "New York")
```

```
# A tibble: 62 x 4
   state    county      population unemployment
   <chr>    <chr>            <dbl>        <dbl>
 1 New York Kings          2595259         10
 2 New York Queens         2301139          8.6
 3 New York New York       1629507          7.5
 4 New York Suffolk        1501373          6.4
 5 New York Bronx          1428357         14
 6 New York Nassau         1354612          6.4
 7 New York Westchester     967315          7.6
 8 New York Erie            921584          7
 9 New York Monroe          749356          7.7
10 New York Richmond        472481          6.9
# i 52 more rows
```

```r
# işsizlik oranı 6'dan küçük olanları filtrele
counties_selected %>%
arrange(desc(population)) %>%
filter(unemployment < 6)
```

```
# A tibble: 949 x 4
   state    county       population unemployment
   <chr>    <chr>             <dbl>        <dbl>
 1 Virginia Fairfax         1128722          4.9
 2 Utah     Salt Lake       1078958          5.8
 3 Hawaii   Honolulu         984178          5.6
 4 Texas    Collin           862215          4.9
 5 Texas    Denton           731851          5.7
 6 Texas    Fort Bend        658331          5.1
 7 Kansas   Johnson          566814          4.5
 8 Maryland Anne Arundel     555280          5.9
 9 Colorado Jefferson        552344          5.9
10 Utah     Utah             551957          5.5
# i 939 more rows
```

```r
# birden fazla koşul
counties_selected %>%
arrange(desc(population)) %>%
filter(state == "New York",unemployment < 6)
```

```
# A tibble: 5 x 4
  state    county     population unemployment
  <chr>    <chr>           <dbl>        <dbl>
1 New York Tompkins       103855          5.9
2 New York Chemung         88267          5.4
3 New York Madison         72427          5.1
4 New York Livingston      64801          5.4
5 New York Seneca          35144          5.5
```

```r
# veya kullanımı
counties_selected %>%
arrange(desc(population)) %>%
filter(state == "New York"| unemployment < 6)
```

```
# A tibble: 1,006 x 4
   state    county      population unemployment
   <chr>    <chr>            <dbl>        <dbl>
 1 New York Kings          2595259         10
 2 New York Queens         2301139          8.6
 3 New York New York       1629507          7.5
 4 New York Suffolk        1501373          6.4
 5 New York Bronx          1428357         14
 6 New York Nassau         1354612          6.4
 7 Virginia Fairfax        1128722          4.9
 8 Utah     Salt Lake      1078958          5.8
 9 Hawaii   Honolulu        984178          5.6
10 New York Westchester     967315          7.6
# i 996 more rows
```

mutate

dplyr paketindeki **mutate()** fonksiyonu, bir veri çerçevesinde yeni sütunlar oluşturmak veya mevcut sütunları dönüştürmek için kullanılır. Bu fonksiyon, veri çerçevesindeki herhangi bir sütunu işleyerek yeni bilgiler eklemenize veya mevcut sütunları değiştirmenize olanak tanır. **mutate()** fonksiyonu, veri analizi sırasında verilerinizi özelleştirmek için oldukça kullanışlıdır.

```r
# işsiz nüfus sayısına ilişkin değişken üretme
counties_selected %>%
mutate(unemployed_population = population * unemployment / 100)
```

```
# A tibble: 3,138 x 5
   state   county   population unemployment unemployed_population
   <chr>   <chr>         <dbl>        <dbl>                 <dbl>
 1 Alabama Autauga       55221          7.6                 4197.
 2 Alabama Baldwin      195121          7.5                14634.
 3 Alabama Barbour       26932         17.6                 4740.
 4 Alabama Bibb          22604          8.3                 1876.
 5 Alabama Blount        57710          7.7                 4444.
 6 Alabama Bullock       10678         18                   1922.
 7 Alabama Butler        20354         10.9                 2219.
 8 Alabama Calhoun      116648         12.3                14348.
 9 Alabama Chambers      34079          8.9                 3033.
10 Alabama Cherokee      26008          7.9                 2055.
# i 3,128 more rows
```

```r
# yeni sütun ekle
counties_selected %>%
mutate(unemployed_population = population * unemployment / 100) %>%
arrange(desc(unemployed_population))
```

```
# A tibble: 3,138 x 5
   state      county         population unemployment unemployed_population
   <chr>      <chr>               <dbl>        <dbl>                 <dbl>
 1 California Los Angeles      10038388         10                1003839.
 2 Illinois   Cook              5236393         10.7                560294.
 3 Texas      Harris            4356362          7.5                326727.
 4 Arizona    Maricopa          4018143          7.7                309397.
 5 California Riverside         2298032         12.9                296446.
 6 California San Diego         3223096          8.7                280409.
 7 Michigan   Wayne             1778969         14.9                265066.
 8 California San Bernardino    2094769         12.6                263941.
 9 Florida    Miami-Dade        2639042         10                  263904.
10 New York   Kings             2595259         10                  259526.
# i 3,128 more rows
```

```r
# var olan sütunu güncelle
counties %>%
  select(state, county, population, men,women) %>%
mutate(population = men + women)
```

```
# A tibble: 3,138 x 5
   state   county   population   men women
   <chr>   <chr>         <dbl> <dbl> <dbl>
 1 Alabama Autauga       55221 26745 28476
 2 Alabama Baldwin      195121 95314 99807
 3 Alabama Barbour       26932 14497 12435
 4 Alabama Bibb          22604 12073 10531
 5 Alabama Blount        57710 28512 29198
 6 Alabama Bullock       10678  5660  5018
 7 Alabama Butler        20354  9502 10852
 8 Alabama Calhoun      116648 56274 60374
 9 Alabama Chambers      34079 16258 17821
10 Alabama Cherokee      26008 12975 13033
# i 3,128 more rows
```

```r
# birden fazla yeni değişken üretme
counties %>%
  select(state, county, population, men,women) %>%
mutate(men_ratio = men/population*100,
       women_ratio = women/population*100)
```

```
# A tibble: 3,138 x 7
   state   county   population   men women men_ratio women_ratio
   <chr>   <chr>         <dbl> <dbl> <dbl>     <dbl>       <dbl>
 1 Alabama Autauga       55221 26745 28476      48.4        51.6
 2 Alabama Baldwin      195121 95314 99807      48.8        51.2
 3 Alabama Barbour       26932 14497 12435      53.8        46.2
 4 Alabama Bibb          22604 12073 10531      53.4        46.6
 5 Alabama Blount        57710 28512 29198      49.4        50.6
 6 Alabama Bullock       10678  5660  5018      53.0        47.0
 7 Alabama Butler        20354  9502 10852      46.7        53.3
 8 Alabama Calhoun      116648 56274 60374      48.2        51.8
 9 Alabama Chambers      34079 16258 17821      47.7        52.3
10 Alabama Cherokee      26008 12975 13033      49.9        50.1
# i 3,128 more rows
```

```r
# transmute sadece yeni eklenen değişkenleri gösterir

counties %>%
  select(state, county, population, men,women) %>%
transmute(men_ratio = men/population*100,
```

```
        women_ratio = women/population*100)
```

```
# A tibble: 3,138 x 2
   men_ratio women_ratio
       <dbl>       <dbl>
 1      48.4        51.6
 2      48.8        51.2
 3      53.8        46.2
 4      53.4        46.6
 5      49.4        50.6
 6      53.0        47.0
 7      46.7        53.3
 8      48.2        51.8
 9      47.7        52.3
10      49.9        50.1
# i 3,128 more rows
```

```
# mutate_at ile koşula göre birden fazla değişkene aynı fonksiyon uygulanabilir.
scale2 <- function(x, na.rm = FALSE) (x - mean(x, na.rm = na.rm)) / sd(x, na.rm)

counties_selected %>%
  mutate_at(c("population","unemployment"),scale2)
```

```
# A tibble: 3,138 x 4
   state   county   population unemployment
   <chr>   <chr>         <dbl>        <dbl>
 1 Alabama Autauga     -0.141       -0.0563
 2 Alabama Baldwin      0.292       -0.0846
 3 Alabama Barbour     -0.228        2.78
 4 Alabama Bibb        -0.242        0.142
 5 Alabama Blount      -0.133       -0.0279
 6 Alabama Bullock     -0.278        2.89
 7 Alabama Butler      -0.249        0.880
 8 Alabama Calhoun      0.0495       1.28
 9 Alabama Chambers    -0.206        0.313
10 Alabama Cherokee    -0.231        0.0288
# i 3,128 more rows
```

```
  counties_selected %>% # birden fazla argüman kullanımı
    mutate_at(c("population","unemployment"),scale2,na.rm = TRUE)
```

```
# A tibble: 3,138 x 4
   state   county   population unemployment
   <chr>   <chr>         <dbl>        <dbl>
 1 Alabama Autauga     -0.141       -0.0563
 2 Alabama Baldwin      0.292       -0.0846
 3 Alabama Barbour     -0.228        2.78
 4 Alabama Bibb        -0.242        0.142
 5 Alabama Blount      -0.133       -0.0279
 6 Alabama Bullock     -0.278        2.89
 7 Alabama Butler      -0.249        0.880
 8 Alabama Calhoun      0.0495       1.28
 9 Alabama Chambers    -0.206        0.313
10 Alabama Cherokee    -0.231        0.0288
# i 3,128 more rows
```

```
  # mutate_if ile koşula göre birden fazla değişkende değişiklik yapılabilir.
  str(counties_selected)
```

```
spc_tbl_ [3,138 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ state       : chr [1:3138] "Alabama" "Alabama" "Alabama" "Alabama" ...
 $ county      : chr [1:3138] "Autauga" "Baldwin" "Barbour" "Bibb" ...
 $ population  : num [1:3138] 55221 195121 26932 22604 57710 ...
 $ unemployment: num [1:3138] 7.6 7.5 17.6 8.3 7.7 18 10.9 12.3 8.9 7.9 ...
```

```
  counties_selected <- counties_selected %>%
    mutate_if(is.character,as.factor)

  str(counties_selected)
```

```
spc_tbl_ [3,138 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ state       : Factor w/ 50 levels "Alabama","Alaska",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ county      : Factor w/ 1847 levels "Abbeville","Acadia",..: 82 89 100 149 164 225 235 240
 $ population  : num [1:3138] 55221 195121 26932 22604 57710 ...
 $ unemployment: num [1:3138] 7.6 7.5 17.6 8.3 7.7 18 10.9 12.3 8.9 7.9 ...
```

```r
counties_selected %>%
  mutate_if(is.numeric, scale2, na.rm = TRUE)
```

```
# A tibble: 3,138 x 4
   state   county   population unemployment
   <fct>   <fct>         <dbl>        <dbl>
 1 Alabama Autauga     -0.141       -0.0563
 2 Alabama Baldwin      0.292       -0.0846
 3 Alabama Barbour     -0.228        2.78
 4 Alabama Bibb        -0.242        0.142
 5 Alabama Blount      -0.133       -0.0279
 6 Alabama Bullock     -0.278        2.89
 7 Alabama Butler      -0.249        0.880
 8 Alabama Calhoun      0.0495       1.28
 9 Alabama Chambers    -0.206        0.313
10 Alabama Cherokee    -0.231        0.0288
# i 3,128 more rows
```

> **Dikkat**
>
> **mutate()** ve **transmute()** fonksiyonları, **dplyr** paketinde veri çerçevelerini işlerken kullanılan iki farklı fonksiyondur. Her ikisi de yeni sütunlar oluşturmanıza veya mevcut sütunları dönüştürmenize olanak tanır, ancak aralarındaki temel fark işlevlerinin dönüş değerleridir. Ancak kullanırken aşağıda belirtilen hususlara dikkat etmek gerekir:
>
> - **mutate()**, veri çerçevesine yeni sütunlar eklerken, orijinal veri çerçevesini değiştirmez. Yani, yeni sütunlar eklerken orijinal veri çerçevesinin boyutu artar. **mutate()** fonksiyonu, orijinal veri çerçevesini döndürürken eklenen sütunlarla birlikte veriyi içeren yeni bir veri çerçevesi döndürür.
> - **transmute()**, yeni sütunlar oluştururken orijinal veri çerçevesini değiştirmez. Ancak, **transmute()** fonksiyonu yalnızca belirtilen sütunları ve yeni sütunları içeren bir veri çerçevesi döndürür. Diğer orijinal sütunlar bu yeni veri çerçevesinde yer almaz. Bu, veri çerçevesini daha küçük ve özgünleştirilmiş bir hale getirir.

rename

rename() fonksiyonu, R programlama dilinde veri çerçevesi içindeki sütunların adlarını değiştirmek için kullanılır. Veri çerçevesi sütunlarının daha açıklayıcı veya kullanıcı dostu

adlara sahip olmasını sağlar. Bu, veri analizi ve raporlama süreçlerini daha anlaşılır ve düzenli hale getirmenize yardımcı olabilir.

```
# yeniden isimlendirmede eşitliği sol tarafı yeni isim olmalı
counties_selected %>%
rename(unemployment_rate = unemployment)
```

```
# A tibble: 3,138 x 4
   state   county   population unemployment_rate
   <fct>   <fct>         <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
```

```
# select ile beraber de yeniden isimlendirme yapılabilir
counties_selected %>%
select(state, county, population, unemployment_rate = unemployment)
```

```
# A tibble: 3,138 x 4
   state   county   population unemployment_rate
   <fct>   <fct>         <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
```

> Dikkat
>
> **rename** fonksiyonunda eşitliğin sol tarafına yeni isim, sağ tarafına ise önceki isim yazılır.

count

count() fonksiyonu, R programlama dilindeki **dplyr** paketinde bulunan ve belirli bir sütuna göre veri çerçevesindeki gözlemlerin sayısını hesaplamak için kullanılan bir fonksiyondur. Bu fonksiyon, veri çerçevesindeki belirli bir kategorik değişkenin benzersiz değerlerini ve her bir değer için kaç gözlemin olduğunu hesaplamak için oldukça kullanışlıdır.

count() fonksiyonu, veri analizi sürecinde veri özeti oluşturmak ve belirli bir değişkenin frekansını görmek için sıkça kullanılır. Ayrıca, veri çerçevesindeki her bir kategorik değeri ve bu değerlere ait gözlem sayılarını içeren yeni bir veri çerçevesi döndürür.

```
# count ile veri setinde sayma işlemleri yapılır
counties %>%
count()
```

```
# A tibble: 1 x 1
      n
  <int>
1  3138
```

```
# state dağılımını elde etmek
counties %>%
count(state)
```

```
# A tibble: 50 x 2
   state           n
   <chr>       <int>
 1 Alabama        67
 2 Alaska         28
 3 Arizona        15
 4 Arkansas       75
 5 California     58
 6 Colorado       64
 7 Connecticut     8
 8 Delaware        3
```

```
 9 Florida        67
10 Georgia       159
# i 40 more rows
```

```r
# sort = TRUE ile büyükten küçüge sıralama yapılabilir
counties %>%
count(state, sort = TRUE)
```

```
# A tibble: 50 x 2
   state              n
   <chr>          <int>
 1 Texas            253
 2 Georgia          159
 3 Virginia         133
 4 Kentucky         120
 5 Missouri         115
 6 Kansas           105
 7 Illinois         102
 8 North Carolina   100
 9 Iowa              99
10 Tennessee         95
# i 40 more rows
```

```r
# wt argümanı ile değişken toplamları hesaplanabilir
counties %>%
count(state, wt = population, sort = TRUE)
```

```
# A tibble: 50 x 2
   state                n
   <chr>            <dbl>
 1 California     38421464
 2 Texas          26538497
 3 New York       19673174
 4 Florida        19645772
 5 Illinois       12873761
 6 Pennsylvania   12779559
 7 Ohio           11575977
 8 Georgia        10006693
 9 Michigan        9900571
10 North Carolina  9845333
# i 40 more rows
```

group_by ve summarize

group_by() ve **summarize()** fonksiyonları, R programlama dilinde veri çerçevesi üzerinde gruplama ve özetleme işlemleri yapmak için kullanılan önemli **dplyr** fonksiyonlarıdır. Bu fonksiyonlar, veri analizi sürecinde verilerinizi daha iyi anlamak ve özetlemek için oldukça güçlü araçlardır.

group_by() fonksiyonu, veri çerçevesindeki verileri belirli bir sütuna veya birden fazla sütuna göre gruplamak için kullanılır. Bu gruplandırma işlemi, veriyi belirli bir kategoriye veya sınıfa göre ayırmak için kullanılır.

summarize() fonksiyonu, gruplanmış veri üzerinde istatistiksel veya özetleyici işlemler yapmak için kullanılır. Bu fonksiyon, belirli bir grup için özet bilgileri hesaplamak için kullanılır.

```
counties %>%
summarize(total_population = sum(population))
```

```
# A tibble: 1 x 1
  total_population
             <dbl>
1        315845353
```

```
counties %>%
summarize(total_population = sum(population),
average_unemployment = mean(unemployment))
```

```
# A tibble: 1 x 2
  total_population average_unemployment
             <dbl>                <dbl>
1        315845353                 7.80
```

```
# istenilen düzeye göre hesaplamalar group_by ile yapılır
counties %>%
group_by(state) %>%
summarize(total_pop = sum(population),
average_unemployment = sum(unemployment))
```

```
# A tibble: 50 x 3
   state       total_pop average_unemployment
   <chr>           <dbl>                <dbl>
```

```
 1 Alabama       4830620               758.
 2 Alaska         725461               257.
 3 Arizona       6641928               180.
 4 Arkansas      2958208               674.
 5 California   38421464               626.
 6 Colorado      5278906               477.
 7 Connecticut   3593222                65.3
 8 Delaware       926454                23.8
 9 Florida      19645772               696.
10 Georgia      10006693              1586.
# i 40 more rows
```

```
counties %>%
group_by(state) %>%
summarize(total_pop = sum(population),
average_unemployment = mean(unemployment)) %>%
arrange(desc(average_unemployment))
```

```
# A tibble: 50 x 3
   state          total_pop average_unemployment
   <chr>              <dbl>                <dbl>
 1 Mississippi      2988081                12.0
 2 Arizona          6641928                12.0
 3 South Carolina   4777576                11.3
 4 Alabama          4830620                11.3
 5 California      38421464                10.8
 6 Nevada           2798636                10.5
 7 North Carolina   9845333                10.5
 8 Florida         19645772                10.4
 9 Georgia         10006693                 9.97
10 Michigan         9900571                 9.96
# i 40 more rows
```

```
# birden fazla değişken düzeyinde gruplama
counties %>%
group_by(state, metro) %>%
summarize(total_pop = sum(population))
```

```
`summarise()` has grouped output by 'state'. You can override using the
`.groups` argument.
```

```
# A tibble: 97 x 3
# Groups:   state [50]
   state      metro    total_pop
   <chr>      <chr>        <dbl>
 1 Alabama    Metro      3671377
 2 Alabama    Nonmetro   1159243
 3 Alaska     Metro       494990
 4 Alaska     Nonmetro    230471
 5 Arizona    Metro      6295145
 6 Arizona    Nonmetro    346783
 7 Arkansas   Metro      1806867
 8 Arkansas   Nonmetro   1151341
 9 California Metro     37587429
10 California Nonmetro    834035
# i 87 more rows
```

```r
# elde edilen veri üzerinden devam edilecekse ungroup kullanılmalı.
# ungroup kullanılmazsa sonradan yapılan işlemler group_by değişkenleri düzeyinde
# devam eder

counties %>%
group_by(state, metro) %>%
summarize(total_pop = sum(population)) %>%
ungroup()
```

```
`summarise()` has grouped output by 'state'. You can override using the
`.groups` argument.
```

```
# A tibble: 97 x 3
   state      metro    total_pop
   <chr>      <chr>        <dbl>
 1 Alabama    Metro      3671377
 2 Alabama    Nonmetro   1159243
 3 Alaska     Metro       494990
 4 Alaska     Nonmetro    230471
 5 Arizona    Metro      6295145
 6 Arizona    Nonmetro    346783
 7 Arkansas   Metro      1806867
 8 Arkansas   Nonmetro   1151341
 9 California Metro     37587429
10 California Nonmetro    834035
# i 87 more rows
```

```r
# top_n en yüksek ya da en düşük sonuçları listeleme
counties_selected %>%
group_by(state) %>%
top_n(1, population) # her eyaletteki en yüksek nüfuslu yer
```

```
# A tibble: 50 x 4
# Groups:   state [50]
   state       county                 population unemployment
   <fct>       <fct>                       <dbl>        <dbl>
 1 Alabama     Jefferson                  659026          9.1
 2 Alaska      Anchorage Municipality     299107          6.7
 3 Arizona     Maricopa                  4018143          7.7
 4 Arkansas    Pulaski                    390463          7.5
 5 California  Los Angeles              10038388         10
 6 Colorado    El Paso                    655024          8.4
 7 Connecticut Fairfield                  939983          9
 8 Delaware    New Castle                 549643          7.4
 9 Florida     Miami-Dade                2639042         10
10 Georgia     Fulton                     983903          9.9
# i 40 more rows
```

```r
counties_selected %>%
group_by(state) %>%
top_n(-1, population) # her eyaletteki en düşük nüfuslu yer
```

```
# A tibble: 50 x 4
# Groups:   state [50]
   state       county                   population unemployment
   <fct>       <fct>                         <dbl>        <dbl>
 1 Alabama     Greene                         8697         20.4
 2 Alaska      Yakutat City and Borough        643          7.9
 3 Arizona     Greenlee                       9023         10
 4 Arkansas    Calhoun                        5245          7.2
 5 California  Alpine                         1131         10.7
 6 Colorado    San Juan                        606         13.8
 7 Connecticut Windham                      117470          9.3
 8 Delaware    Kent                         169509          8.4
 9 Florida     Liberty                        8295         10.2
10 Georgia     Taliaferro                     1721         12.1
# i 40 more rows
```

```
  counties_selected %>%
  group_by(state) %>%
  top_n(2, population) # her eyaletteki en yüksek nüfuslu 2 yer
```

```
# A tibble: 100 x 4
# Groups:   state [50]
   state      county                         population unemployment
   <fct>      <fct>                               <dbl>        <dbl>
 1 Alabama    Jefferson                          659026          9.1
 2 Alabama    Mobile                             414251          9.8
 3 Alaska     Anchorage Municipality             299107          6.7
 4 Alaska     Fairbanks North Star Borough        99705          7.9
 5 Arizona    Maricopa                          4018143          7.7
 6 Arizona    Pima                               998537         10
 7 Arkansas   Benton                             238198          4.2
 8 Arkansas   Pulaski                            390463          7.5
 9 California Los Angeles                       10038388         10
10 California San Diego                          3223096          8.7
# i 90 more rows
```

```
  # summarise_all bütün değişkenler için özetleme yapar
  counties_selected %>% summarise_all(nlevels)
```

```
# A tibble: 1 x 4
  state county population unemployment
  <int>  <int>      <int>        <int>
1    50   1847          0            0
```

```
  counties_selected %>%
    select(-county) %>%
    group_by(state) %>%
    summarise_all(mean)
```

```
# A tibble: 50 x 3
   state       population unemployment
   <fct>            <dbl>        <dbl>
 1 Alabama         72099.        11.3
 2 Alaska          25909.         9.19
 3 Arizona        442795.        12.0
```

```
 4 Arkansas        39443.         8.98
 5 California     662439.        10.8
 6 Colorado        82483.         7.46
 7 Connecticut    449153.         8.16
 8 Delaware       308818          7.93
 9 Florida        293220.        10.4
10 Georgia         62935.         9.97
# i 40 more rows
```

```
# summarise_at belli değişkenler için özetleme yapar
counties_selected %>%
  select(-county) %>%
  group_by(state) %>%
  summarise_at("population",mean)
```

```
# A tibble: 50 x 2
   state       population
   <fct>            <dbl>
 1 Alabama         72099.
 2 Alaska          25909.
 3 Arizona        442795.
 4 Arkansas        39443.
 5 California     662439.
 6 Colorado        82483.
 7 Connecticut    449153.
 8 Delaware       308818
 9 Florida        293220.
10 Georgia         62935.
# i 40 more rows
```

```
# summarise_if ile koşula göre özetleme yapar
counties_selected %>%
  summarize_if(is.numeric, mean, na.rm = TRUE)
```

```
# A tibble: 1 x 2
  population unemployment
       <dbl>        <dbl>
1    100652.         7.80
```

Dikkat

ungroup() fonksiyonu, **dplyr** paketinde kullanılan bir işlevdir ve bir veri çerçevesini veya gruplanmış bir veri çerçevesini gruplardan çıkarmak için kullanılır. **group_by()** fonksiyonu ile gruplanmış bir veri çerçevesini oluşturduğunuzda, veri çerçevesi belirli sütunlar üzerinde gruplama yapar ve her grup için ayrı işlemler yapmanıza olanak tanır. Ancak bazen gruplamadan çıkmak ve orijinal veri çerçevesini elde etmek isteyebilirsiniz.

```r
# Örnek bir veri çerçevesi oluşturalım
veri <- data.frame(
  Sehir = c("İstanbul", "Ankara", "İstanbul", "Ankara", "İzmir"),
  Cinsiyet = c("Erkek", "Kadın", "Erkek", "Kadın", "Erkek"),
  Yas = c(28, 32, 22, 24, 30),
  Puan = c(90, 85, 78, 92, 88)
)

# Şehir sütununa göre veriyi grupla
gruplu_veri <- group_by(veri, Sehir)
gruplu_veri |> summarise(mean(Puan))
```

```
# A tibble: 3 x 2
  Sehir    `mean(Puan)`
  <chr>           <dbl>
1 Ankara           88.5
2 İstanbul         84
3 İzmir            88
```

```r
# Grubu çıkarma
gruplu_veri <- ungroup(gruplu_veri)
gruplu_veri |> summarise(mean(Puan))
```

```
# A tibble: 1 x 1
  `mean(Puan)`
         <dbl>
1         86.6
```

Aynı veri setinde farklı sonuçlar elde edildiğine dikkat edelim. Eğer **group_by** ile oluşturulan veri setinde başka işlemler yapacaksanız öncesinde **ungroup()** yapmayı ihmal etmeyin.

> **Not**
>
> **group_by** sadece **summarize** fonksiyonu ile değil **mutate**, **transmute** gibi diğer fonksiyonlar ile birlikte de kullanılabilir.

case when

case_when() fonksiyonu, R programlama dilinde **dplyr** paketi içinde bulunan ve çoklu koşullara dayalı olarak yeni bir sütun oluşturmak veya mevcut bir sütunu dönüştürmek için kullanılan bir fonksiyondur. Bu fonksiyon, özellikle veri çerçevelerinde veya veri tablolarında, belirli koşullara dayalı olarak işlem yapmanız gerektiğinde oldukça kullanışlıdır. **case_when()** fonksiyonu, birden fazla koşulu kontrol ederek her bir koşula uygun bir değer veya işlem döndürmenizi sağlar. **case_when()** fonksiyonu, bir veya daha fazla koşul ifadesi ve bu koşullara karşılık gelecek değerler içeren çiftlerin bir listesini alır. Bu çiftler, **~** operatörü ile ayrılır.

```
# Örnek bir veri çerçevesi oluşturalım
veri <- data.frame(
  Ogrenci_Ad = c("Ali", "Esra", "Erkan", "Derya"),
  Puan = c(90, 75, 60, 80)
)

# Yeni bir sütun oluşturma: Puan kategorisi
veri <- veri %>%
  mutate(Puan_Kategorisi = case_when(
    Puan >= 90 ~ "AA",
    Puan >= 80 ~ "BA",
    Puan >= 70 ~ "BB",
    Puan >= 60 ~ "CB",
    TRUE ~ "FF"  # Tüm diğer durumlar için
  ))

print(veri)
```

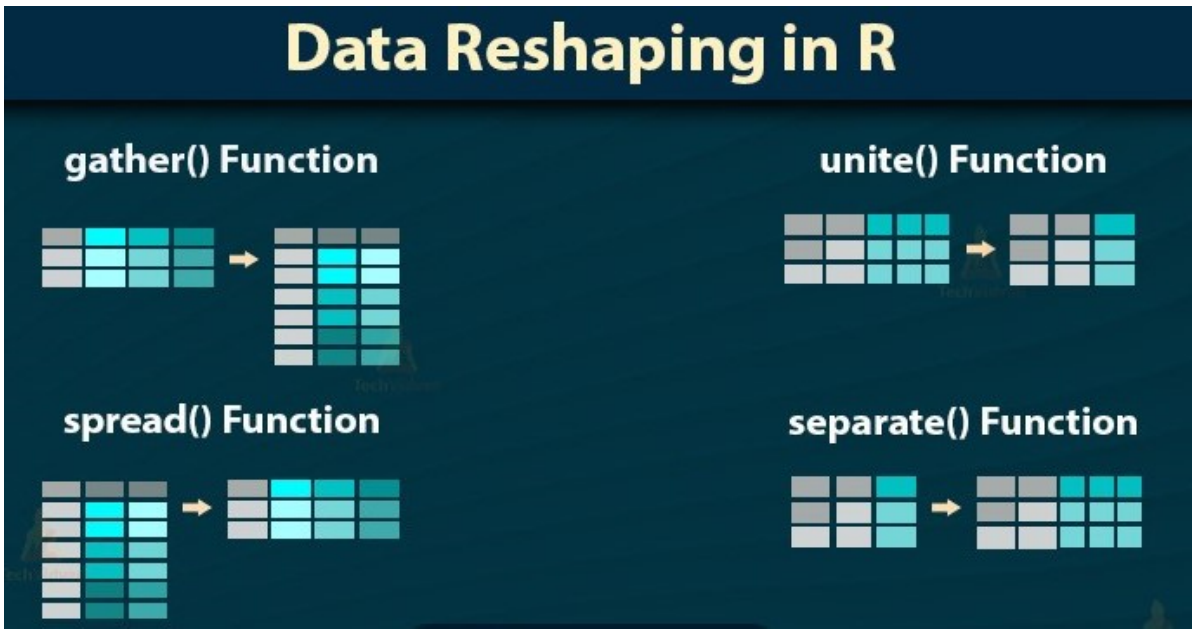
```
  Ogrenci_Ad Puan Puan_Kategorisi
1        Ali   90              AA
2       Esra   75              BB
3      Erkan   60              CB
4      Derya   80              BA
```

case_when() fonksiyonunu birden fazla koşul ile kullanabilirsiniz. Koşullar yukarıdan aşağıya sırayla kontrol edilir ve ilk koşulu sağlayan değer kullanılır.

```
veri <- veri %>%
  mutate(Not_Durumu = case_when(
    Puan >= 90 ~ "Geçti",
    Puan >= 60 & Puan < 70 ~ "Şartlı Geçti",
    Puan < 60 ~ "Kaldı",
    TRUE ~ "Bilinmiyor"  # Tüm diğer durumlar için
  ))

veri
```

```
  Ogrenci_Ad Puan Puan_Kategorisi   Not_Durumu
1        Ali   90               AA        Geçti
2       Esra   75               BB   Bilinmiyor
3      Erkan   60               CB Şartlı Geçti
4      Derya   80               BA   Bilinmiyor
```

reshaping



R programlama dilinde **tidyr** paketinin içinde bulunan **gather()**, **spread()**, **unite()**, ve **separate()** gibi fonksiyonlar, veri manipülasyonu ve veri dönüşümü işlemlerinde kullanılır.

Bu fonksiyonlar, veri çerçevesi içindeki verileri yeniden düzenlemek, sütunları birleştirmek veya bölmek, veriyi daha uygun bir yapıya getirmek için kullanılır.

gather() fonksiyonu, geniş formatlı (wide format) veriyi uzun formatlı (long format) bir yapıya dönüştürmek için kullanılır. Genellikle sütun adlarını bir "anahtar" sütununda toplamak için kullanılır.

```
library(tidyr)

veri <- data.frame(
  isim = c("Ali", "Esra"),
  Matematik = c(90, 85),
  Fizik = c(88, 76)
)

uzun_format_veri <- gather(veri, Ders, Not, -isim)

print(uzun_format_veri)
```

```
  isim      Ders Not
1  Ali Matematik  90
2 Esra Matematik  85
3  Ali     Fizik  88
4 Esra     Fizik  76
```

Bu kod, **Matematik** ve **Fizik** sütunlarını toplar ve bir "Ders" sütunu oluşturur.

spread() fonksiyonu, uzun formatlı veriyi geniş formatlı bir yapıya dönüştürmek için kullanılır. Genellikle "anahtar" sütunu içindeki değerleri sütun adları olarak kullanmak için kullanılır.

```
genis_format_veri <- spread(uzun_format_veri, Ders, Not)

print(genis_format_veri)
```

```
  isim Fizik Matematik
1  Ali    88        90
2 Esra    76        85
```

Bu kod, "Ders" sütunundaki değerleri sütun adlarına dönüştürür.

unite() fonksiyonu, iki veya daha fazla sütunu birleştirerek yeni bir sütun oluşturmak için kullanılır.

```r
veri <- data.frame(
  Ad = c("Ali", "Esra"),
  Soyad = c("Yılmaz", "Mutlu")
)

veri <- unite(veri, AdSoyad, Ad, Soyad, sep = " ")

print(veri)
```

```
    AdSoyad
1 Ali Yılmaz
2 Esra Mutlu
```

Bu kod, "Ad" ve "Soyad" sütunlarını birleştirerek "AdSoyad" sütununu oluşturur.

separate() fonksiyonu, bir sütunu belirli bir ayırıcı karakterle bölmek için kullanılır. Bölen değerleri yeni sütunlarda saklar.

```r
veri <- data.frame(
  AdSoyad = c("Ali Yılmaz", "Esra Mutlu")
)

veri <- separate(veri, AdSoyad, c("Ad", "Soyad"), sep = " ")

print(veri)
```

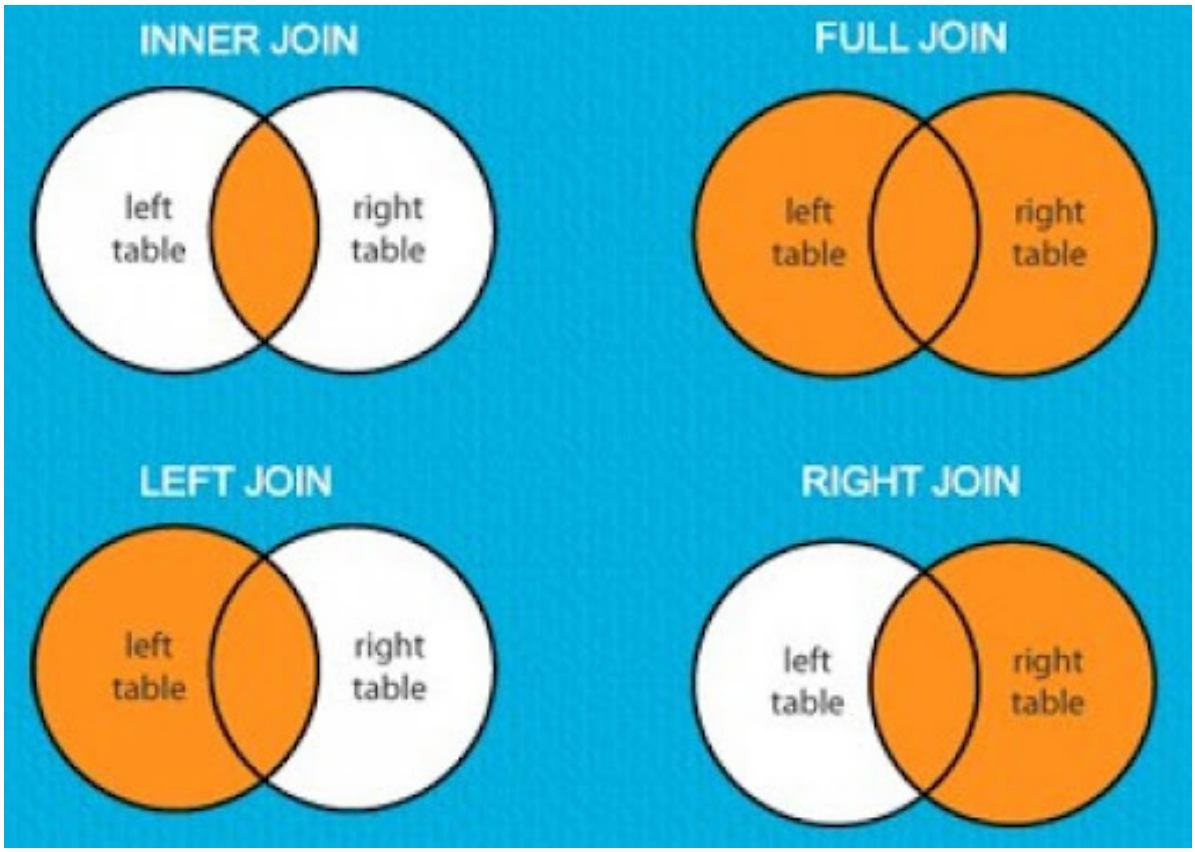
```
   Ad  Soyad
1  Ali Yılmaz
2 Esra  Mutlu
```

Bu kod, "AdSoyad" sütununu boşluk karakterine göre böler ve "Ad" ve "Soyad" sütunlarını oluşturur.

> **Tavsiye**
>
> Bu fonksiyonlar, veri çerçevesi içindeki verileri dönüştürmek ve düzenlemek için oldukça kullanışlıdır. Veri analizi sürecinde veri çerçevesini istediğiniz formata getirmek ve veriyi daha iyi anlamak için bu fonksiyonları kullanabilirsiniz. Daha ileri seviyede kullanmak için fonksiyonların argümanlarının nasıl kullanıldığını help kısmından ya da internetten araştırmanızı tavsiye ederim.
> Ayrıca **gather** ve **spread** fonksiyonları yerine bunların daha kullanışlı bir karşılığı olan

`pivot_longer` ve `pivot_wider` fonksiyonlarını da tercih edebilirsiniz. Bu konudaki örnekleri incelemenizi tavisye ederim. Belki bunları daha çok sevebilirsiniz.

join



R programlama dilinde `dplyr` paketinde bulunan `left_join()`, `right_join()`, `inner_join()`, ve `full_join()` gibi fonksiyonlar, veri çerçeveleri veya veri tabloları arasında birleştirme (join) işlemleri yapmak için kullanılır. Bu fonksiyonlar, farklı veri kaynaklarını birleştirmenizi veya ilişkilendirmenizi sağlar.

`left_join()` fonksiyonu, sol veri çerçevesi ile sağ veri çerçevesi arasında birleştirme işlemi yapar ve sol veri çerçevesindeki tüm gözlemleri korur. Eğer sağ veri çerçevesinde eşleşen gözlem yoksa, NA değerleri ile doldurulur.

```r
veri1 <- data.frame(
  Öğrenci = c("Ali", "Esra", "Osman"),
  Puan1 = c(90, 85, 78)
)

veri2 <- data.frame(
  Öğrenci = c("Ali", "Derya", "Merve"),
  Puan2 = c(88, 92, 85)
)

birlesik_veri <- left_join(veri1, veri2, by = "Öğrenci")

print(birlesik_veri)
```

```
  Öğrenci Puan1 Puan2
1     Ali    90    88
2    Esra    85    NA
3   Osman    78    NA
```

Bu kod, "Öğrenci" sütununa göre iki veri çerçevesini birleştirir. Sol veri çerçevesi (**veri1**) tüm gözlemleri içerir ve sağ veri çerçevesinde (**veri2**) eşleşen değerler varsa birleştirir.

right_join() fonksiyonu, **left_join()** ile benzerdir, ancak sağ veri çerçevesindeki tüm gözlemleri korur. Eğer sol veri çerçevesinde eşleşen gözlem yoksa, NA değerleri ile doldurulur.

```r
birlesik_veri <- right_join(veri1, veri2, by = "Öğrenci")

print(birlesik_veri)
```

```
  Öğrenci Puan1 Puan2
1     Ali    90    88
2   Derya    NA    92
3   Merve    NA    85
```

Bu kod, sağ veri çerçevesi (**veri2**) tüm gözlemleri içerir ve sol veri çerçevesinde (**veri1**) eşleşen değerler varsa birleştirir.

inner_join() fonksiyonu, sol ve sağ veri çerçeveleri arasında iç birleştirme yapar ve yalnızca ortak gözlemleri korur. Ortak gözlemleri içermeyen diğer gözlemleri atar.

```r
birlesik_veri <- inner_join(veri1, veri2, by = "Öğrenci")

print(birlesik_veri)
```

```
  Öğrenci Puan1 Puan2
1     Ali    90    88
```

Bu kod, sadece sol ve sağ veri çerçevelerinde (**veri1** ve **veri2**) ortak olan gözlemleri korur.

full_join() fonksiyonu, sol ve sağ veri çerçeveleri arasında tam birleştirme yapar ve tüm gözlemleri korur. Ortak olmayan değerler NA ile doldurulur.

```r
birlesik_veri <- full_join(veri1, veri2, by = "Öğrenci")

print(birlesik_veri)
```

```
  Öğrenci Puan1 Puan2
1     Ali    90    88
2    Esra    85    NA
3   Osman    78    NA
4   Derya    NA    92
5   Merve    NA    85
```

Bu kod, sol ve sağ veri çerçevelerini (**veri1** ve **veri2**) tamamen birleştirir ve tüm gözlemleri içerir.

> **Not**
>
> Bu dört join fonksiyonu, farklı veri kaynaklarını birleştirme işlemlerinde kullanılır ve veri analizi sürecinde verileri daha kapsamlı bir şekilde incelemek için oldukça kullanışlıdır. Hangi join işleminin kullanılacağı, veri yapısına ve ihtiyaca bağlı olarak değişebilir.

Keşifçi Veri Analizi

Keşifçi Veri Analizi (Exploratory Data Analysis veya kısaca EDA), veri setinizi anlamak, içindeki örüntüleri ve ilişkileri belirlemek ve olası sorunları tanımlamak amacıyla veriye yakından bakmanızı sağlayan bir veri analizi yaklaşımıdır. EDA, verileri tanımanıza veya verilerdeki olası özellikler ve ilişkiler hakkında daha derin bir anlayış kazanmanıza yardımcı olabilir. EDA, yeni bir şey değildir, ancak EDA, birkaç nedenden dolayı yakın geçmişte önemli ölçüde büyümüştür:

- Veriler her zamankinden daha hızlı ve daha büyük miktarlarda üretiliyor, bu yüzden incelememiz gereken çok şey var.
- Bilgisayarlar ve yazılımlar (R gibi) EDA yapma fırsatlarını genişletmiştir.
- İstatistiksel model seçeneklerindeki artış, genellikle doğrudan geleneksel bir modele gitmek yerine verilerimize daha yakından bakmamızı gerektirmektedir.

EDA, verilerinizin nihai analizi açısından genellikle istatistiksel değildir, ancak EDA'nın geçiş süreci olarak düşünülmesi gerekir. EDA'dan öğrendikleriniz modellemenize rehberlik edecek ve istatistiksel araçlar hakkında verdiğiniz kararları doğrudan bilgilendirecektir. R gibi programlama dilleri ve istatistiksel araçlar, EDA sürecini kolaylaştırmak ve verileri görselleştirmek için kullanışlıdır. EDA, veri madenciliği ve veri bilimi projelerinin başlangıcında sıklıkla kullanılır ve aşağıdaki adımları içerir:

1. **Veri İçe Aktarma:** İlk adım, analiz yapmak için veriyi içe aktarmaktır. Veriyi R ortamına çeşitli formatlardan (CSV, Excel, SQL veritabanları, vb.) içe aktarabilirsiniz.
2. **Veriye Genel Bakış:** Veri setinize ilk bakışta, kaç gözlem ve değişken olduğunu, değişken türlerini (sayısal, kategorik, metinsel vb.) ve eksik verilerin varlığını incelemelisiniz. Bu bilgi, veri hakkında ilk fikirlerinizi oluşturmanıza yardımcı olur.
3. **Veri Görselleştirme:** Verileri görselleştirmek, EDA'nın önemli bir parçasıdır. R'nin ggplot2 gibi kütüphaneleri, verilerinizi grafiklerle görselleştirmek için kullanışlı araçlar sunar. Histogramlar, kutu grafikleri, çubuk grafikleri ve dağılım grafikleri gibi grafikler oluşturarak verilerinizi daha iyi anlayabilirsiniz.
4. **Merkezi Eğilim ve Dağılım Ölçüleri:** Veri setinizin merkezi eğilimini (ortalama, medyan, mod) ve dağılımını (standart sapma, varyans, çeyrekler arası aralık) hesaplayarak verilerinizin genel özelliklerini değerlendirebilirsiniz.

5. **Değişkenler Arası İlişkiler:** Değişkenler arasındaki ilişkileri anlamak için korelasyon analizi, scatter plotlar ve faktör analizi gibi teknikleri kullanabilirsiniz.
6. **Aykırı Değerler ve Eksik Veriler:** Aykırı değerleri tanımlayın ve bunların analiz üzerindeki etkilerini değerlendirin. Ayrıca eksik verileri ele alın (örneğin, eksik verileri doldurma veya eksik gözlemleri çıkarma).
7. **Veri Gruplama ve Alt Kümelere Bölme:** İhtiyaca göre veriyi gruplara ayırabilir veya alt kümeler oluşturabilirsiniz. Bu, farklı veri alt kümeleri arasındaki farkları incelemek için kullanışlı olabilir.
8. **Hipotez Testleri ve İstatistiksel Analiz:** EDA süreci sırasında, veriler üzerinde belirli hipotezleri test etmek için istatistiksel testler (t-test, ANOVA, vb.) uygulayabilirsiniz. Bu, verilerinizde anlamlı farklılıkları veya özellikleri tespit etmenize yardımcı olur.
9. **Sonuçların Yorumlanması:** EDA sürecinin sonunda, elde edilen sonuçları yorumlamalı ve bulgularınızı raporlamalısınız. Bulgularınız, daha sonraki analiz aşamaları veya veri madenciliği projeleri için temel oluşturur.

EDA, veri analizi sürecinin önemli bir parçasıdır çünkü veriyi daha iyi anlamanızı ve daha ileri analizler için yol haritasını belirlemenizi sağlar. Aynı zamanda veri setinizdeki hataları veya tutarsızlıkları tespit etmenize ve düzeltmenize de yardımcı olur.

Veri ile Tanışma

Veri analizinin başlangıç aşamasında, verinin yapısına, ne tür değişkenler içerdiğine, çeşitli özet istatistiklerine bakmak ve gerekli ise ne tür dönüşümler yapmak gerektiğini bilmek önemlidir. Bu süreçler daha derin analizlere daha kolay devam edebilmek için de önemlidir. Bunları gerçekleştirmek için hem özet tablolar hem de grafikler yardımıyla verileri tanımak gerekmektedir.

Tek ve iki değişkenli olarak sayısal ve kategorik veri analizi **mpg** verisi kullanılarak yapılacaktır. Bu veri setinde 38 farklı aracın yakıt verileri bulunmaktadır.

```
# mpg verisi ggplot2 paketinde olduğundan paketi çağırıyoruz
library(ggplot2)

head(mpg)
```

```
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
```

```
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

```
nrow(mpg)
```

```
[1] 234
```

```
ncol(mpg)
```

```
[1] 11
```

```
str(mpg)
```

```
tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
 $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
 $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
 $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv         : chr [1:234] "f" "f" "f" "f" ...
 $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : chr [1:234] "p" "p" "p" "p" ...
 $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

```
colnames(mpg)
```

```
 [1] "manufacturer" "model"        "displ"        "year"         "cyl"
 [6] "trans"        "drv"          "cty"          "hwy"          "fl"
[11] "class"
```

```
summary(mpg)
```

```
 manufacturer          model               displ            year
 Length:234         Length:234         Min.   :1.600   Min.   :1999
 Class :character   Class :character   1st Qu.:2.400   1st Qu.:1999
 Mode  :character   Mode  :character   Median :3.300   Median :2004
                                       Mean   :3.472   Mean   :2004
                                       3rd Qu.:4.600   3rd Qu.:2008
                                       Max.   :7.000   Max.   :2008
      cyl           trans               drv                 cty
 Min.   :4.000   Length:234         Length:234         Min.   : 9.00
 1st Qu.:4.000   Class :character   Class :character   1st Qu.:14.00
 Median :6.000   Mode  :character   Mode  :character   Median :17.00
 Mean   :5.889                                         Mean   :16.86
 3rd Qu.:8.000                                         3rd Qu.:19.00
 Max.   :8.000                                         Max.   :35.00
      hwy             fl                class
 Min.   :12.00   Length:234         Length:234
 1st Qu.:18.00   Class :character   Class :character
 Median :24.00   Mode  :character   Mode  :character
 Mean   :23.44
 3rd Qu.:27.00
 Max.   :44.00
```

```
df <- mpg

# class değişkenini faktöre çevirip, kategorilerine bakalım
df$class <- factor(df$class)
levels(df$class)
```

```
[1] "2seater"    "compact"    "midsize"    "minivan"    "pickup"
[6] "subcompact" "suv"
```

```
dplyr::glimpse(df)
```

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
```

```
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
$ class        <fct> compact, compact, compact, compact, compact, compact, com~
```

Tavsiye

Veri analizi için **skimr** paketi de kullanılabilir. **skimr**, R programlama dilinde veri setlerinin hızlı bir şekilde özetlenmesini sağlayan bir pakettir. Veri setlerinin yapısını, özelliklerini ve bazı istatistiksel özetlerini görsel ve açıklayıcı bir şekilde sunar. Bu paket, veri keşfi aşamasında veri setinin genel özelliklerini anlamak için kullanılır.
skimr paketi, veri setinizdeki değişkenlerin türlerine göre istatistiksel özetler sunar. Örneğin, sayısal değişkenler için merkezi eğilim ölçüleri (ortalama, medyan), dağılım (standart sapma, min-max değerleri), faktör değişkenleri için sınıf sayısı, en sık rastlanan sınıf ve eksik veri durumları gibi bilgileri sunar.
Bu paket, veri setinin yapısını hızlıca anlamak ve önemli özelliklerini keşfetmek için kullanılır. Özellikle veri setlerinin keşfedilmesi, temizlenmesi ve analiz edilmesi aşamalarında oldukça faydalıdır. Bu, veri analiz sürecinde veriye daha derinlemesine bakmayı ve hangi analiz tekniklerinin kullanılacağına dair daha iyi bir anlayış geliştirmeyi sağlar.

Sürekli Değişkenler

Veri analizi, birçok farklı değişken türünün incelenmesini gerektirir. Bu değişkenler arasında sürekli değişkenler özellikle önemlidir. Sürekli değişkenler, belirli bir aralıktaki değerleri alabilen ve sonsuz sayıda mümkün değer içeren değişkenlerdir. Örnek olarak, yaş, gelir, sıcaklık gibi değerler sürekli değişkenlere örnektir. Sürekli değişkenlerin analizi, verileri anlamak ve içindeki örüntüleri keşfetmek için kullanılır. Bu analiz, genellikle aşağıdaki adımları içerir:

1. **Veri Görselleştirme:**Sürekli değişkenlerin analizine başlamak için verilerinizi görselleştirmek önemlidir. Histogramlar, kutu grafikleri, yoğunluk grafikleri ve saçılım grafikleri gibi grafikler, veri dağılımını ve örüntülerini görsel olarak incelemenize yardımcı olur. Bu grafikler, veri setinizin merkezi eğilimini (ortalama veya medyan), yayılımını ve aykırı değerleri hızla görmeye yardımcı olur.
2. **Merkezi Eğilim ve Dağılım Ölçüleri:** Sürekli değişkenlerin merkezi eğilimini ve dağılımını hesaplamak verileri özetlemenin önemli bir yoludur. Bu ölçümler, veri setinin

merkezi noktasını ve veri noktalarının nasıl dağıldığını anlamamıza yardımcı olur. Örnek olarak, ortalama (mean), medyan (median), standart sapma (standard deviation) ve varyans (variance) gibi ölçümler bu aşamada kullanılır.

3. **Korelasyon Analizi:** Eğer birden fazla sürekli değişken arasındaki ilişkiyi anlamak istiyorsanız, korelasyon analizi yapabilirsiniz. Korelasyon, iki değişken arasındaki ilişkinin gücünü ve yönünü ölçer. Korelasyon katsayısı, bu ilişkiyi değerlendirmek için kullanılır. Pozitif bir korelasyon, iki değişkenin aynı yönde değiştiğini, negatif bir korelasyon ise iki değişkenin ters yönde değiştiğini gösterir.

4. **Hipotez Testleri:** Sürekli değişkenler arasındaki farklılıkları değerlendirmek için hipotez testleri kullanılabilir. Örneğin, iki grup arasındaki ortalama değerlerin istatistiksel olarak anlamlı bir farklılık gösterip göstermediğini belirlemek için t-testleri veya ANOVA analizi kullanılabilir.

5. **Güven Aralıkları:** Sürekli değişkenlerin analizi sırasında, belirli bir parametre (örneğin, ortalama) hakkında güven aralıkları hesaplanabilir. Bu güven aralıkları, parametrenin belirli bir güven düzeyinde bulunduğu aralığı gösterir. Bu, parametrenin tahmini kesinliğini değerlendirmek için kullanışlıdır.

Sürekli değişkenlerin analizi, verileri anlama ve kararlarınızı destekleme sürecinin önemli bir parçasıdır. İyi bir analiz, veri setinizdeki örüntüleri ve ilişkileri açığa çıkarmanıza yardımcı olur ve bilinçli kararlar almanıza yardımcı olur. Bu nedenle, sürekli değişkenlerin analizi yaparken yukarıda belirtilen adımları takip etmek önemlidir.

```
# cty ve hwy değişkenlerini inceleyelim.
# cty şehiriçi, hwy şehirarasını ifade ediyor.

summary(df$cty)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   14.00   17.00   16.86   19.00   35.00
```

```
var(df$cty)
```

```
[1] 18.11307
```

```
mean(df$cty)
```

```
[1] 16.85897
```

```r
summary(df$hwy)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.00   18.00   24.00   23.44   27.00   44.00
```

```r
var(df$hwy)
```

```
[1] 35.45778
```

```r
mean(df$hwy)
```

```
[1] 23.44017
```

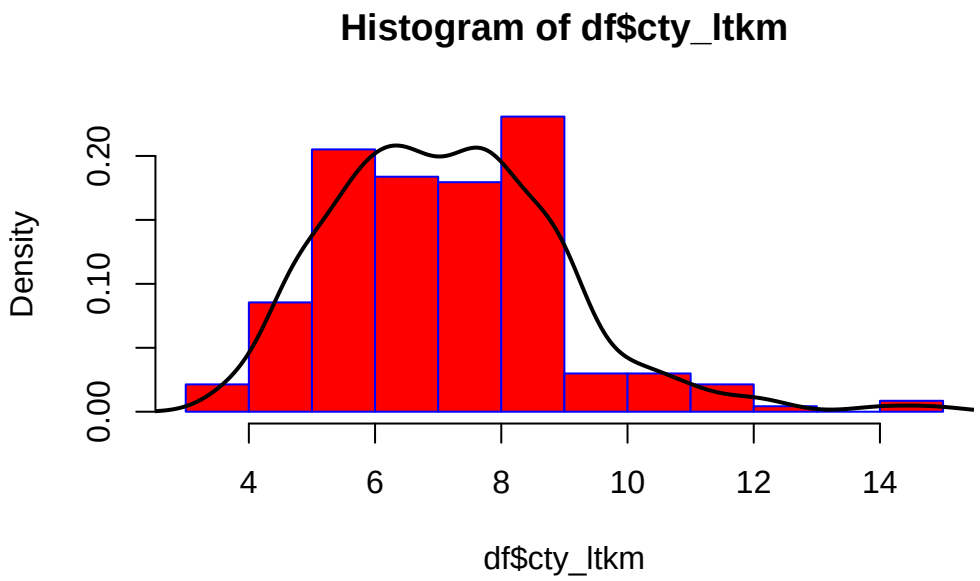
```r
# 1 mile= 1.609 km
# 1 galon = 3.79 lt

# litre başına km hesaplama
galonmil_to_ltkm <- function(x){

  km <- x * 1.609/3.79
  return(km)
}

df$cty_ltkm <- galonmil_to_ltkm(df$cty)
df$hwy_ltkm <- galonmil_to_ltkm(df$hwy)
quantile(df$cty_ltkm)
```

```
      0%       25%       50%       75%      100%
3.820844  5.943536  7.217150  8.066227 14.858839
```

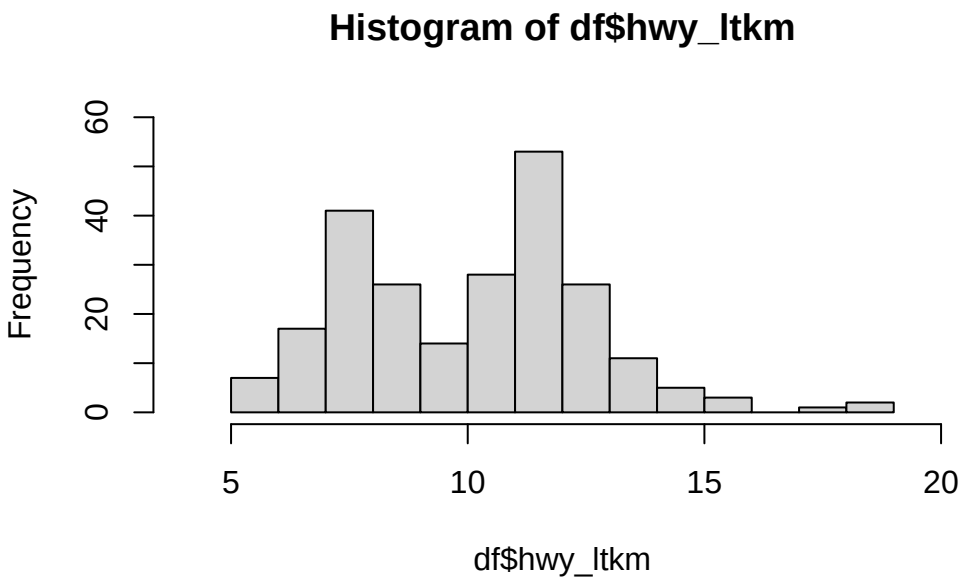
```r
# şehiriçi araçların % 75'i 1 lt ile 8.06 km den az yol alıyor.
quantile(df$hwy_ltkm)
```

```
      0%       25%       50%       75%      100%
5.094459  7.641689 10.188918 11.462533 18.679683
```

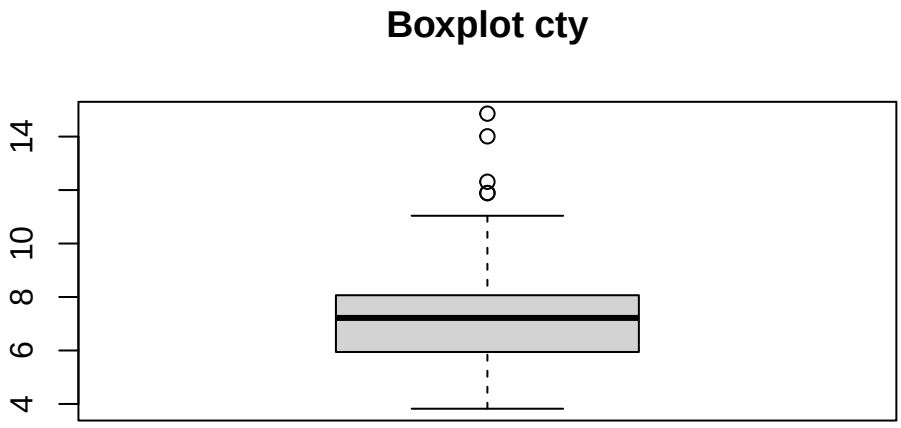
```r
# şehirlerarası araçların % 75'i 1 lt ile 11.46 km den az yol alıyor.


# değişken dağılımı için histogram grafiği kullanılabilir.
hist(df$cty_ltkm,freq = FALSE,col = "red",border = "blue")
lines(density(df$cty_ltkm), col = "black", lwd = 2,)
```




```r
hist(df$hwy_ltkm,xlim = c(4,20), ylim = c(0,60), breaks = 10)
```



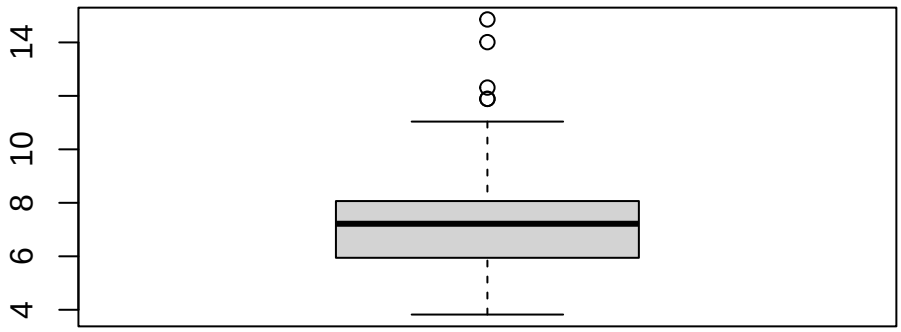

```r
# Boxplot
boxplot(df$cty_ltkm, main = "Boxplot cty")
```



```
fivenum(df$cty_ltkm) # minimum, Q1, median, Q3, maximum
```

```
[1]  3.820844  5.943536  7.217150  8.066227 14.858839
```

```
# outliers
boxplot(df$cty_ltkm)$out
```

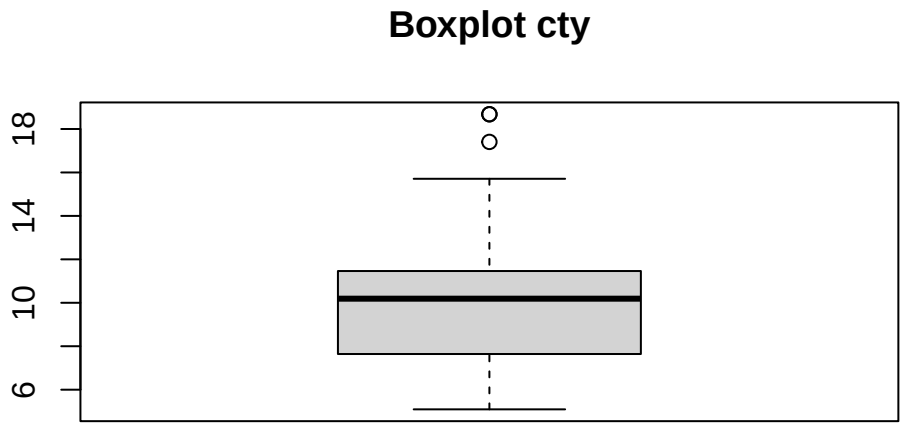



```
[1] 11.88707 11.88707 14.00976 14.85884 12.31161
```

```
# outliers hangi sıralarda
which(df$cty_ltkm %in% boxplot(df$cty_ltkm)$out)
```

```
[1] 100 197 213 222 223
```

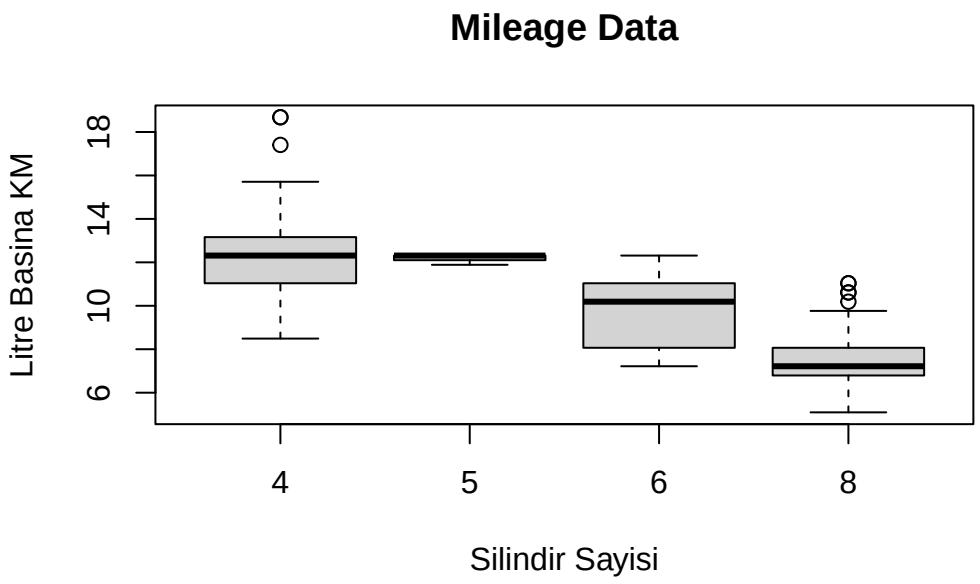
```
boxplot(df$hwy_ltkm, main = "Boxplot cty")
```




```
fivenum(df$hwy_ltkm) # minimum, Q1, median, Q3, maximum
```

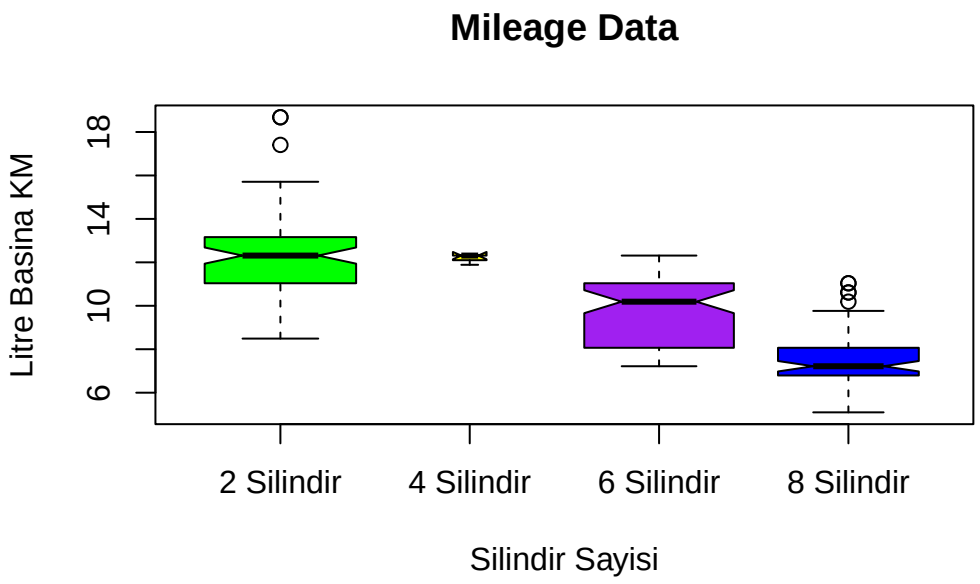
```
[1]  5.094459  7.641689 10.188918 11.462533 18.679683
```

```
boxplot(hwy_ltkm ~ cyl, data = df, xlab = "Silindir Sayısı",
   ylab = "Litre Başına KM", main = "Mileage Data")
```




```
boxplot(hwy_ltkm ~ cyl, data = df,
   xlab = "Silindir Sayısı",
   ylab = "Litre Başına KM",
   main = "Mileage Data",
   notch = TRUE,
   varwidth = TRUE,
   col = c("green","yellow","purple","blue"),
   names = c("2 Silindir","4 Silindir","6 Silindir","8 Silindir")
)
```

```
Warning in (function (z, notch = FALSE, width = NULL, varwidth = FALSE, : some
notches went outside hinges ('box'): maybe set notch=FALSE
```




```
# Sürekli iki değişken incelemek istersek;

# displ ve cty_ltkm değişkenlerini inceleyelim
# displ motor hacmini ifade ediyor

summary(df$displ)
```

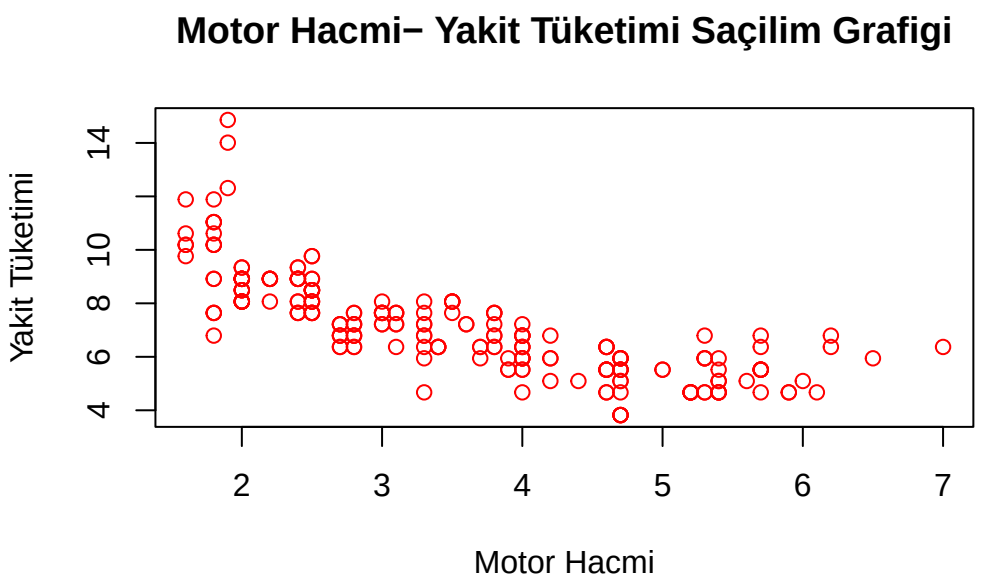
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.600   2.400   3.300   3.472   4.600   7.000
```

```
with(df,cor(displ,cty_ltkm))
```

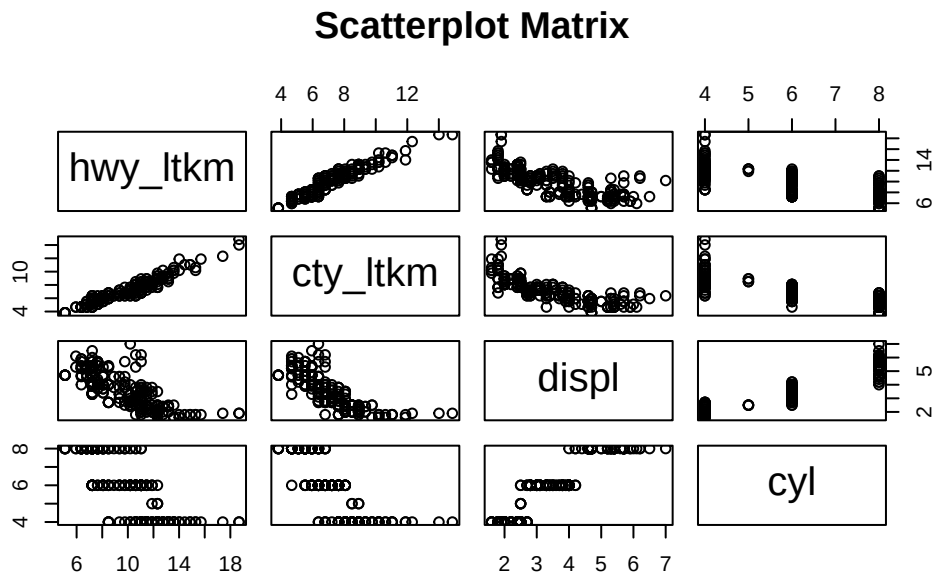
```
[1] -0.798524
```

```
# motor hacmi ile lt başına km ters ilişkili

plot(df$displ,df$cty_ltkm,
     main = "Motor Hacmi- Yakıt Tüketimi Saçılım Grafiği",
     col="red",
     xlab = "Motor Hacmi",
```

```
      ylab = "Yakıt Tüketimi")
```




```
# birden fazla değişkenin saçılım grafiği
pairs(~hwy_ltkm+cty_ltkm+displ+cyl,data = df,main = "Scatterplot Matrix")
```



Kategorik Değişkenler

Veri analizi sürecinde, kategorik değişkenler (veya gruplar) genellikle çok önemli bir rol oynar. Kategorik değişkenler, belirli bir sınıfı veya kategoriyi temsil eden değişkenlerdir ve tipik olarak metin veya sembollerle ifade edilirler. Örnek olarak, cinsiyet, eğitim seviyesi, ürün kategorileri gibi değişkenler kategorik değişkenlere örnektir. Kategorik değişkenlerin analizi, bu değişkenlerin içindeki örüntüleri, dağılımları ve ilişkileri anlamamıza yardımcı olur. Aşağıda, kategorik değişkenlerin analizi için izlenebilecek temel adımları bulabilirsiniz:

1. **Frekans Tabloları ve Görselleştirme:** Kategorik değişkenlerin frekans tablolarını ve grafiklerini oluşturarak, her kategori veya sınıfın veri setinde ne kadar sık görüldüğünü anlayabilirsiniz. Örneğin, bar grafikleri, pasta grafikleri veya çubuk grafikleri kullanarak kategori frekanslarını görselleştirebilirsiniz. `summary()` ve `table()` gibi R fonksiyonları ile bu verileri inceleyebilirsiniz.

2. **İlişkileri İnceleme:** Kategorik değişkenler arasındaki ilişkileri anlamak önemlidir. İki kategorik değişken arasındaki ilişkiyi değerlendirmek için çapraz tablolar (cross-tabulation) ve ki-kare (chi-squared) istatistiksel testleri kullanabilirsiniz. Bu testler, iki değişken arasındaki bağımlılığı değerlendirmek için kullanılır.

3. **İstatistiksel Testler:** Kategorik değişkenlerin analizi sırasında, gruplar arasındaki farkları değerlendirmek için hipotez testleri kullanabilirsiniz. İki kategorik değişken arasındaki ilişkinin istatistiksel olarak anlamlı olup olmadığını belirlemek için ki-kare testi veya

Fisher'in kesin testi gibi testler kullanabilirsiniz. Ayrıca ANOVA gibi testler, bir kategorik değişkenin birden fazla grup üzerindeki etkisini değerlendirmek için kullanılabilir.

4. **Veri Görselleştirme:** Kategorik değişkenlerin analizinde, gruplar arasındaki farkları daha iyi anlamak için grafikler kullanabilirsiniz. Bar grafikleri, grupların frekanslarını görselleştirmek için sıklıkla kullanılırken, gruplar arasındaki ilişkiyi anlamak için mozaik grafikleri veya heatmap'leri de kullanabilirsiniz.

Kategorik değişkenlerin analizi, veri setinizin içindeki desenleri ve ilişkileri anlamanıza yardımcı olur. Bu analiz, kararlarınızı desteklemek ve veriyi daha iyi anlamak için önemlidir. R programlama dili, kategorik değişkenlerin analizi için bir dizi kullanışlı fonksiyon ve paket sunar. Bu adımları takip ederek, veri analiz projelerinizde kategorik değişkenleri etkili bir şekilde analiz edebilirsiniz.

```
# class ve trans değişkenlerine bakalım
# class araç sınıfı, trans ise vites türünü ifade ediyor.

summary(df$class)
```

```
   2seater    compact    midsize    minivan     pickup subcompact        suv
         5         47         41         11         33         35         62
```

```
table(df$class)
```

```
   2seater    compact    midsize    minivan     pickup subcompact        suv
         5         47         41         11         33         35         62
```

```
xtabs(~class,data=df)
```

```
class
   2seater    compact    midsize    minivan     pickup subcompact        suv
         5         47         41         11         33         35         62
```

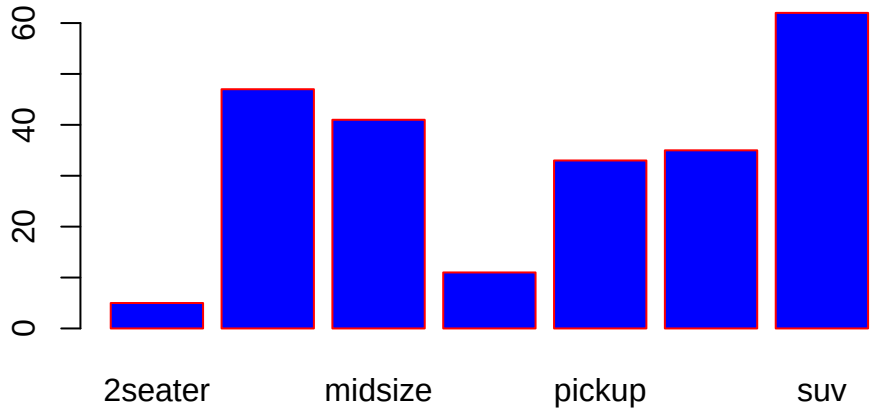
```
table(df$trans)
```

```
  auto(av)   auto(l3)   auto(l4)   auto(l5)   auto(l6)   auto(s4)   auto(s5)
         5          2         83         39          6          3          3
  auto(s6) manual(m5) manual(m6)
        16         58         19
```

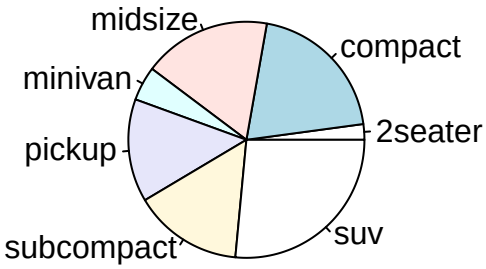
```r
prop.table(table(df$class))
```

```
   2seater    compact    midsize    minivan     pickup subcompact        suv
0.02136752 0.20085470 0.17521368 0.04700855 0.14102564 0.14957265 0.26495726
```

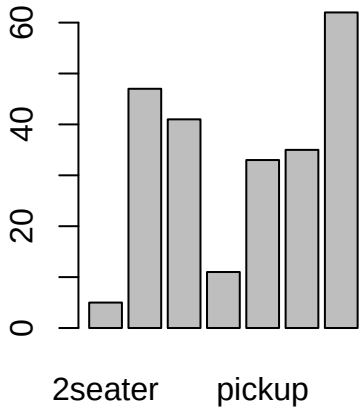
```r
tab <- table(df$class)
barplot(tab,col="blue",border="red")
```

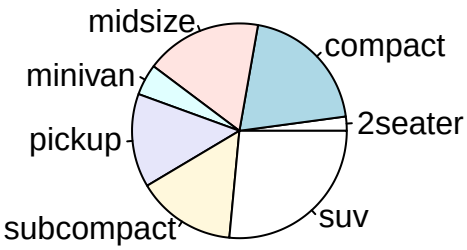



```r
pie(tab)
```




```
par(mfrow = c(1, 2))
barplot(tab)
pie(tab)
```






```r
# Kategorik iki değişken incelemek istersek;

xtabs(~trans+class,data=df)
```

```
           class
trans       2seater compact midsize minivan pickup subcompact suv
  auto(av)        0       2       3       0      0          0   0
  auto(l3)        0       1       0       1      0          0   0
  auto(l4)        1       8      14       8     12         11  29
  auto(l5)        0       4       5       0      8          4  18
  auto(l6)        0       0       0       2      0          0   4
  auto(s4)        0       2       1       0      0          0   0
  auto(s5)        0       2       0       0      0          0   1
  auto(s6)        1       5       6       0      0          1   3
  manual(m5)      0      18       9       0      8         16   7
  manual(m6)      3       5       3       0      5          3   0
```

```r
prop.table(table(df$year,df$class),1) # satır toplamları 1' eşittir
```

```
        2seater    compact    midsize    minivan     pickup subcompact
1999 0.01709402 0.21367521 0.17094017 0.05128205 0.13675214 0.16239316
2008 0.02564103 0.18803419 0.17948718 0.04273504 0.14529915 0.13675214

            suv
1999 0.24786325
2008 0.28205128
```

```r
prop.table(table(df$year,df$class),2) # sütun toplamları 1' eşittir
```

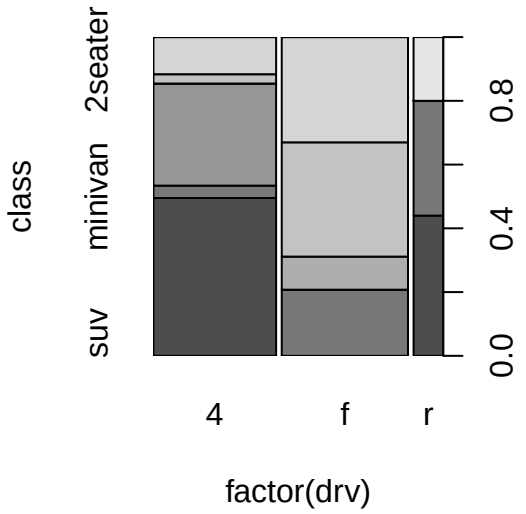
```
       2seater   compact   midsize   minivan    pickup subcompact       suv
1999 0.4000000 0.5319149 0.4878049 0.5454545 0.4848485  0.5428571 0.4677419
2008 0.6000000 0.4680851 0.5121951 0.4545455 0.5151515  0.4571429 0.5322581
```

```r
proportions(xtabs(~ manufacturer + year, data = df), 1)
```

```
            year
manufacturer      1999      2008
  audi       0.5000000 0.5000000
  chevrolet  0.3684211 0.6315789
  dodge      0.4324324 0.5675676
  ford       0.6000000 0.4000000
  honda      0.5555556 0.4444444
  hyundai    0.4285714 0.5714286
  jeep       0.2500000 0.7500000
  land rover 0.5000000 0.5000000
  lincoln    0.6666667 0.3333333
  mercury    0.5000000 0.5000000
  nissan     0.4615385 0.5384615
  pontiac    0.6000000 0.4000000
  subaru     0.4285714 0.5714286
  toyota     0.5882353 0.4117647
  volkswagen 0.5925926 0.4074074
```

```r
# araç sınıfı ile drv değişkenine birlikte bakalım
# f = front-wheel drive (önden çekiş),
# r = rear wheel drive (arkadan çekiş),
# 4 = 4wd (4 çeker)
```

```
plot(class ~ factor(drv), data = df)
```



Eğer hem sürekli hem de kategorik değişkenleri incelemek istersek, benzer şekilde görselleştirme ve kategoriler arasında merkezi eğilim ölçüleri hesaplanabilir. Bunlar dışında uygun istatistiksel testler de gerçekleştirilebilir.

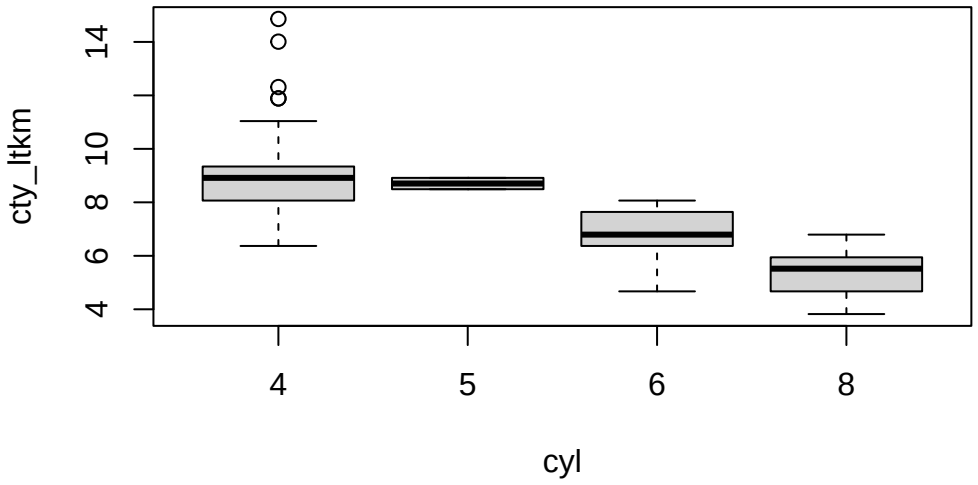
```
# Silindir düzeyinde yakıt tüketimi
tapply(df$cty_ltkm, df$cyl, mean)
```

```
       4        5        6        8
8.920545 8.703034 6.883968 5.337052
```

```
# Same using aggregate()
aggregate(cty_ltkm ~ cyl, data = df, FUN = mean)
```

```
  cyl cty_ltkm
1   4 8.920545
2   5 8.703034
3   6 6.883968
4   8 5.337052
```

```r
boxplot(cty_ltkm ~ cyl, data = df)
```



Zaman Serileri

R programlama dili, zaman serileri analizi için kapsamlı bir dizi fonksiyon ve paket sunar. Zaman serileri analizi, zaman içindeki veri noktalarının örüntülerini ve trendlerini incelemeyi amaçlar. R'de zaman serileri ile çalışmak için **`ts`** (time series) nesnesi kullanılır. Bu nesne, zaman serisi verilerini zaman dilimleri (örneğin aylar, yıllar) veya tarihler ile ilişkilendirerek işlem yapmanıza olanak tanır.

```r
AirPassengers
```

```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
```

```
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

```
class(AirPassengers)
```

```
[1] "ts"
```

```
diff(AirPassengers) # fark alma
```

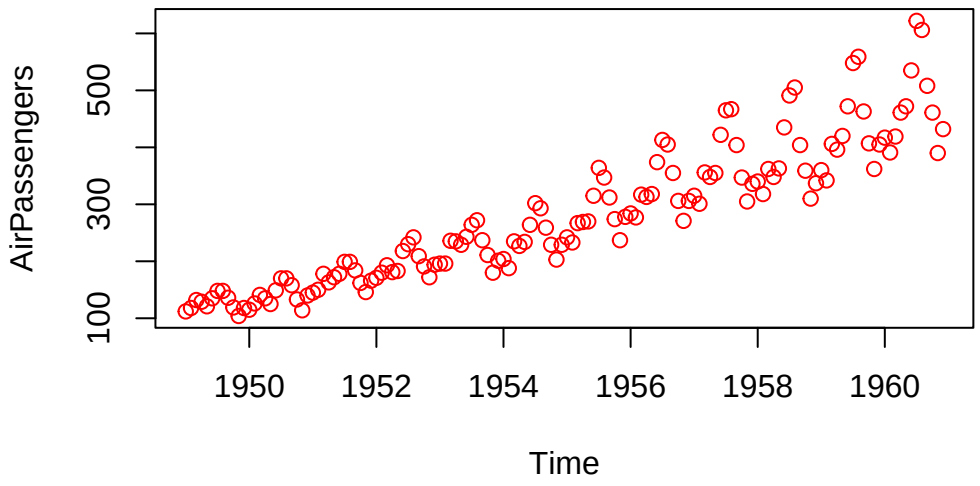
```
      Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
1949        6   14   -3   -8   14   13    0  -12  -17  -15   14
1950   -3   11   15   -6  -10   24   21    0  -12  -25  -19   26
1951    5    5   28  -15    9    6   21    0  -15  -22  -16   20
1952    5    9   13  -12    2   35   12   12  -33  -18  -19   22
1953    2    0   40   -1   -6   14   21    8  -35  -26  -31   21
1954    3  -16   47   -8    7   30   38   -9  -34  -30  -26   26
1955   13   -9   34    2    1   45   49  -17  -35  -38  -37   41
1956    6   -7   40   -4    5   56   39   -8  -50  -49  -35   35
1957    9  -14   55   -8    7   67   43    2  -63  -57  -42   31
1958    4  -22   44  -14   15   72   56   14 -101  -45  -49   27
1959   23  -18   64  -10   24   52   76   11  -96  -56  -45   43
1960   12  -26   28   42   11   63   87  -16  -98  -47  -71   42
```

```
stats::lag(AirPassengers,-1) # 1. gecikmesini alma
```

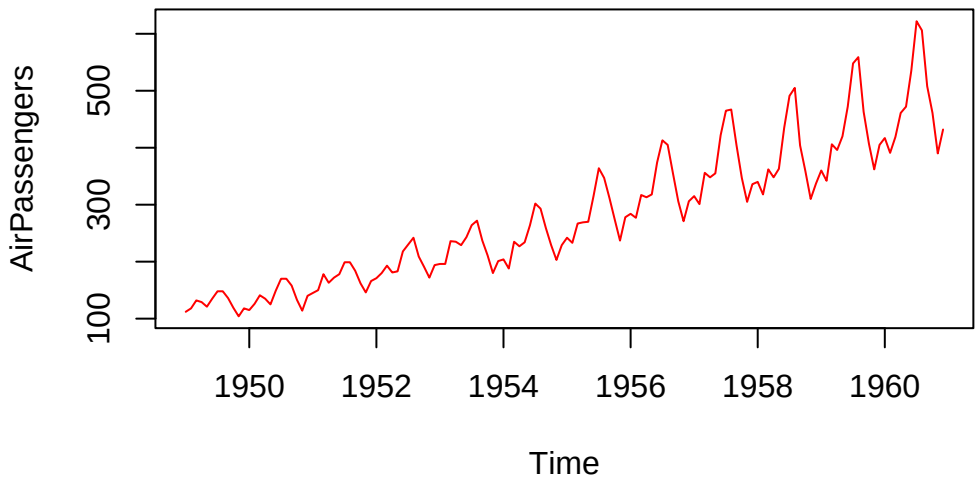
```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949     112 118 132 129 121 135 148 148 136 119 104
1950 118 115 126 141 135 125 149 170 170 158 133 114
1951 140 145 150 178 163 172 178 199 199 184 162 146
1952 166 171 180 193 181 183 218 230 242 209 191 172
1953 194 196 196 236 235 229 243 264 272 237 211 180
1954 201 204 188 235 227 234 264 302 293 259 229 203
1955 229 242 233 267 269 270 315 364 347 312 274 237
1956 278 284 277 317 313 318 374 413 405 355 306 271
1957 306 315 301 356 348 355 422 465 467 404 347 305
```

```
1958 336 340 318 362 348 363 435 491 505 404 359 310
1959 337 360 342 406 396 420 472 548 559 463 407 362
1960 405 417 391 419 461 472 535 622 606 508 461 390
1961 432
```

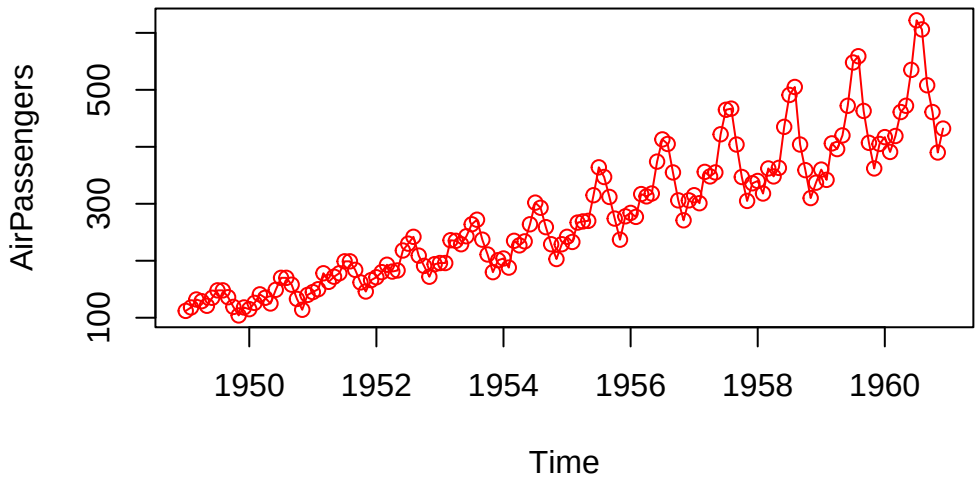
```r
plot(AirPassengers,type = "p", col = "red") # points
```



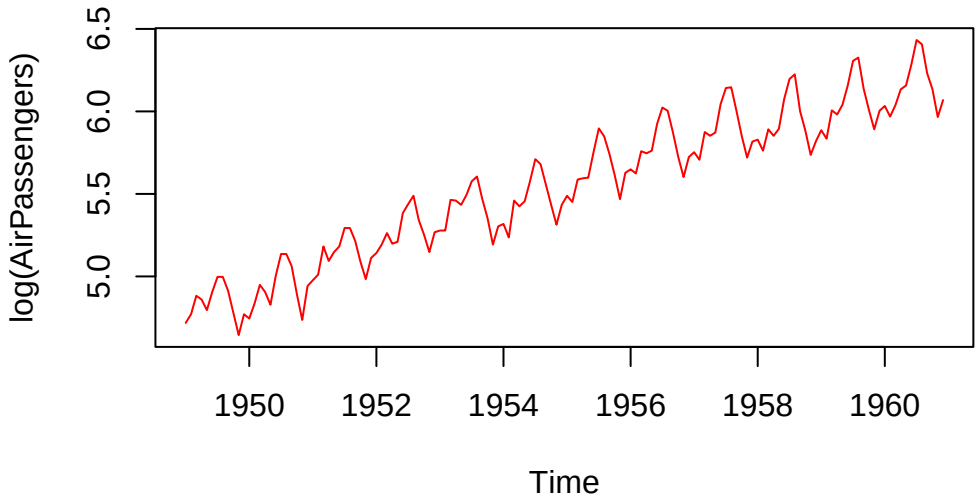

```r
plot(AirPassengers,type = "l", col = "red") # line
```



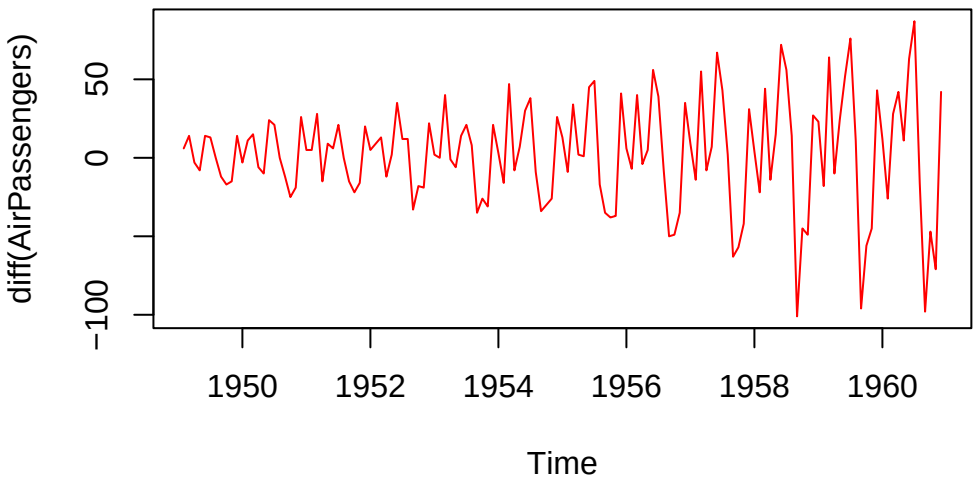

```
plot(AirPassengers,type = "o", col = "red") # points and line
```



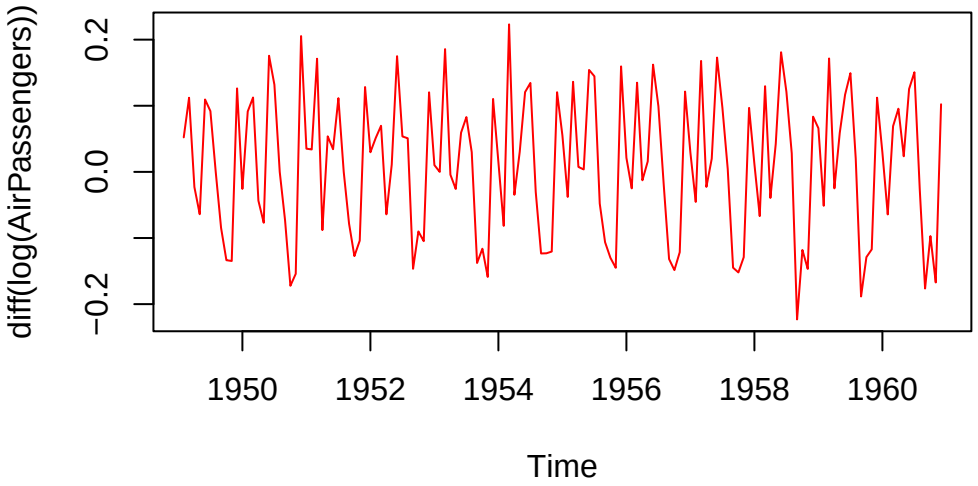
```r
plot(log(AirPassengers),type = "l", col = "red") # line
```




```r
plot(diff(AirPassengers),type = "l", col = "red") # line
```



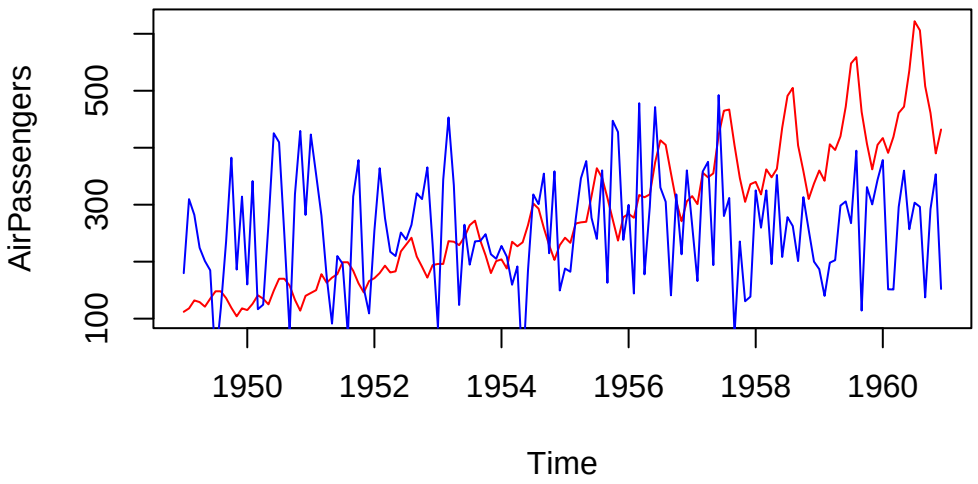

```
plot(diff(log(AirPassengers)),type = "l", col = "red") # line
```



```r
# çoklu zaman serisi
ts <- ts(rnorm(length(AirPassengers),250,100),start = c(1949,1),frequency=12)
ts
```

```
           Jan        Feb        Mar        Apr        May        Jun
1949 179.808315 309.751776 282.172902 224.347397 201.118120 184.854166
1950 159.992201 341.454637 116.559736 124.505482 266.325929 425.379615
1951 423.216699 352.816637 280.852724 172.809732  91.228299 209.857092
1952 254.520674 364.031496 276.441697 217.312438 209.832943 251.126070
1953  84.045094 343.893632 453.136057 334.123378 123.892533 264.815557
1954 227.607040 208.765137 159.514275 191.607779   2.182761 184.187226
1955 187.909648 182.259691 273.971787 346.388021 376.526984 276.928163
1956 299.707302 144.118592 478.141594 177.679782 296.175250 471.332244
1957 263.856784 165.936321 358.287919 375.261600 193.925229 492.371439
1958 325.114131 259.654614 325.156095 195.601724 352.144156 208.186136
1959 186.829530 139.900177 198.227299 202.596868 298.415176 305.769123
1960 378.340469 151.479795 151.140899 295.975133 359.786871 256.926163
           Jul        Aug        Sep        Oct        Nov        Dec
1949  22.698459 117.602002 235.455519 382.534941 186.095419 314.160898
1950 409.565682 247.594933  76.015116 318.379549 429.449441 282.004482
1951 196.467192  70.878911 314.526384 378.165212 153.296099 109.163020
1952 238.694414 264.593333 319.963872 309.844123 365.530431 229.688777
1953 194.754297 235.652666 236.453804 248.226145 212.982516 205.091206
1954 317.973039 300.983033 354.530638 214.799913 358.547188 149.396713
1955 239.880083 360.258157 162.876774 447.281152 427.493681 238.190162
1956 330.228209 305.078865 140.881663 318.130116 212.916490 360.215028
1957 280.097791 311.784731  72.724153 235.380967 130.527426 138.579397
1958 278.110828 262.488396 200.982009 313.248937 255.920028 200.111731
1959 267.574597 394.770661 114.103836 330.788117 300.530799 343.541722
1960 303.534628 296.187598 137.291803 292.347829 353.470966 152.071025
```

```r
plot(AirPassengers,type = "l",col = "red")
lines(ts, type = "l", col = "blue")
```

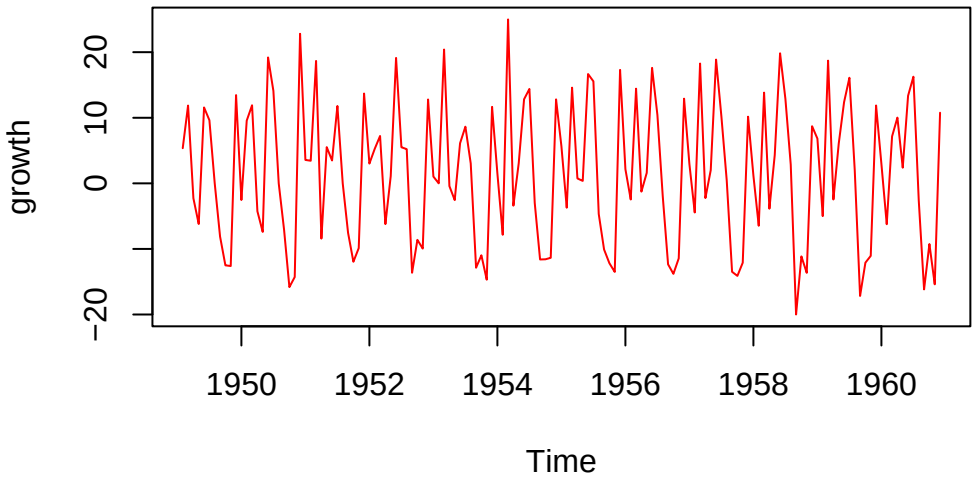



```
# yüzde değişim
growth <- AirPassengers/stats::lag(AirPassengers,-1)*100-100
growth
```

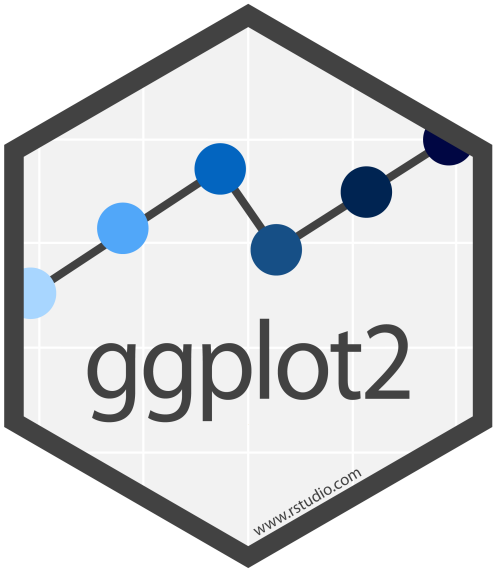
```
             Jan         Feb         Mar         Apr         May         Jun
1949                5.3571429  11.8644068  -2.2727273  -6.2015504  11.5702479
1950  -2.5423729   9.5652174  11.9047619  -4.2553191  -7.4074074  19.2000000
1951   3.5714286   3.4482759  18.6666667  -8.4269663   5.5214724   3.4883721
1952   3.0120482   5.2631579   7.2222222  -6.2176166   1.1049724  19.1256831
1953   1.0309278   0.0000000  20.4081633  -0.4237288  -2.5531915   6.1135371
1954   1.4925373  -7.8431373  25.0000000  -3.4042553   3.0837004  12.8205128
1955   5.6768559  -3.7190083  14.5922747   0.7490637   0.3717472  16.6666667
1956   2.1582734  -2.4647887  14.4404332  -1.2618297   1.5974441  17.6100629
1957   2.9411765  -4.4444444  18.2724252  -2.2471910   2.0114943  18.8732394
1958   1.1904762  -6.4705882  13.8364780  -3.8674033   4.3103448  19.8347107
1959   6.8249258  -5.0000000  18.7134503  -2.4630542   6.0606061  12.3809524
1960   2.9629630  -6.2350120   7.1611253  10.0238663   2.3861171  13.3474576
             Jul         Aug         Sep         Oct         Nov         Dec
1949   9.6296296   0.0000000  -8.1081081 -12.5000000 -12.6050420  13.4615385
1950  14.0939597   0.0000000  -7.0588235 -15.8227848 -14.2857143  22.8070175
1951  11.7977528   0.0000000  -7.5376884 -11.9565217  -9.8765432  13.6986301
1952   5.5045872   5.2173913 -13.6363636  -8.6124402  -9.9476440  12.7906977
```

```
1953   8.6419753   3.0303030 -12.8676471 -10.9704641 -14.6919431  11.6666667
1954  14.3939394  -2.9801325 -11.6040956 -11.5830116 -11.3537118  12.8078818
1955  15.5555556  -4.6703297 -10.0864553 -12.1794872 -13.5036496  17.2995781
1956  10.4278075  -1.9370460 -12.3456790 -13.8028169 -11.4379085  12.9151292
1957  10.1895735   0.4301075 -13.4903640 -14.1089109 -12.1037464  10.1639344
1958  12.8735632   2.8513238 -20.0000000 -11.1386139 -13.6490251   8.7096774
1959  16.1016949   2.0072993 -17.1735242 -12.0950324 -11.0565111  11.8784530
1960  16.2616822  -2.5723473 -16.1716172  -9.2519685 -15.4013015  10.7692308
```

```r
plot(growth,type = "l", col = "red")
```



ggplot2 ile Veri Görselleştirme



Bu bölümde ggplot2 paketi ile verilerin nasıl görselleştirldiğine bakacağız. ggplot2 grafiklerin dil bilgisi **(grammar of graphics)** prensiplerini temel alarak oluşturulmuştur. Bu prensiplere göre her grafik aynı parçalardan oluşturulabilir: bir veri seti, koordinat sistemi, ve "**`geom`**"lar - veri noktalarını temsil eden görsel işaretler.

ggplot2 ile veri görselleştirebilmemiz için önce grafik yapısını iyi tanımamız gerekiyor. Yatay eksen x ekseni, dikey eksen ise y ekseni olarak kabul ediliyor. Veri görselleştirmede **`ggplot()`** fonksiyonunu kullanıyoruz. ggplot() fonksiyonu içinde veri seti ismi ve **`aes()`** adlı estetik argümanına yatay ve dikey eksende kullanacağımız değişkenler (sütun isimleri) ile yer veriyoruz. Sonrasında, tercih edeceğimiz grafik tipine göre, **`geom`** fonksiyonlarından birini kullanacağız. Sıklıkla kullanılan geom fonksiyonları şunlardır:

- Nokta grafiği için `geom_point()`
- Çubuk veya sütun grafik için `geom_col()` ve `geom_bar()`
- Çizgi grafiği için `geom_line()`
- Histogram grafiği için `geom_histogram()`

- Boxplot grafiği için `geom_boxplot()`

Saçılım Grafikleri

Saçılım grafiği, genellikle fizik ve istatistik gibi bilimlerde kullanılan bir grafik türüdür. Saçılım grafiği, iki değişken arasındaki ilişkiyi görsel olarak göstermek için kullanılır. Bir eksende bir değişkenin değerleri, diğer eksende ise diğer değişkenin değerleri yer alır, ve her veri noktası bu iki değişkenin birleşimini temsil eder. Saçılım grafiği, veri noktalarının dağılımını, yoğunluklarını, odaklanma noktalarını ve olası eğilimleri anlamak için kullanılır. Bu grafik, veri setindeki aykırı değerleri tespit etmek, iki değişken arasındaki ilişkiyi değerlendirmek ve korelasyonu görsel olarak incelemek için oldukça kullanışlıdır.

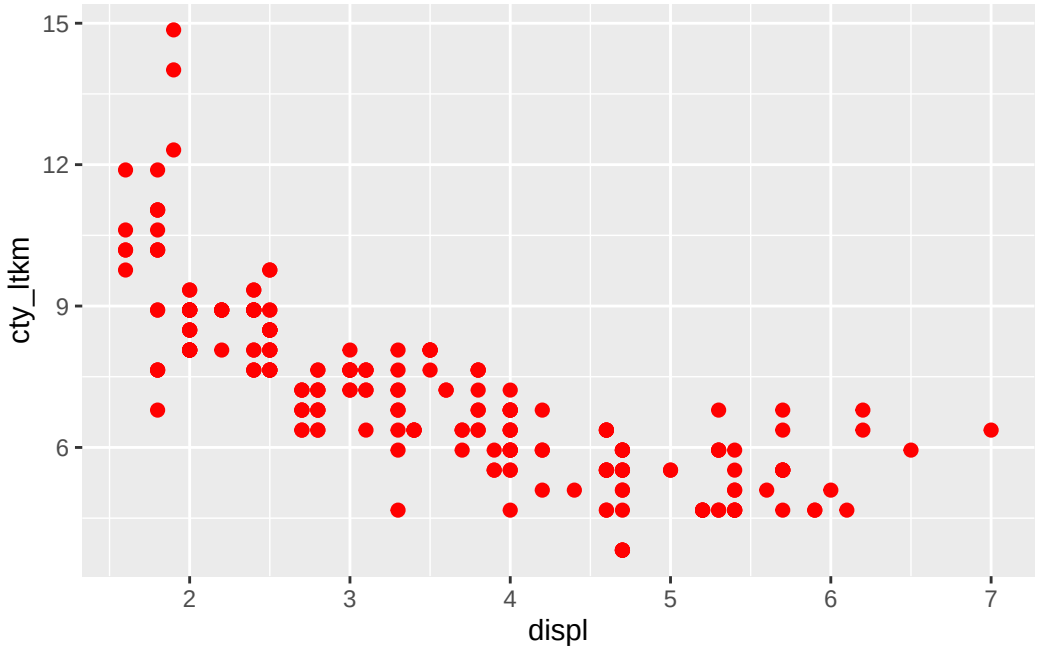
Saçılım grafiği kullanarak, iki değişken arasındaki ilişkinin doğası hakkında bilgi edinebilirsiniz. Örneğin, pozitif bir korelasyon varsa, veri noktaları genellikle yukarı doğru bir eğilim gösterirken, negatif bir korelasyon varsa, veri noktaları genellikle aşağı doğru bir eğilim gösterir. Korelasyon olmaması durumunda ise veri noktaları dağınık bir şekilde yayılmış olur. Saçılım grafiği, istatistiksel analizlerde veri keşfi yapmak ve ilişkileri anlamak için önemli bir araçtır.

```
library(ggplot2)
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
```

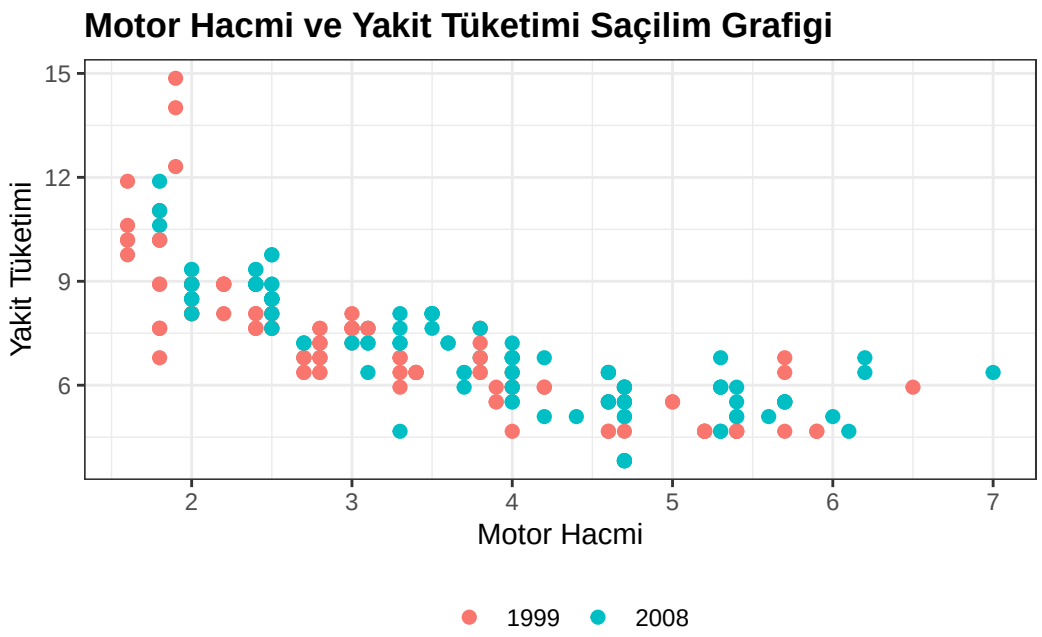
```
# Bir önceki bölümde üretilen yeni değişkenleri mpg veri setine yine ekleyelim.

# litre başına km hesaplama
galonmil_to_ltkm <- function(x){

  km <- x * 1.609/3.79
  return(km)
}
```

```r
df <- mpg
df$cty_ltkm <- galonmil_to_ltkm(df$cty)
df$hwy_ltkm <- galonmil_to_ltkm(df$hwy)

p1 <- ggplot(df,aes(x=displ,y=cty_ltkm)) +
  geom_point(size=2,color="red")
p1
```

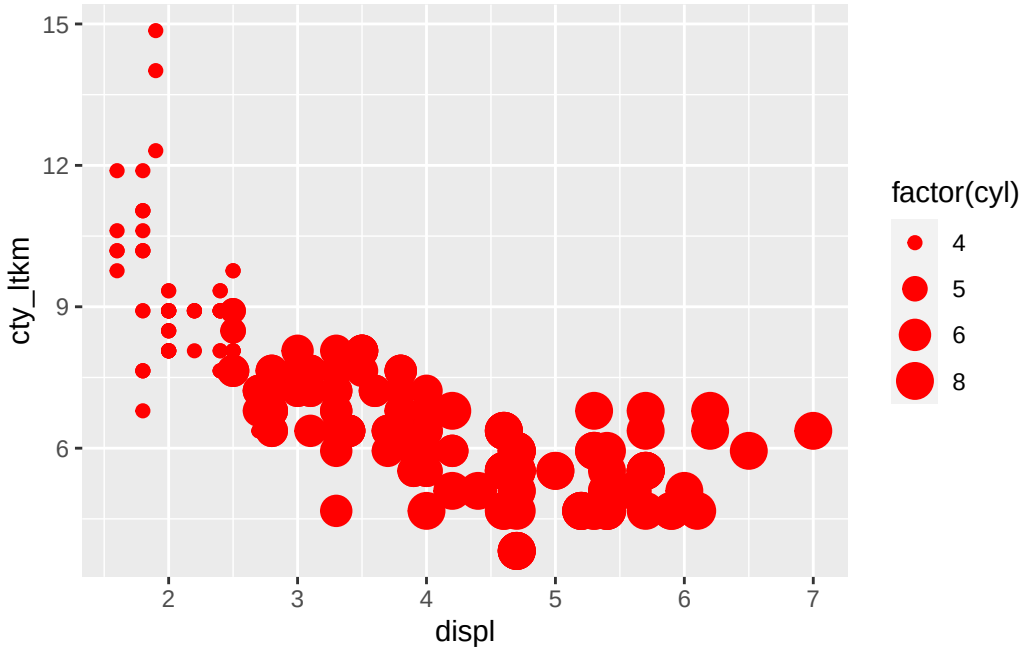



```r
# gruplar düzeyinde grafiği çizdirme
p2 <- ggplot(df,aes(x=displ,y=cty_ltkm,colour=as.factor(year))) +
  geom_point(size=2) +
  # grafiğe başlık ekleme
  ggtitle("Motor Hacmi ve Yakıt Tüketimi Saçılım Grafiği") +
  #eksenleri isimlendirme
  xlab("Motor Hacmi") +
  ylab("Yakıt Tüketimi")+
  theme_bw() + # tema değiştirme
  theme(legend.position = "bottom",  # gruplama değişkeninin poziyounun değiştirme
        plot.title = element_text(face = "bold"), # kalın başlık
        legend.title = element_blank()) # grup başlığını kaldırma
```

```
p2
```

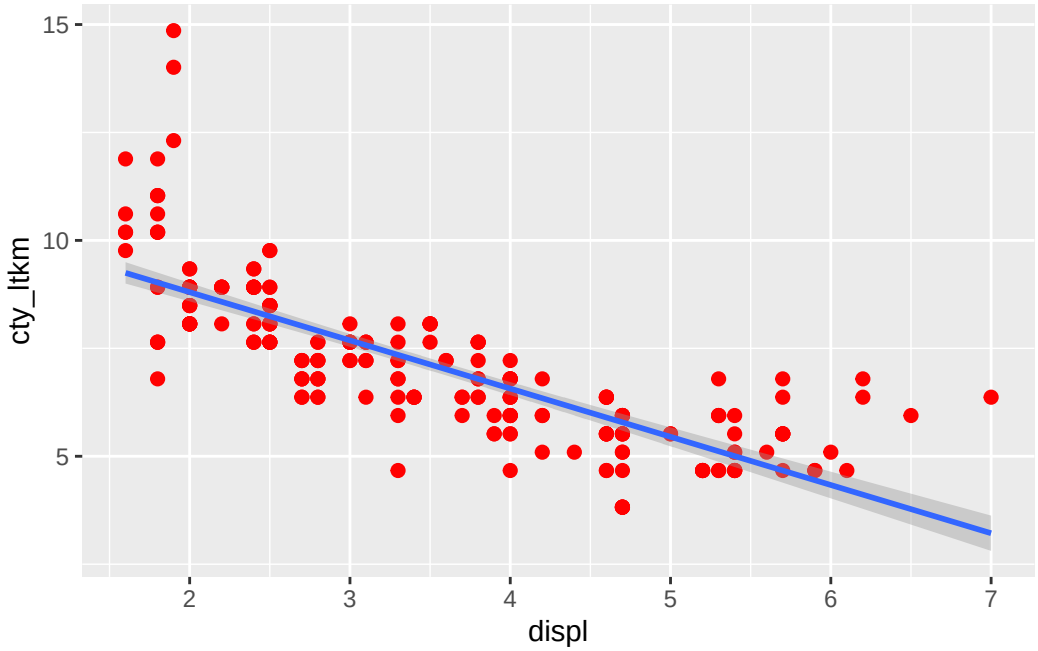



```
ggplot(df,aes(x=displ,y=cty_ltkm)) +
  geom_point(aes(size=factor(cyl)),color="red")
```



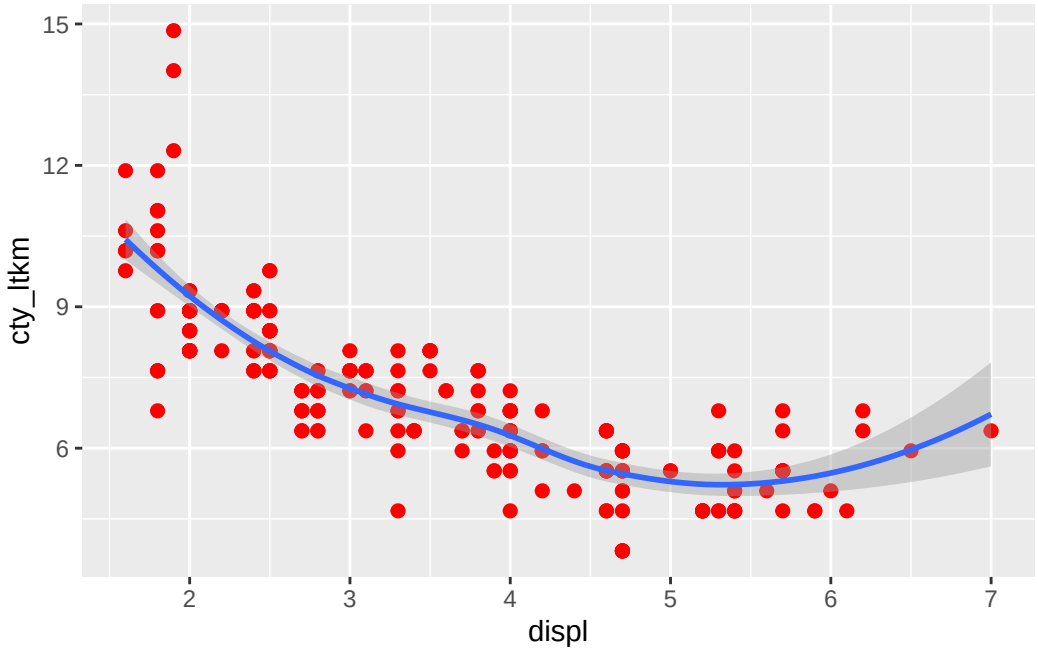

```
# grafiğe model eğrisi ekleme
p1 + geom_smooth(method = lm, se = TRUE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



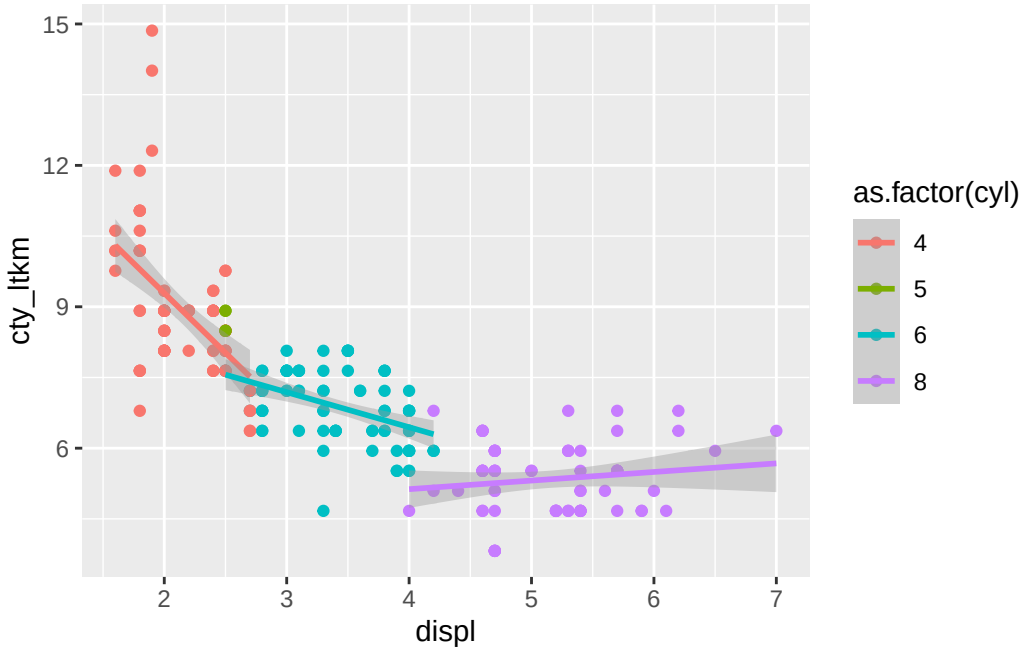

```
p1 + geom_smooth(method = loess, se = TRUE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



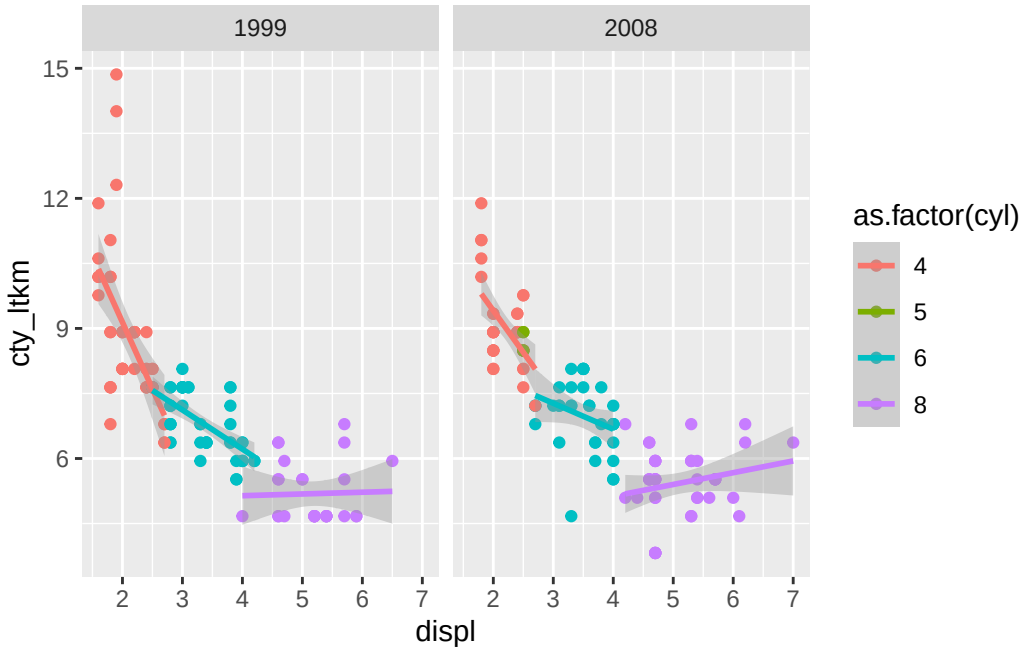

```
# grup düzeyinde model eğrileri ve saçılım grafiği
p3 <- df %>%
  ggplot(aes(x=displ,y=cty_ltkm,color=as.factor(cyl))) +
  geom_point()  +
  geom_smooth(method = lm, se = TRUE)
p3
```

```
`geom_smooth()` using formula = 'y ~ x'
```



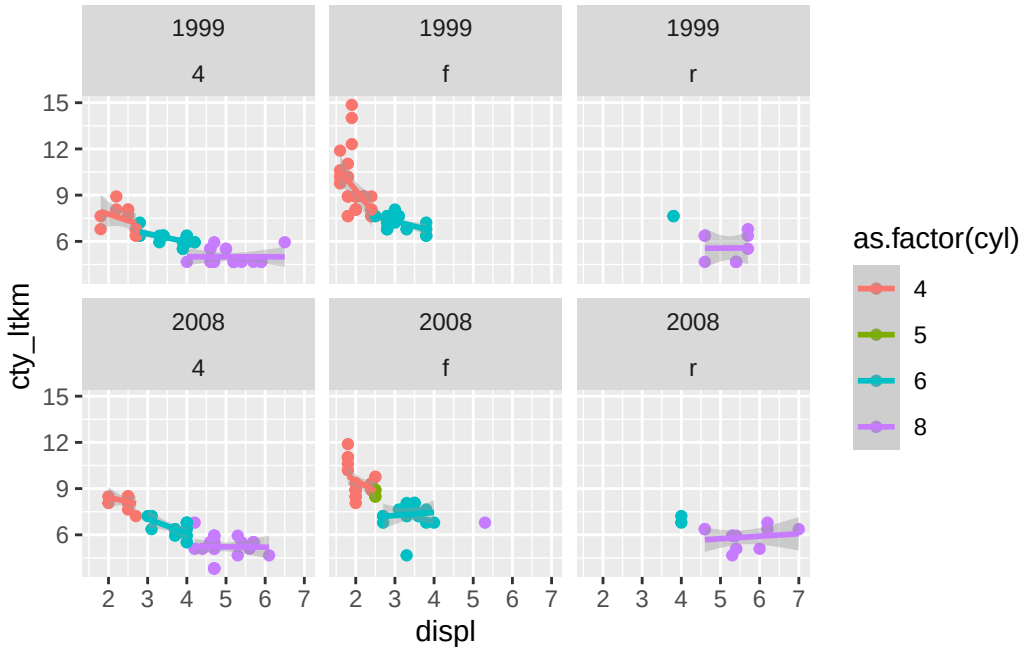

```
# grup ve yıl düzeyinde model eğrileri ve saçılım grafiği
p3 + facet_wrap(~ year)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



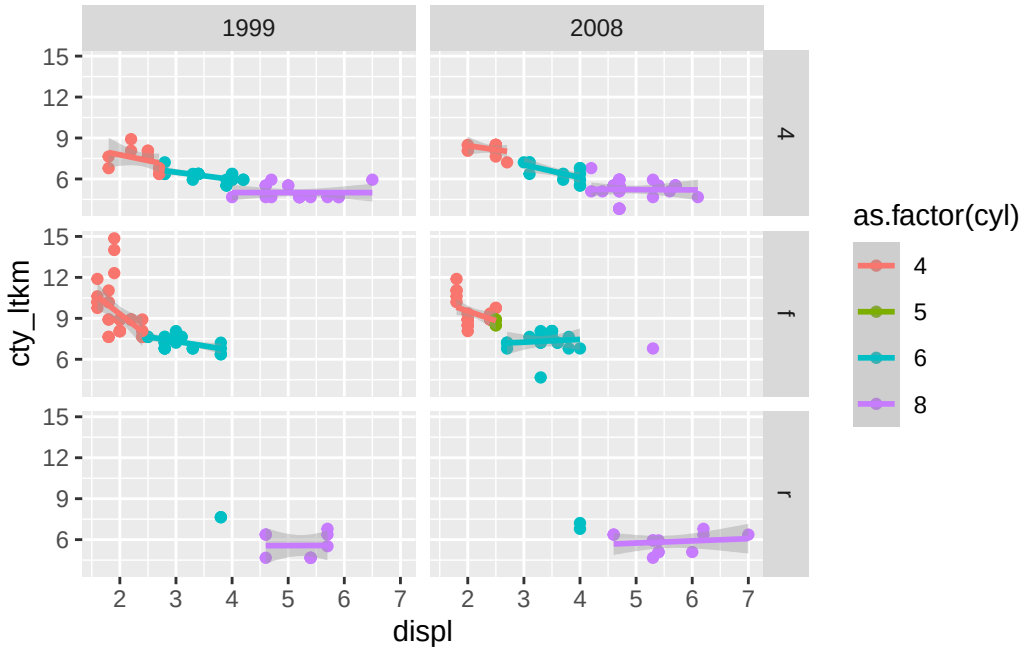

```
p3 + facet_wrap(~ year+drv)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



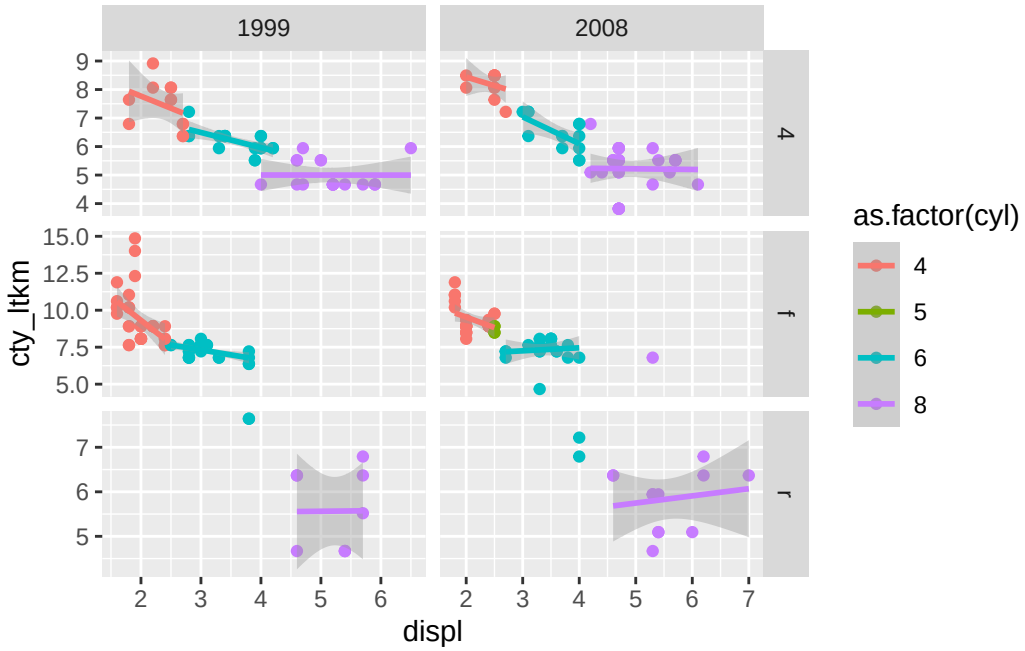

```
p3 + facet_grid(drv ~ year) # eksen aralıkları sabit
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
p3 + facet_grid(drv ~ year,scales = "free") # eksen aralıkları değişken
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Zaman Serisi Grafikleri

Zaman serisi grafikleri, zamanla değişen verileri görsel olarak temsil etmek için kullanılan grafiklerdir. Bu tür grafikler, belirli bir süre boyunca gözlemlenen verileri analiz etmek, eğilimleri belirlemek, dönemsel desenleri tanımak ve istatistiksel analizler yapmak için yaygın olarak kullanılır. Zaman serisi verileri genellikle sabit aralıklarla veya farklı zaman dilimlerinde toplanır. En yaygın olan türü çizgi grafikleri olmakla birlikte sütun ve alan grafikleri de zaman serilerinin görselleştirilmesinde kullanılabilmektedir.

Örnekler ggplot2 paketi ile birlikte gelen **economics** veri seti ile yapılacaktır.

```
economics
```

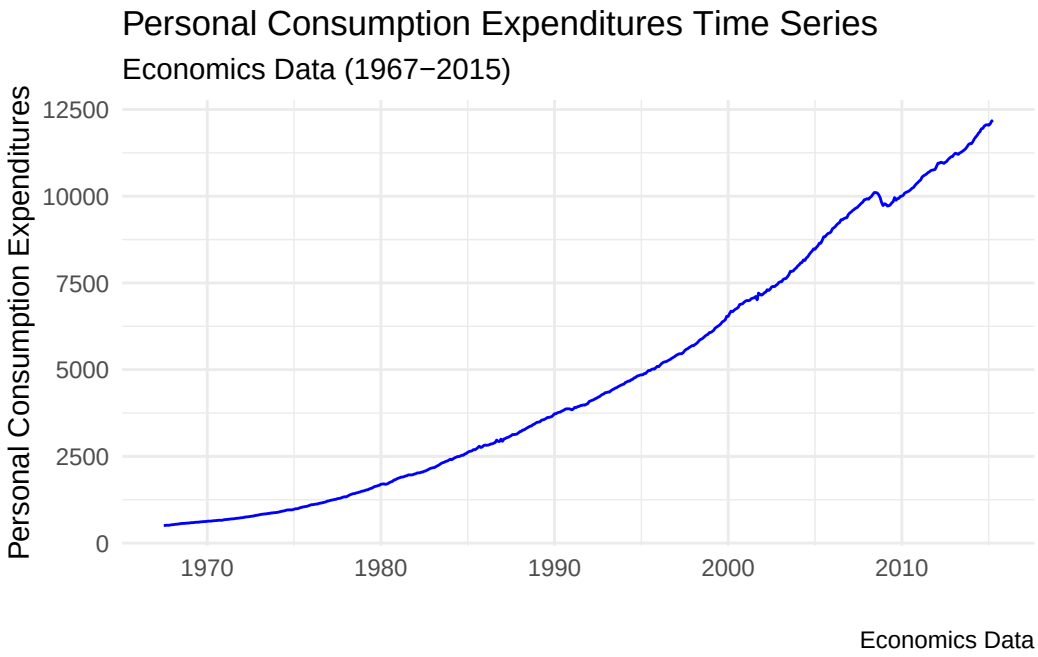
```
# A tibble: 574 x 6
   date         pce    pop psavert uempmed unemploy
   <date>     <dbl>  <dbl>   <dbl>   <dbl>     <dbl>
 1 1967-07-01  507. 198712    12.6     4.5      2944
 2 1967-08-01  510. 198911    12.6     4.7      2945
 3 1967-09-01  516. 199113    11.9     4.6      2958
 4 1967-10-01  512. 199311    12.9     4.9      3143
 5 1967-11-01  517. 199498    12.8     4.7      3066
```

```
 6 1967-12-01  525. 199657    11.8     4.8     3018
 7 1968-01-01  531. 199808    11.7     5.1     2878
 8 1968-02-01  534. 199920    12.3     4.5     3001
 9 1968-03-01  544. 200056    11.7     4.1     2877
10 1968-04-01  544  200208    12.3     4.6     2709
# i 564 more rows
```

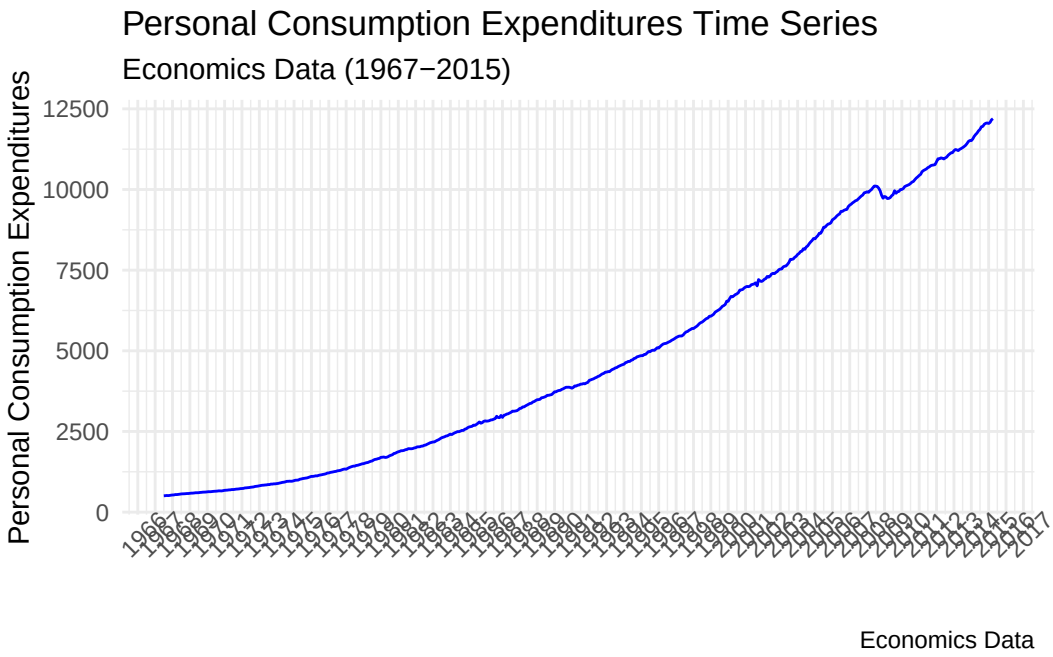
```r
summary(economics)
```

```
      date                 pce                pop            psavert
 Min.   :1967-07-01   Min.   :  506.7   Min.   :198712   Min.   : 2.200
 1st Qu.:1979-06-08   1st Qu.: 1578.3   1st Qu.:224896   1st Qu.: 6.400
 Median :1991-05-16   Median : 3936.8   Median :253060   Median : 8.400
 Mean   :1991-05-17   Mean   : 4820.1   Mean   :257160   Mean   : 8.567
 3rd Qu.:2003-04-23   3rd Qu.: 7626.3   3rd Qu.:290291   3rd Qu.:11.100
 Max.   :2015-04-01   Max.   :12193.8   Max.   :320402   Max.   :17.300
    uempmed          unemploy
 Min.   : 4.000   Min.   : 2685
 1st Qu.: 6.000   1st Qu.: 6284
 Median : 7.500   Median : 7494
 Mean   : 8.609   Mean   : 7771
 3rd Qu.: 9.100   3rd Qu.: 8686
 Max.   :25.200   Max.   :15352
```

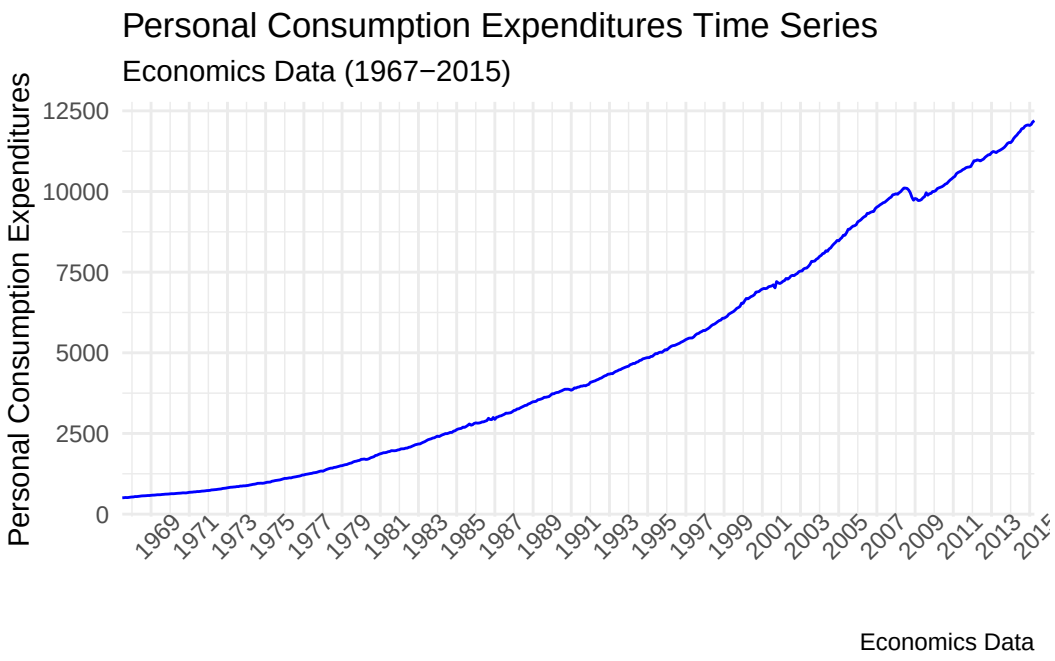
```r
p4 <- economics %>%
  ggplot(aes(x=date,y=pce)) +
  geom_line(color="blue") +
  theme_minimal() +
  labs(x = "",
       y = "Personal Consumption Expenditures",
       title = "Personal Consumption Expenditures Time Series",
       caption = "Economics Data",
       subtitle = "Economics Data (1967-2015)")
p4
```




```
# zaman eksenini ayarlama
p4 +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  theme(axis.text.x = element_text(angle = 45), legend.position = "top")
```

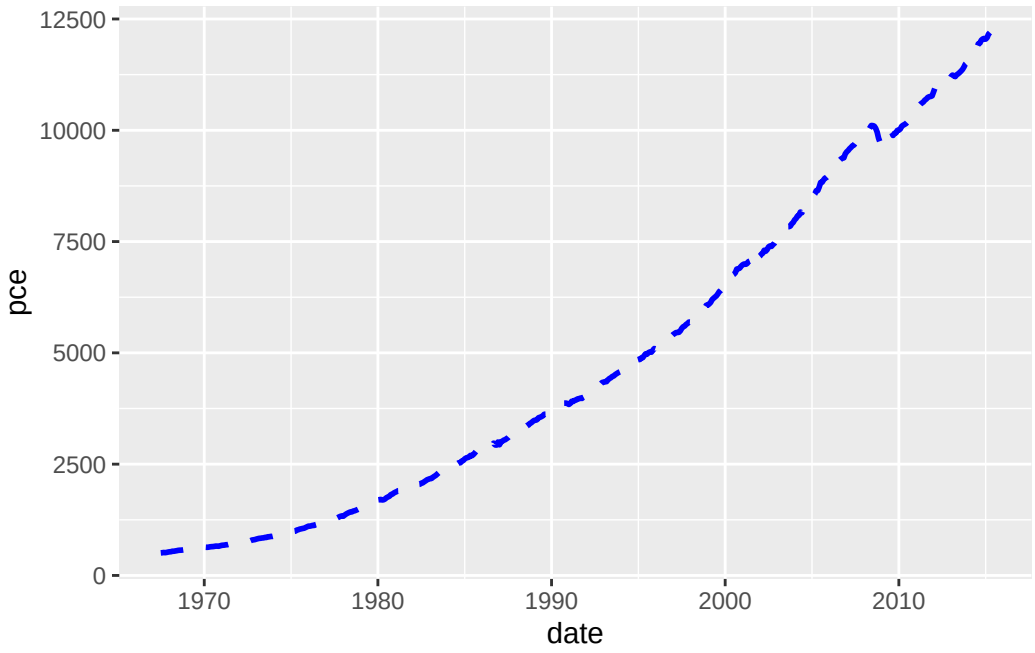



```
p4 +
  scale_x_date(date_breaks = "2 year", date_labels = "%Y",expand = c(0,0)) +
  theme(axis.text.x = element_text(angle = 45), legend.position = "top")
```

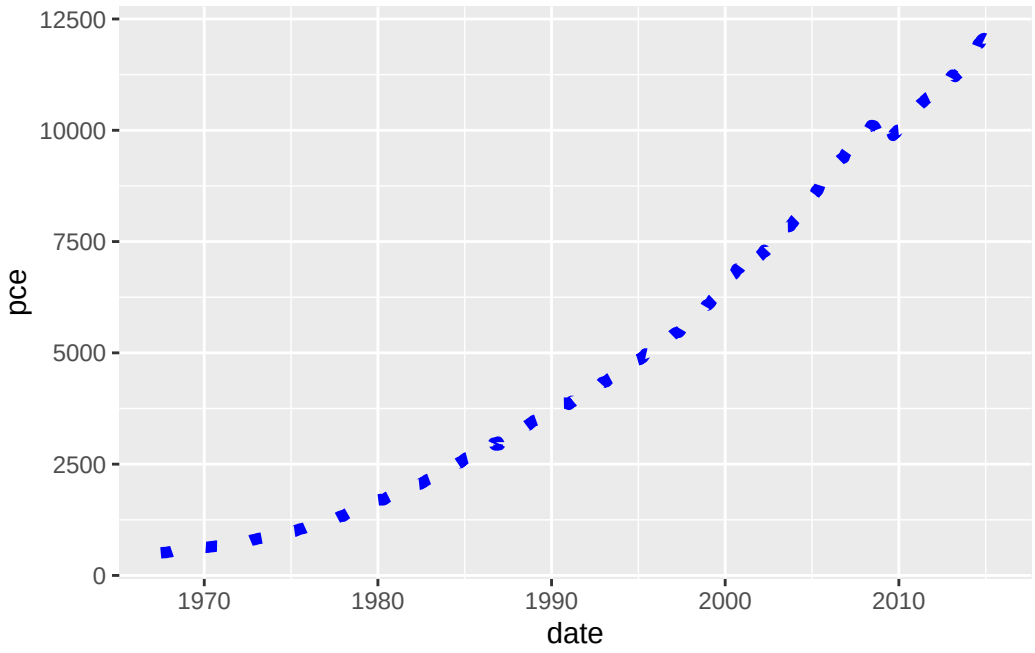



```
# çizgi türü değiştirilebilir
economics %>%
  ggplot(aes(x=date,y=pce)) +
  geom_line(linetype = "dashed", size = 1, colour = "blue")
```

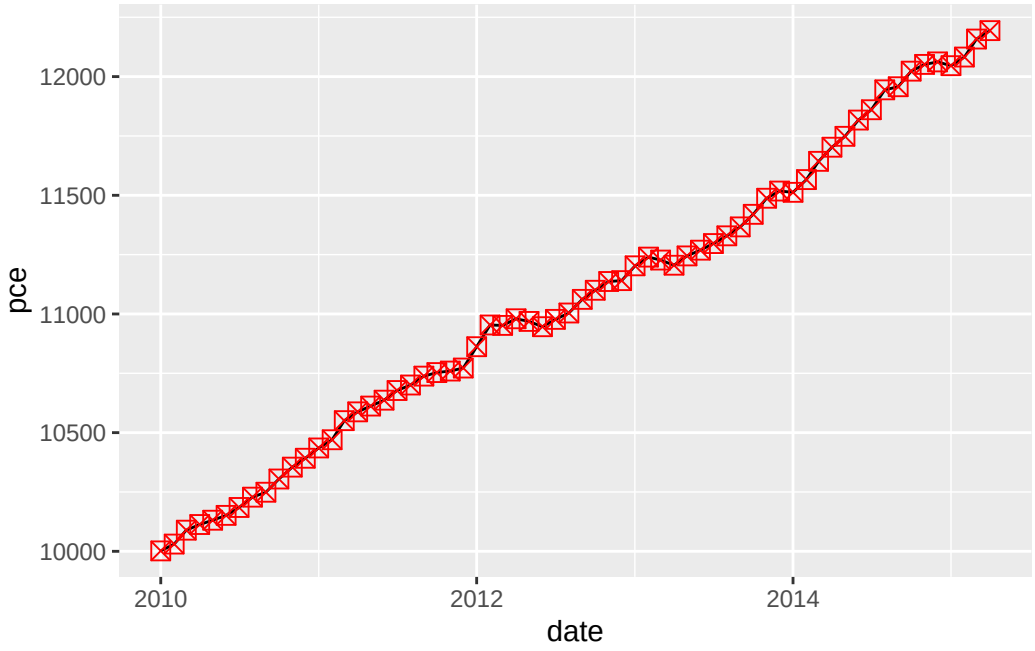
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



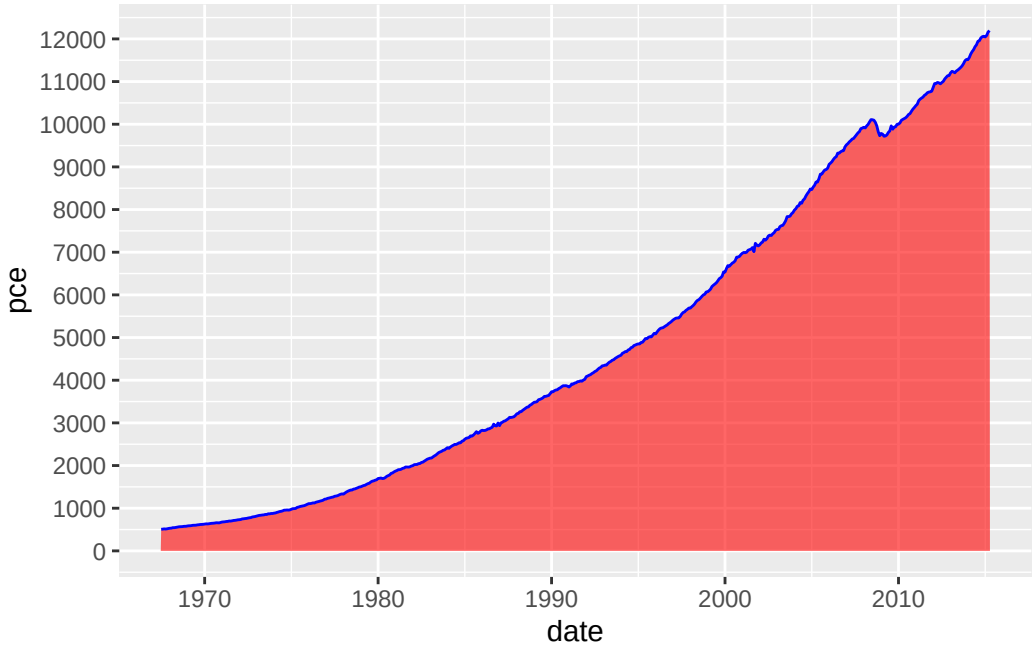

```
economics %>%
  ggplot(aes(x=date,y=pce)) +
  geom_line(linetype = "dotted", size = 2, colour = "blue")
```



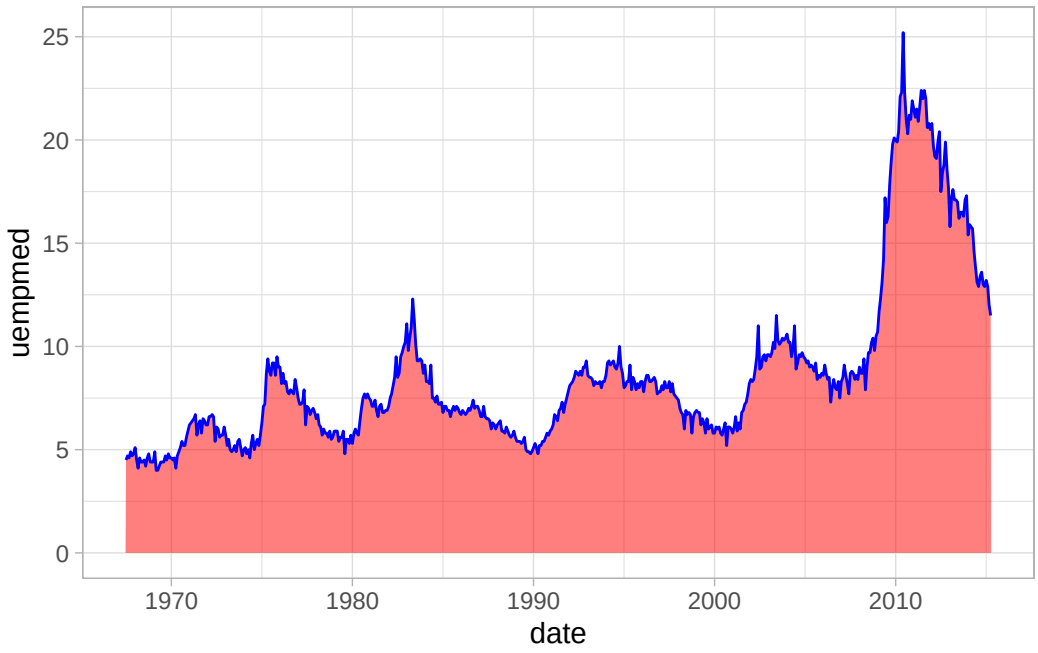

```
# zaman grafiğine noktalar ekleme
economics %>%
  filter(lubridate::year(date) >= 2010) %>%
  ggplot(aes(x=date,y=pce)) +
  geom_line()+
  geom_point(size = 3, shape= 7, colour = "red")
```




```r
# gölgeli zaman grafiği
economics %>%
  ggplot(aes(x=date,y=pce)) +
  geom_area(color="blue",fill="red",alpha=0.6) +
  # y ekseni aralıklarını ayarlama
  scale_y_continuous(breaks = seq(0, max(economics$pce), by = 1000))
```




```
economics %>%
  ggplot(aes(x=date,y=uempmed )) +
  geom_area(color="blue",fill="red",alpha=0.5) +
  theme_light()
```

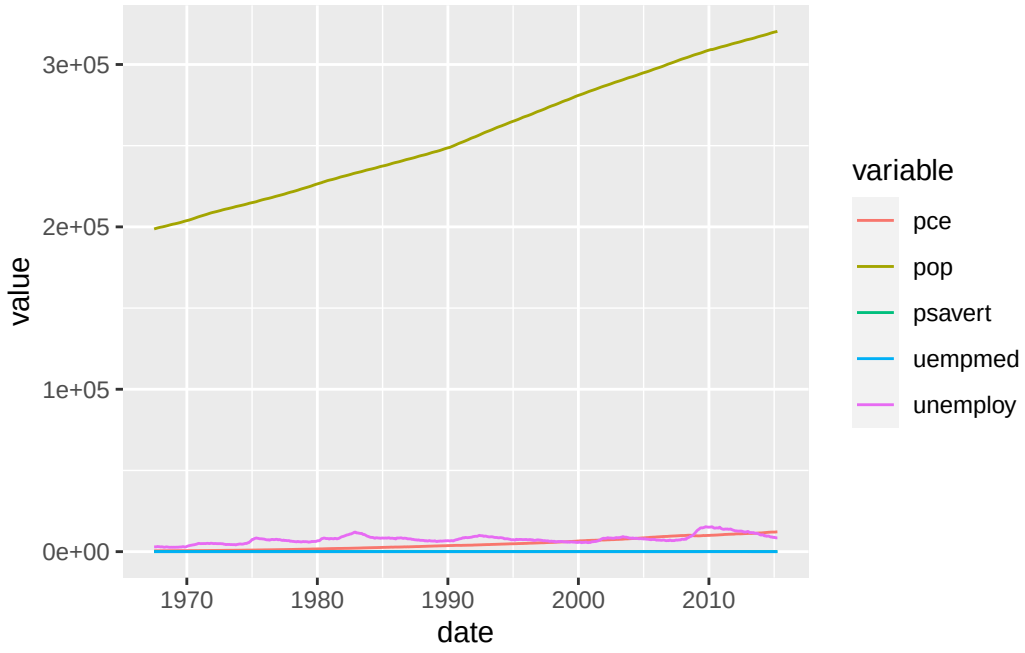



```
# çoklu zaman serisi grafiği
economics_long
```

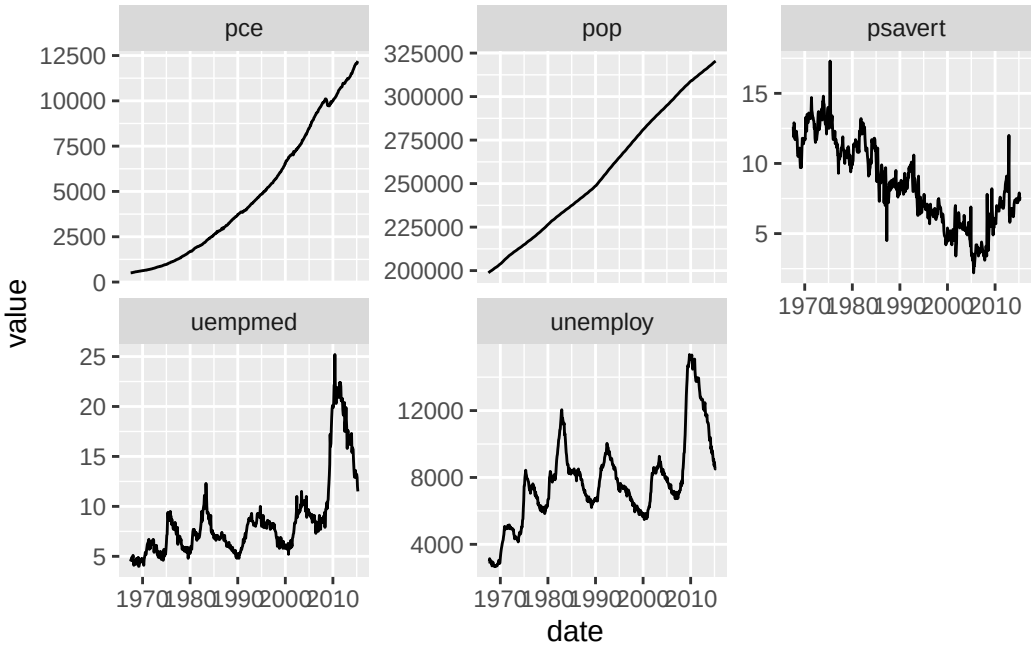
```
# A tibble: 2,870 x 4
   date       variable value  value01
   <date>     <chr>    <dbl>    <dbl>
 1 1967-07-01 pce       507. 0
 2 1967-08-01 pce       510. 0.000265
 3 1967-09-01 pce       516. 0.000762
 4 1967-10-01 pce       512. 0.000471
 5 1967-11-01 pce       517. 0.000916
 6 1967-12-01 pce       525. 0.00157
 7 1968-01-01 pce       531. 0.00207
 8 1968-02-01 pce       534. 0.00230
 9 1968-03-01 pce       544. 0.00322
10 1968-04-01 pce       544  0.00319
# i 2,860 more rows
```

```
# serilerin ölçekleri farklı
economics_long %>%
  ggplot(aes(x=date,y=value,color=variable))+
```

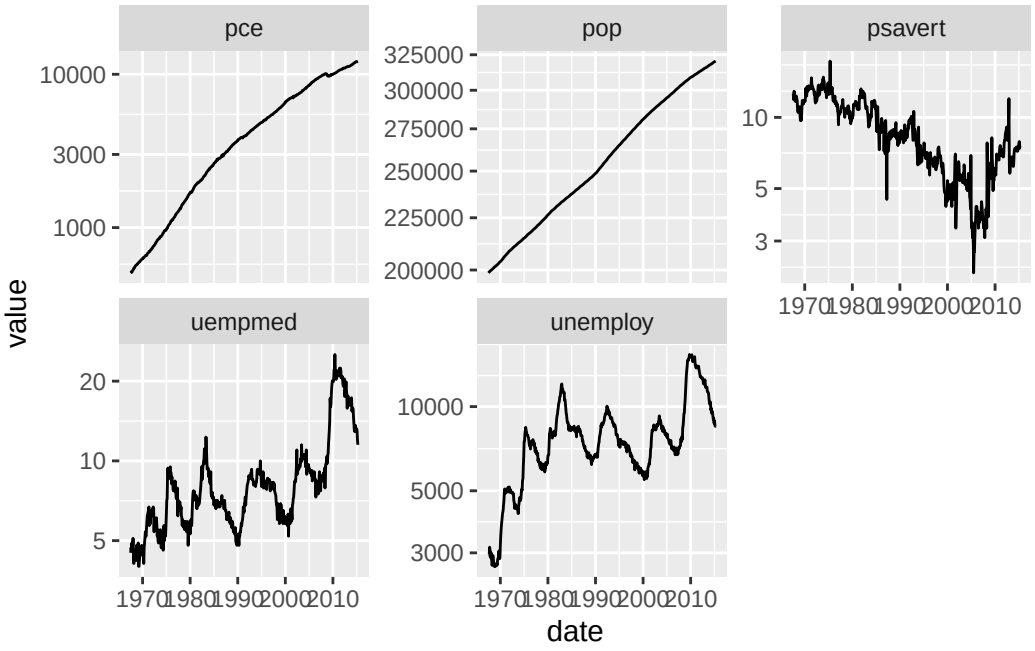
```
  geom_line()
```




```
economics_long %>%
  ggplot(aes(x=date,y=value))+
  geom_line() +
  facet_wrap(~variable,scales = "free_y")
```




```r
economics_long %>%
  ggplot(aes(x=date,y=value))+
  geom_line() +
  facet_wrap(~variable,scales = "free_y")+
  scale_y_log10() # y eksenlerinin logatirması alınır
```



Sütun Grafikleri

Sütun grafikleri, verileri kategorik veya gruplara göre temsil etmek için kullanılan bir grafik türüdür. Bu grafik türü, farklı kategorilerin veya grupların sayısal değerlerini karşılaştırmak veya görselleştirmek için kullanılır. Sütun grafikleri dikey çubuklardan oluşur ve her çubuk, bir kategori veya grup için bir değeri temsil eder. Sütun grafiklerinin temel bileşenleri şunlardır:

1. **Yatay Eksen (X-Eksen):** Bu eksende kategoriler veya gruplar yer alır. Örneğin, bir yıl boyunca aylar, ürün kategorileri, bölgeler veya şirket departmanları gibi farklı kategoriler olabilir.
2. **Dikey Eksen (Y-Eksen):** Bu eksende sayısal değerler yer alır ve sütunların yükseklikleri bu değerleri temsil eder. Değerler genellikle sayısal verilerdir ve karşılaştırılabilir bir ölçü birimi içinde bulunurlar.
3. **Sütunlar:** Sütunlar, her bir kategori veya grup için bir değeri temsil eder. Sütunların yükseklikleri, karşılaştırılan değerlerin büyüklüğünü veya ilişkilerini gösterir.

Sütun grafikleri, aşağıdaki amaçlar için kullanılır:

- Karşılaştırmalar: Farklı kategorilerin veya grupların değerlerini karşılaştırmak için kullanılır. Örneğin, farklı ülkelerin gayri safi yurtiçi hasıla (GSYİH) değerlerini karşılaştırmak için sütun grafikleri kullanılabilir.

- Zaman İçi Değişim: Zaman serisi verilerini temsil etmek için kullanılabilir. Her sütun, belirli bir zaman dilimindeki değerleri gösterebilir.
- Kategorik Verilerin İncelenmesi: Ürün kategorileri, şirket departmanları veya müşteri segmentleri gibi kategorik verilerin analizi için kullanılabilir.

Sütun grafikleri, verileri görsel olarak anlamak ve veriler arasındaki farkları veya eğilimleri vurgulamak için etkili bir araçtır. Aynı zamanda verilerin daha kolay anlaşılmasına yardımcı olabilir ve karar verme süreçlerine katkı sağlayabilir.

Örnekler ggplot2 paketi ile birlikte gelen **diamonds** veri seti ile yapılacaktır.

```
diamonds
```

```
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
# i 53,930 more rows
```

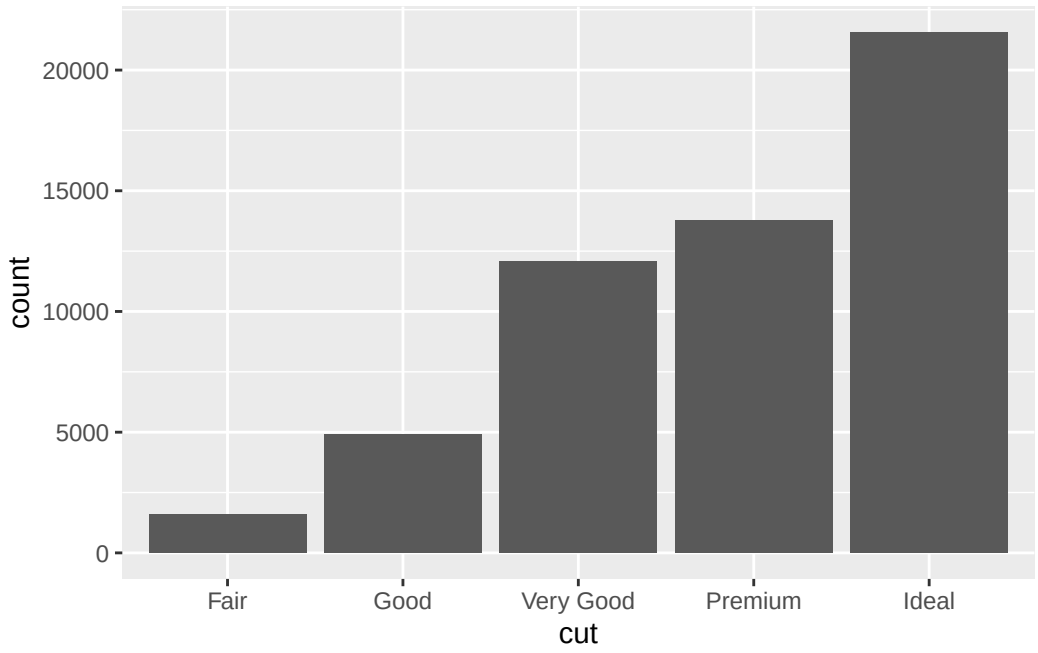
```
glimpse(diamonds)
```

```
Rows: 53,940
Columns: 10
$ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.~
$ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver~
$ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,~
$ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ~
$ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64~
$ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58~
$ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34~
$ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.~
$ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.~
$ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

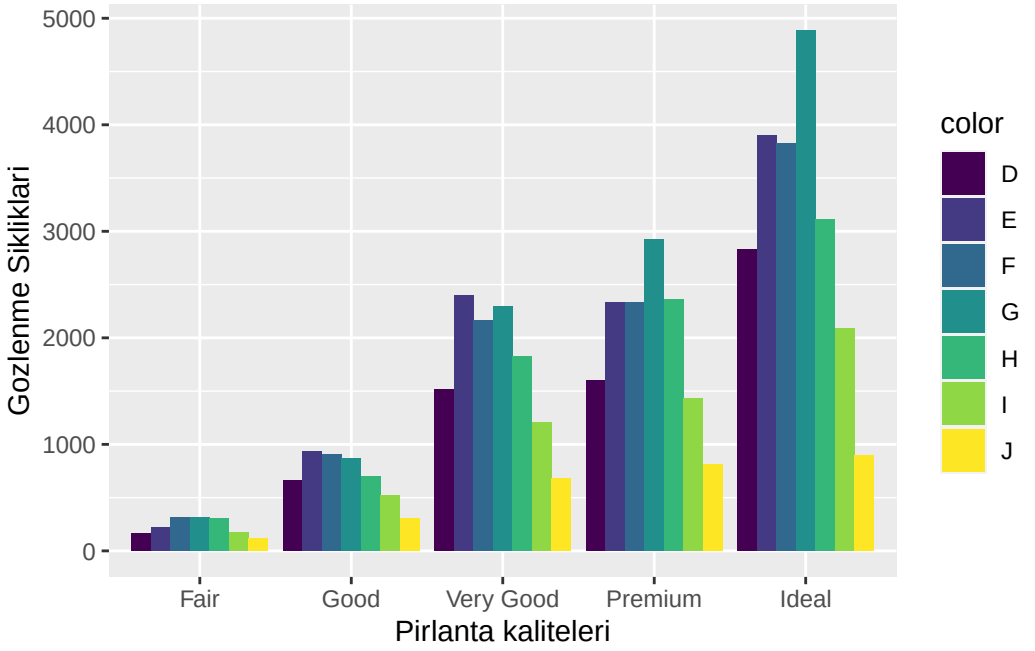
```r
summary(diamonds)
```

```
    carat               cut        color        clarity          depth      
Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065   Min.   :43.00  
1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258   1st Qu.:61.00  
Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194   Median :61.80  
Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75  
3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066   3rd Qu.:62.50  
Max.   :5.0100                     I: 5422   VVS1    : 3655   Max.   :79.00  
                                   J: 2808   (Other): 2531                  
    table           price             x                y         
Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000  
1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720  
Median :57.00   Median : 2401   Median : 5.700   Median : 5.710  
Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735  
3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540  
Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900  

      z         
Min.   : 0.000  
1st Qu.: 2.910  
Median : 3.530  
Mean   : 3.539  
3rd Qu.: 4.040  
Max.   :31.800  
```

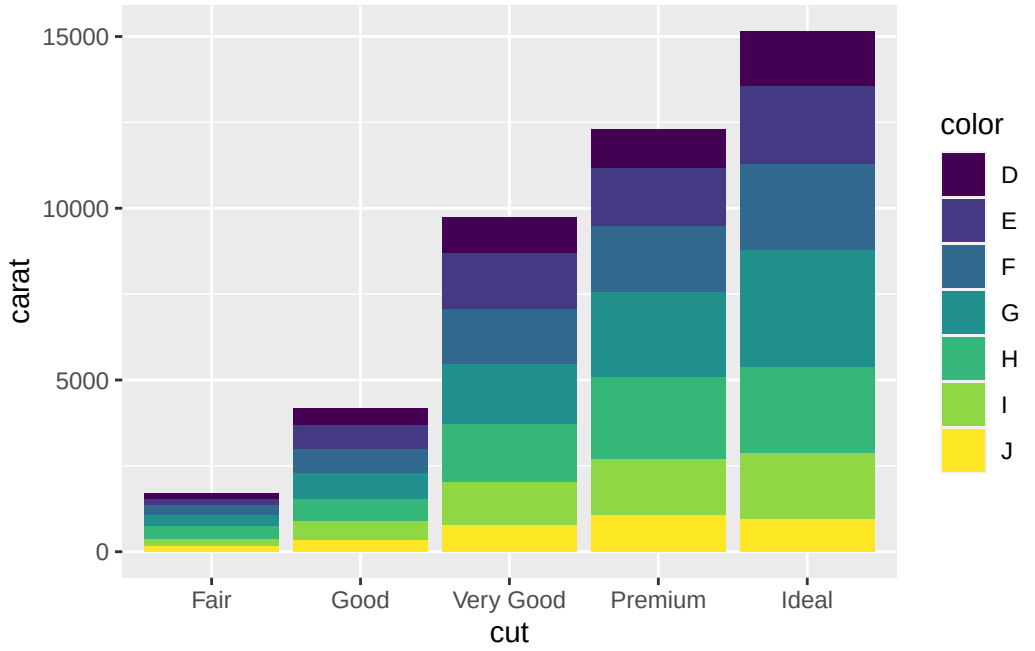
```r
# sıklık durumunu görselleştirme
ggplot(diamonds, aes(cut)) +
  geom_bar()
```



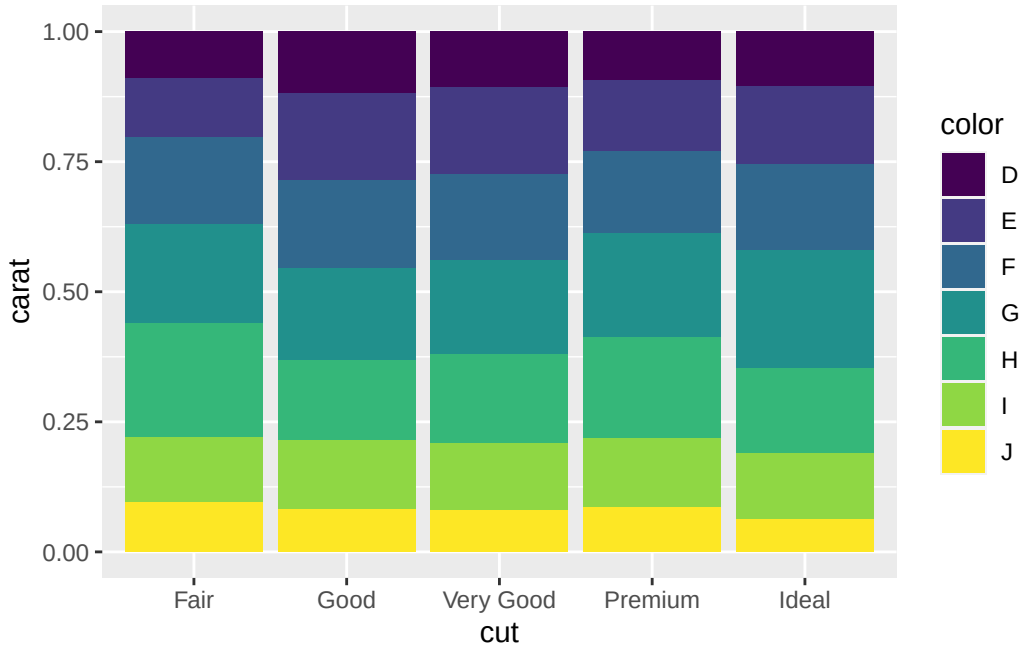

```
ggplot(diamonds, aes(cut, fill = color)) +
  geom_bar(position = position_dodge()) +
  xlab("Pirlanta kaliteleri") +
  ylab("Gozlenme Sikliklari")
```



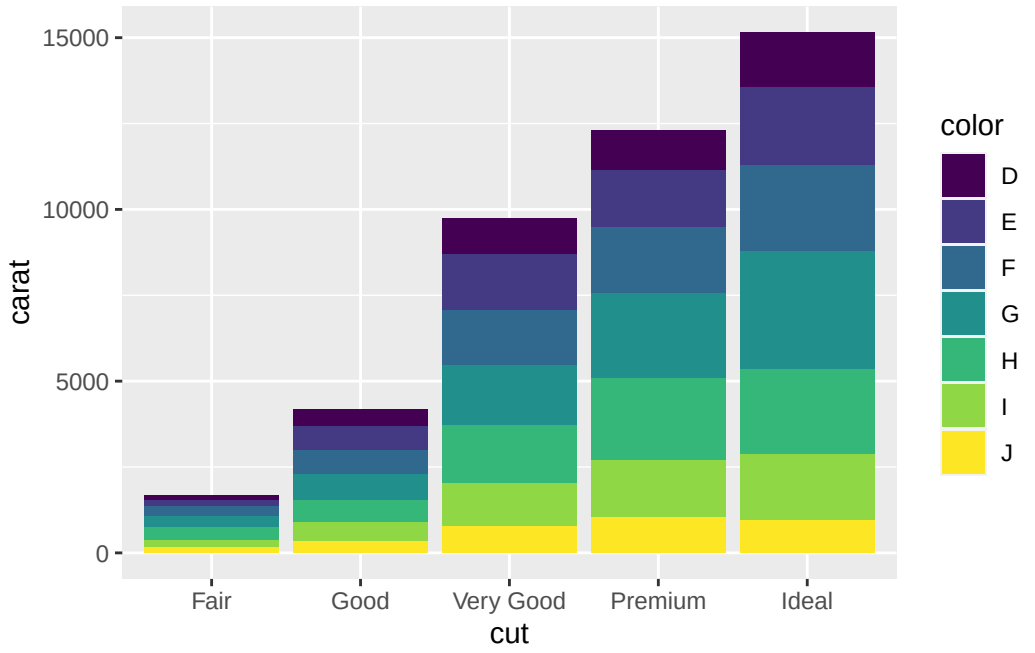

```
ggplot(diamonds, aes(x=cut, y=carat,fill = color)) +
  geom_bar(stat = "identity")
```



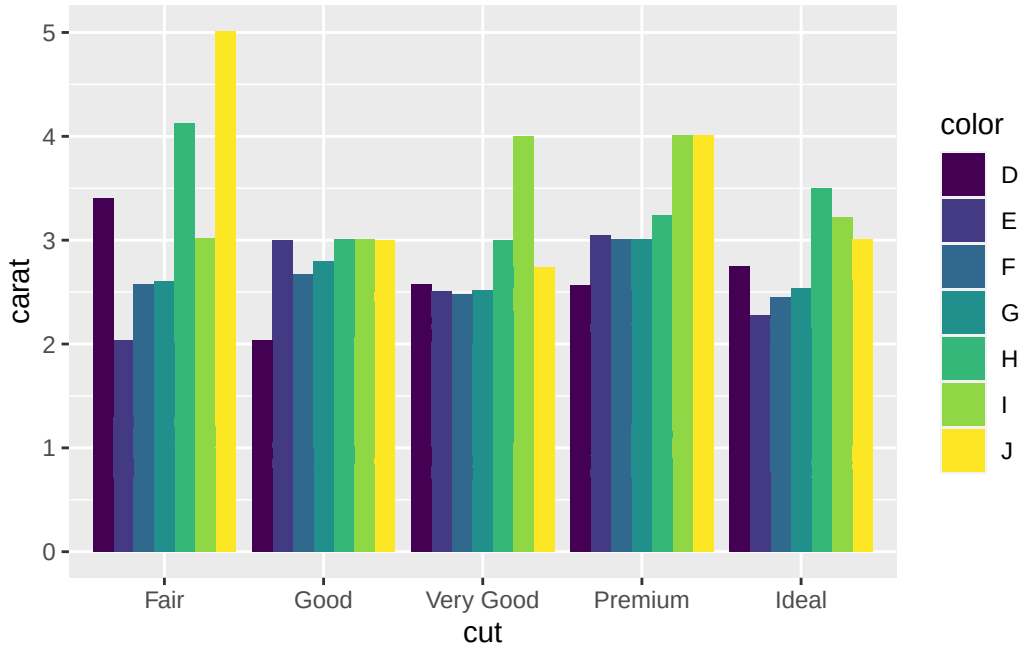
```
ggplot(diamonds, aes(x=cut, y=carat,fill = color)) +
  # fill ile oransal olarak gösterim yapılır
  geom_bar(stat = "identity",position = "fill")
```



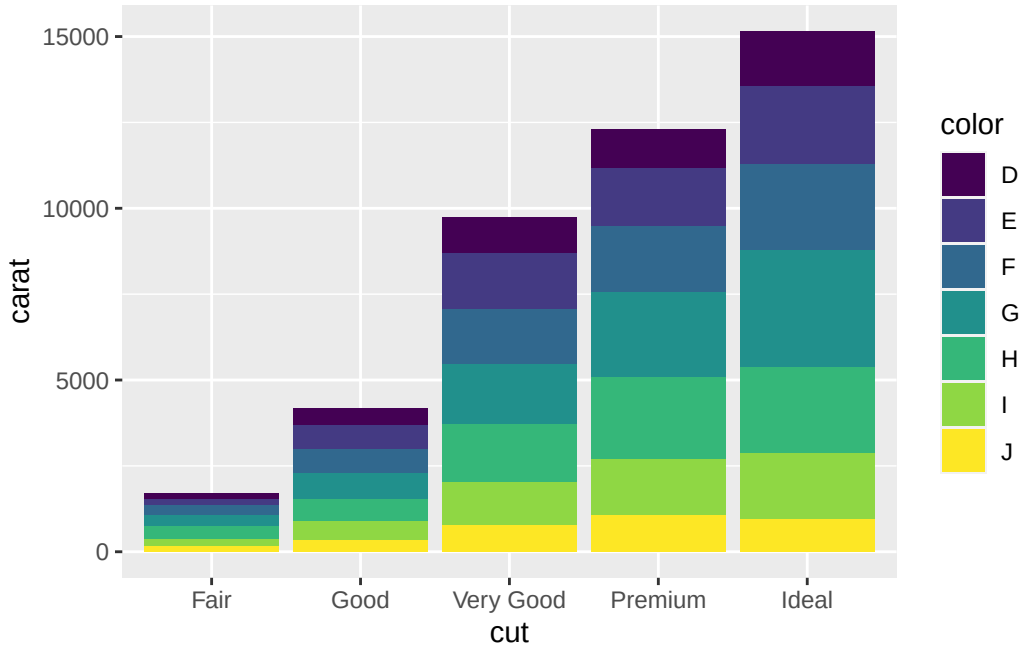

```
ggplot(diamonds, aes(x=cut,y=carat, fill = color)) +
  geom_col() # y ekseni toplanarak yığılmış
```




```
ggplot(diamonds, aes(x=cut,y=carat,, fill = color)) +
  geom_col(position = "dodge") # y ekseni değerleri
```



```
ggplot(diamonds, aes(x=cut,y=carat, fill = color)) +
  geom_col(position = "stack")
```



Dağılım Grafikleri

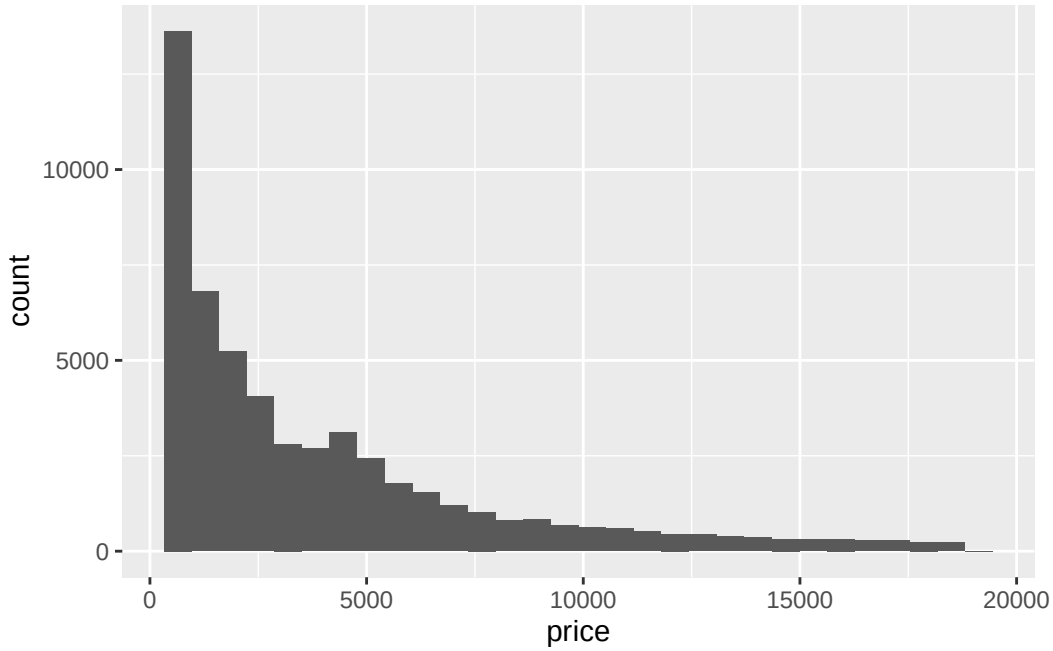
Dağılım grafikleri, veri setinin dağılımını görsel olarak temsil etmek için kullanılan grafik türleridir. Bu grafikler, veri noktalarının, değerlerinin veya gözlemlerinin nasıl dağıldığını incelemek ve veri setindeki desenleri, eğilimleri ve aykırı değerleri anlamak için kullanılır. En yaygın olanı histogram grafikleridir.

Histogram, veri setinin sayısal dağılımını gösteren bir grafiktir. Veri aralığı belli bir aralığa bölen çubuklardan oluşur ve her çubuk, bu aralıktaki veri noktalarının sayısını temsil eder. Histogramlar genellikle sürekli verilerin dağılımını göstermek için kullanılır.

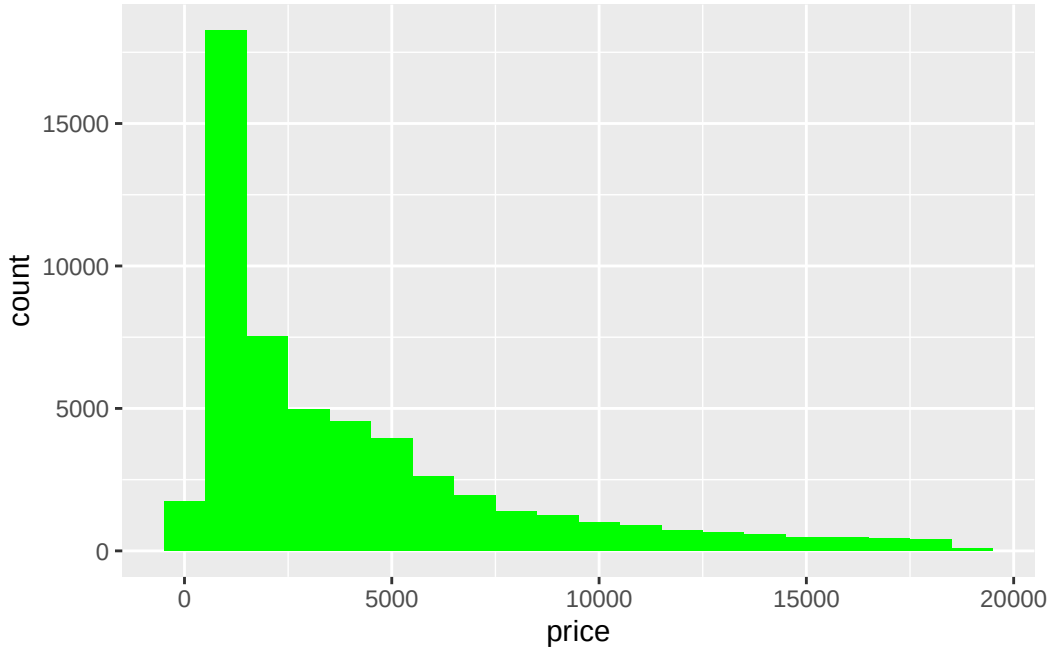
Bunun dışında boxplot (kutu) grafikleri de dağılımı görselleştirmek için kullanılmaktadır. Boxplot, veri setinin beş özet istatistiği (minimum, ilk çeyrek, medyan, üçüncü çeyrek, maksimum) kullanarak veri dağılımını temsil eder. Bu grafik, aykırı değerleri tanımlamak ve merkezi eğilim ile dağılımın yayılmasını görsel olarak incelemek için kullanılır.

```
ggplot(diamonds, aes(price)) +
  geom_histogram()
```

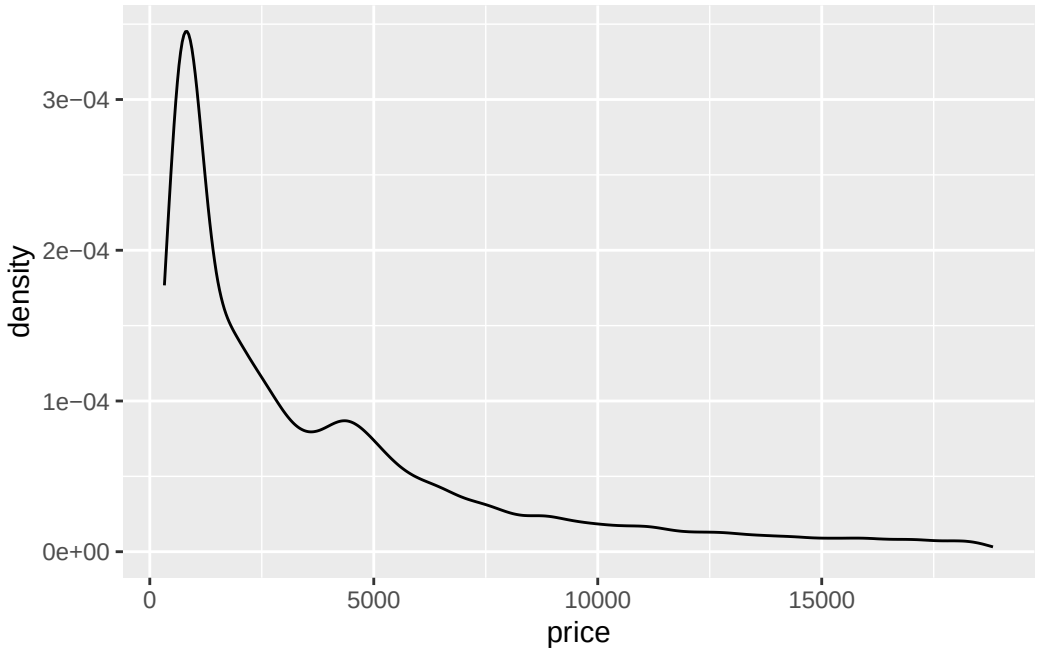
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



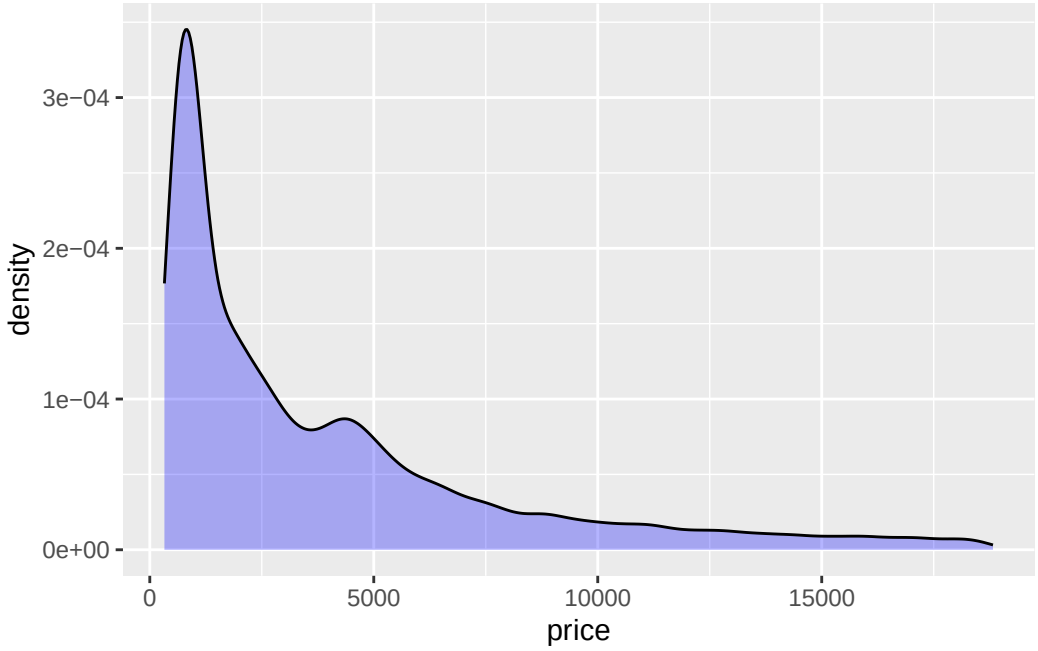

```
ggplot(diamonds, aes(price)) +
  geom_histogram(binwidth = 1000,fill = "green")
```




```
ggplot(diamonds, aes(price)) +
  geom_density()
```



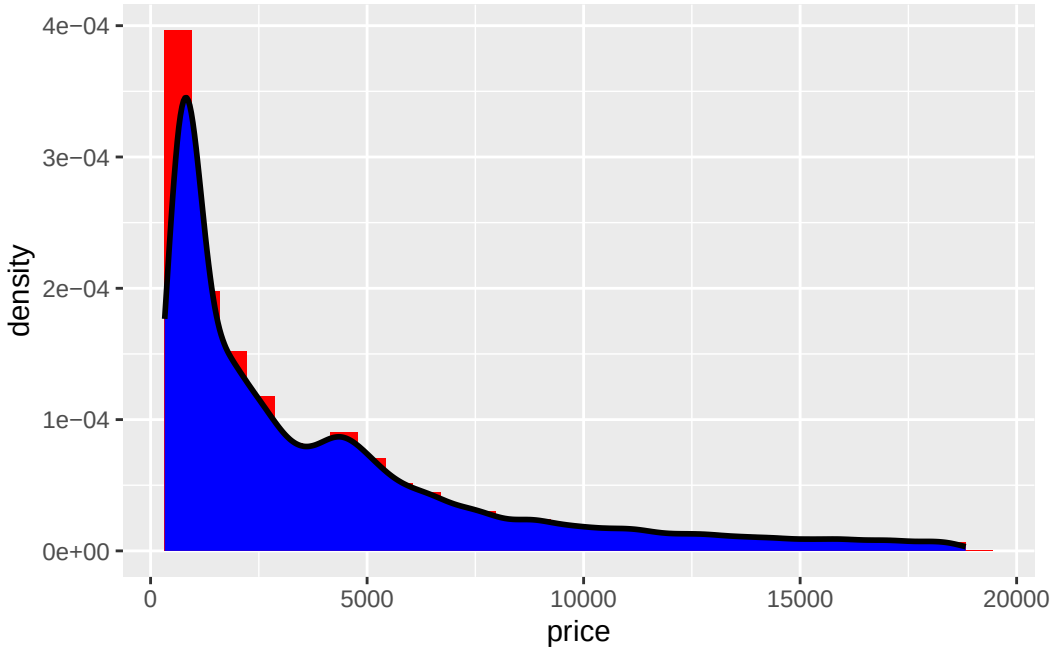
```
ggplot(diamonds, aes(price)) +
  geom_density(alpha = .3, fill = "blue")
```




```
ggplot(diamonds, aes(price)) +
  geom_histogram(aes(y = ..density..),fill = "red") +
  geom_density(size=1,fill = "blue")
```

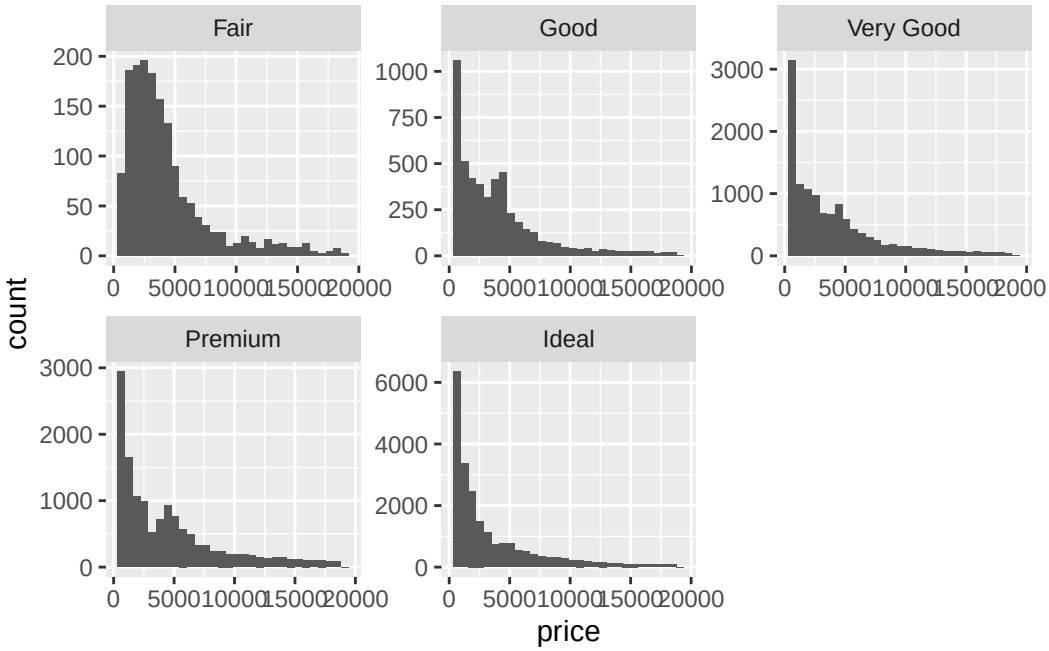
```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



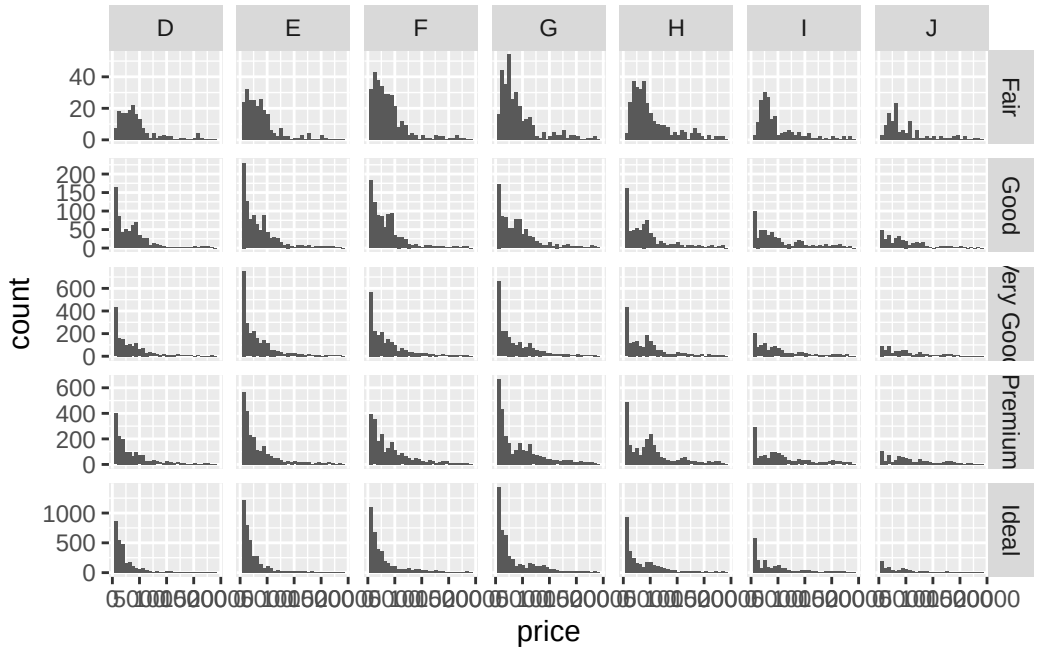

```
ggplot(diamonds, aes(price)) +
  geom_histogram() +
  facet_wrap( ~ cut ,scales = "free" )
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

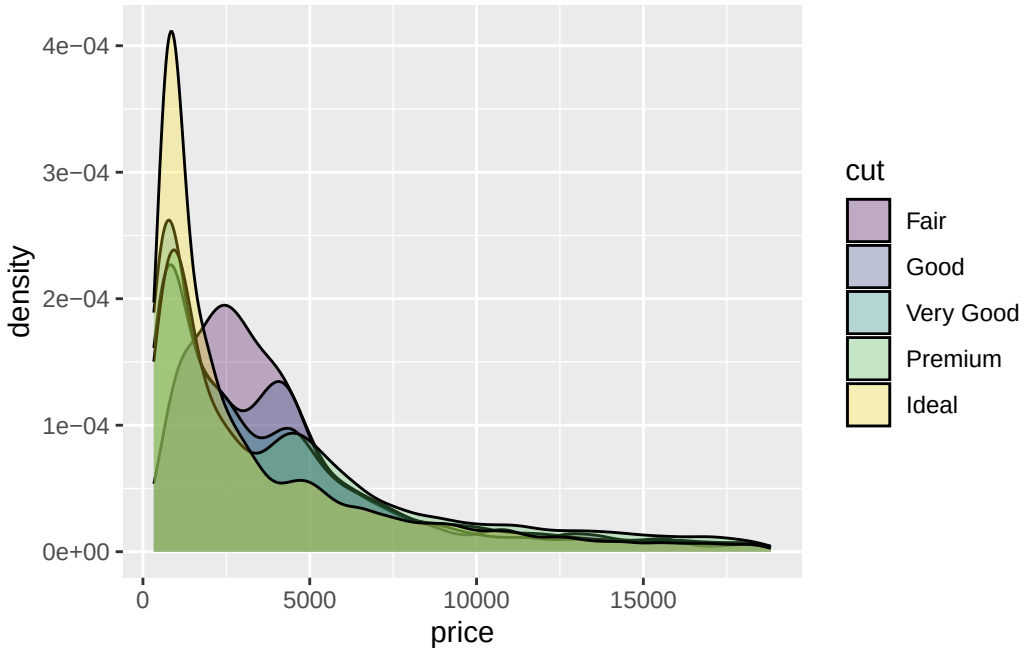



```
ggplot(diamonds, aes(price)) +
  geom_histogram() +
  facet_grid(cut ~ color,scales = "free" )
```

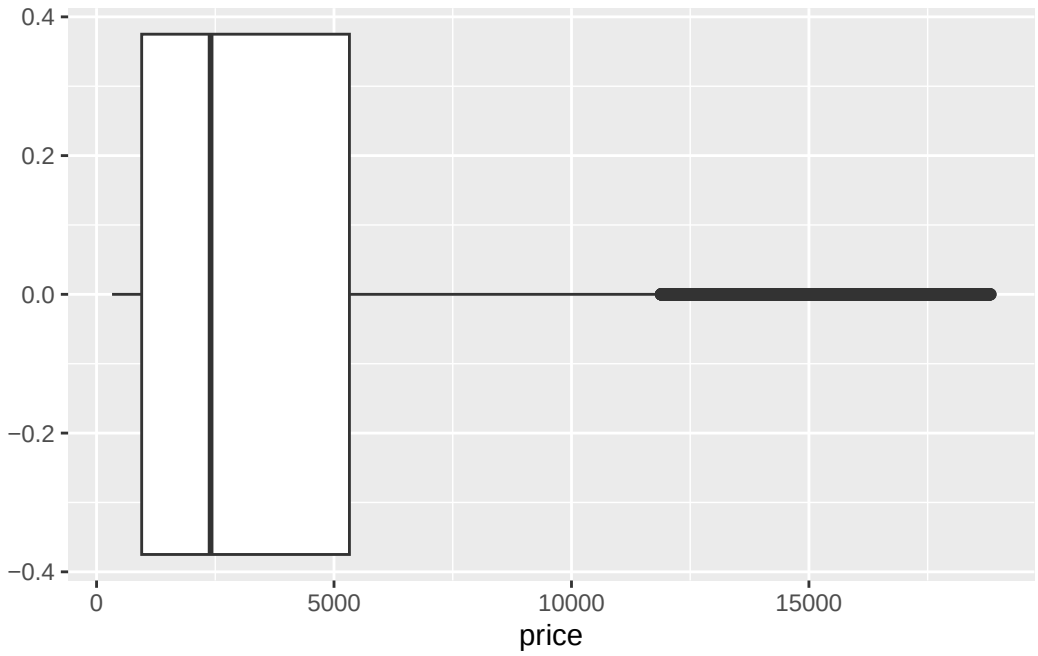
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



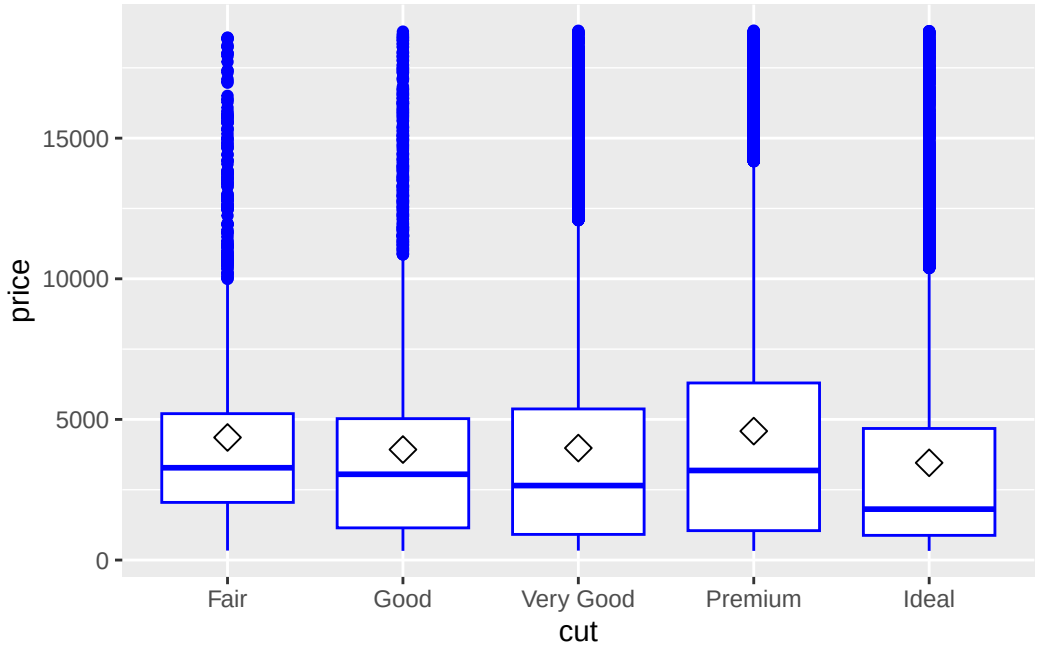

```
ggplot(diamonds, aes(x=price,fill=cut)) +
  geom_density(alpha=.3)
```



```r
# boxplot
ggplot(diamonds, aes(x=price)) +
  geom_boxplot()
```




```r
# boxplot'a ortalama eklemek
ggplot(diamonds, aes(x=cut,y=price)) +
  geom_boxplot(color="blue")+
  stat_summary(fun = "mean", geom = "point", shape = 5, size = 3)
```



Grafiklerin Kaydedilmesi

Grafik oluşturulduktan sonra, grafik objesini bir değişkende saklayabilirsiniz (aşağıdaki örnekte "grafik" adını kullandık). Grafik objesini bir değişkende sakladıktan sonra, **`ggsave()`** fonksiyonunu kullanarak grafik dosyasını kaydedebilirsiniz. Grafikleri ayrıca RStudio penceresinin sağ alt kısmında yer alan **Plots** sekmesindeki **`Export`** ile kayıt altına alabilirsiniz.

```
grafik <- economics %>%
  mutate(uemploy_mom=unemploy/lag(unemploy ) * 100 - 100,
         growth=ifelse(uemploy_mom>0,"pozitif","negatif")) %>%
  na.omit() %>%
  filter(lubridate::year(date)>=2010) %>%
  ggplot(aes(x=date,y=uemploy_mom,fill=growth))+
  geom_col() +
  theme(legend.position = "none") +
  labs(y="Aylık Değişim",
       title="Yıllara göre Aylık İstihdam Değişimi (2010-2015)")

ggsave("grafik1.png", grafik, width = 20, height = 8, units = "cm")
ggsave("grafik1.png", grafik,width = 20, height = 8, unit = "cm", dpi = 300)
```

Veri Ön İşleme

Veri ön işleme; istatistiksel modeller kurulmadan önce veri seti üzerinde yapılan bir takım düzeltme, eksik veriyi tamamlama, tekrarlanan verileri kaldırma, dönüştürme, bütünleştirme, temizleme, normalleştirme, boyut indirgeme vb. işlemlerdir. Bu aşamada ister istemez veri üzerinde bilgi keşfi yapılmış olur. Veri önişleme istatistiksel bir modelleme sürecinin büyük kısmını oluşturmaktadır. Kesin bir rakam olmamakla birlikte modelleme sürecinin yarısından fazlasının bu aşamada harcandığını ifade edebiliriz. Veri ön işleme temel anlamda 4 aşamadan oluşmaktadır. Bunlar sırasıyla şu şekildedir:

1. **Veri Temizleme :** Eksik verilerin tamamlanması, aykırı değerlerin teşhis edilmesi ve verilerdeki tutarsızlıkların giderilmesi gibi işlemler yapılmaktadır.
2. **Veri Birleştirme:** Farklı farklı veri tabanlarında bulunan veri setlerinin tek bir yerde toplanması aşamasının düzenli bir şekilde yürütülmesi sağlanır.
3. **Veri Dönüştürme :** Bu aşamada veriler, modelleme için uygun formlara dönüştürülürler. Veri dönüştürme; düzeltme, birleştirme, genelleştirme ve normalleştirme gibi değişik işlemlerden biri veya bir kaçını içerebilir. Veri normalleştirme , min-max dönüşümü, z standartlaştırması gibi yöntemler en sık kullanılan veri dönüştürme işlemlerinden bazılarıdır.
4. **Veri İndirgeme :** Daha küçük hacimli olarak veri kümesinin indirgenmiş bir örneğinin elde edilmesi amacıyla uygulanır. Bu sayede elde edilen indirgenmiş veri kümesine modelleme teknikleri uygulanarak daha etkin sonuçlar elde edilebilir. Veri Birleştirme (Data Aggregation), Boyut indirgeme (Dimension Reduction), Veri Sıkıştırma (Data Compression), Kesikli hale getirme (Discretization), Özellik Seçimi (Feature Selection) sık kullanılan veri indirgeme işlemlerindendir.

Bu dokümanda eksik veriler (missing values), aykırı değerler (outliers) ve veri normalleştirme işlemleri R uygulamları ile anlatılacaktır.

Eksik Veriler

Eksik veriler (kayıp gözlem), veri toplamada kaçınılmaz bir durumdur ve üzerinde dikkatle durulmalıdır. Sistematik bir kayıp gözlem durumu yoksa ortada ciddi bir sorun yoktur. Ama rastgele olmayan bir hata varsa tüm kitleye dair yanlılık olacağı için bu durum göz ardı edilemez.

```r
df <- data.frame(weight = c(rnorm(15, 70, 10), rep(NA, 5)),
                 height = c(rnorm(17, 165, 20), rep(NA, 3)))

set.seed(12345)
rows <- sample(nrow(df))
df2 <- df[rows, ]

# eksik verilerin sorgulanması

is.na(df2) # sorgulanma
```

```
   weight height
14  FALSE  FALSE
19   TRUE   TRUE
16   TRUE  FALSE
11  FALSE  FALSE
18   TRUE   TRUE
8   FALSE  FALSE
2   FALSE  FALSE
6   FALSE  FALSE
17   TRUE  FALSE
13  FALSE  FALSE
7   FALSE  FALSE
1   FALSE  FALSE
15  FALSE  FALSE
10  FALSE  FALSE
12  FALSE  FALSE
9   FALSE  FALSE
4   FALSE  FALSE
20   TRUE   TRUE
3   FALSE  FALSE
5   FALSE  FALSE
```

```r
which(is.na(df2)) #konum
```

```
[1]  2  3  5  9 18 22 25 38
```

```r
sum(is.na(df2)) # toplam eksik veri sayısı
```

```
[1] 8
```

```r
colSums(is.na(df2)) # değişken düzeyinde eksik veri sayısı
```

```
weight height
     5      3
```

```r
df2[!complete.cases(df2), ] #en az bir tane eksik olan satırlar
```

```
   weight   height
19     NA       NA
16     NA 169.3527
18     NA       NA
17     NA 204.5995
20     NA       NA
```

```r
df2[complete.cases(df2), ]$weight
```

```
 [1] 66.37812 58.37977 60.62641 62.76482 72.92068 78.23465 76.05072 67.90822
 [9] 73.18265 65.54547 82.15976 78.49235 56.25909 68.37293 61.54072
```

```r
# eksik veriden tamamen kurtulma
na.omit(df2)
```

```
     weight   height
14 66.37812 170.1149
11 58.37977 158.7822
8  60.62641 132.3578
2  62.76482 152.2575
6  72.92068 128.8460
13 78.23465 174.2841
7  76.05072 177.9670
1  67.90822 168.6864
15 73.18265 190.9497
10 65.54547 166.6571
12 82.15976 169.2619
9  78.49235 138.3663
4  56.25909 155.3241
3  68.37293 175.6937
5  61.54072 166.3402
```

```
complete.cases(df2)
```

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

```
df2[complete.cases(df2), ] # dolu olanlar satırlar
```

```
     weight   height
14 66.37812 170.1149
11 58.37977 158.7822
8  60.62641 132.3578
2  62.76482 152.2575
6  72.92068 128.8460
13 78.23465 174.2841
7  76.05072 177.9670
1  67.90822 168.6864
15 73.18265 190.9497
10 65.54547 166.6571
12 82.15976 169.2619
9  78.49235 138.3663
4  56.25909 155.3241
3  68.37293 175.6937
5  61.54072 166.3402
```

```
df2[complete.cases(df2), ]$weight # değişken bazında dolu olan satırlar
```

```
 [1] 66.37812 58.37977 60.62641 62.76482 72.92068 78.23465 76.05072 67.90822
 [9] 73.18265 65.54547 82.15976 78.49235 56.25909 68.37293 61.54072
```

İmputasyon

İmputasyon terimi, eksik verilerin yerine konulması veya doldurulması işlemine atıfta bulunur. Eksik veriler, bir veri setinde belirli gözlemler veya değişkenler için eksik veya bilinmeyen değerler içeren durumlardır. İstatistiksel analiz yaparken eksik verilerle başa çıkmak önemlidir çünkü eksik veriler, sonuçları yanıltabilir veya analizleri etkileyebilir.

İmputasyon, eksik verileri doldurmak veya tahmin etmek için kullanılan çeşitli istatistiksel yöntemleri ifade eder. İmputasyon işlemi, eksik verileri analizde kullanılabilir hale getirmek

amacıyla yapılır. İmputasyon yöntemleri, veri setinin yapısına ve eksik verilerin nedenlerine bağlı olarak değişebilir. İşte bazı yaygın imputasyon yöntemleri:

1. Ortalama Değer İmputasyonu: Eksik veriler, değişkenin ortalama değeri ile doldurulabilir. Bu yöntem, eksik verilerin diğer gözlemlerdeki ortalama değerlere benzer olduğu varsayımına dayanır.
2. Medyan Değer İmputasyonu: Eksik veriler, değişkenin medyan değeri ile doldurulabilir. Medyan, verilerdeki aşırı değerlerden etkilenmeyeceği için ortalama değere göre daha dayanıklı bir seçenektir.
3. En Yakın Komşu İmputasyonu: Eksik veriler, benzer diğer gözlemlerin değerleri ile doldurulabilir. Bu yöntemde, eksik veriye sahip olan gözlem, diğer gözlemlerin benzerliklerine göre doldurulur.
4. Regresyon İmputasyonu: Eksik veri içeren bir değişken, diğer değişkenlerle ilişkilendirilerek tahmin edilebilir. Bu yöntem, eksik verinin diğer değişkenlerle ilişkisini kullanarak doldurur.
5. EM (Expectation-Maximization) Algoritması: EM algoritması, eksik veri problemini çözmek için kullanılan bir iteratif istatistiksel yöntemdir. Bu yöntem, eksik verilerin olasılık dağılımlarını tahmin etmek için kullanılır.

İmputasyon yöntemi, veri setinin özelliklerine, eksik verilerin miktarına ve verilerin doğasına bağlı olarak seçilir. Her yöntemin avantajları ve dezavantajları vardır, bu nedenle doğru yöntemi seçmek, analizin doğruluğunu ve güvenilirliğini etkileyebilir. İmputasyonun amacı, eksik verilerin doğru ve güvenilir bir şekilde doldurulmasıdır, böylece analiz sonuçları daha kesin ve anlamlı olur.

```
# eksik verilere basit değer atama
df2$weight2 <- ifelse(is.na(df2$weight),mean(df2$weight, na.rm = TRUE),df2$weight)
df2
```

```
     weight   height  weight2
14 66.37812 170.1149 66.37812
19       NA       NA 68.58776
16       NA 169.3527 68.58776
11 58.37977 158.7822 58.37977
18       NA       NA 68.58776
8  60.62641 132.3578 60.62641
2  62.76482 152.2575 62.76482
6  72.92068 128.8460 72.92068
17       NA 204.5995 68.58776
13 78.23465 174.2841 78.23465
7  76.05072 177.9670 76.05072
```

```
1  67.90822 168.6864 67.90822
15 73.18265 190.9497 73.18265
10 65.54547 166.6571 65.54547
12 82.15976 169.2619 82.15976
9  78.49235 138.3663 78.49235
4  56.25909 155.3241 56.25909
20       NA       NA 68.58776
3  68.37293 175.6937 68.37293
5  61.54072 166.3402 61.54072
```

```
# tek seferde bütün kolonlardaki eksik verileri ortamala ile doldurmak için
sapply(df2, function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x ))
```

```
       weight   height  weight2
 [1,] 66.37812 170.1149 66.37812
 [2,] 68.58776 164.6965 68.58776
 [3,] 68.58776 169.3527 68.58776
 [4,] 58.37977 158.7822 58.37977
 [5,] 68.58776 164.6965 68.58776
 [6,] 60.62641 132.3578 60.62641
 [7,] 62.76482 152.2575 62.76482
 [8,] 72.92068 128.8460 72.92068
 [9,] 68.58776 204.5995 68.58776
[10,] 78.23465 174.2841 78.23465
[11,] 76.05072 177.9670 76.05072
[12,] 67.90822 168.6864 67.90822
[13,] 73.18265 190.9497 73.18265
[14,] 65.54547 166.6571 65.54547
[15,] 82.15976 169.2619 82.15976
[16,] 78.49235 138.3663 78.49235
[17,] 56.25909 155.3241 56.25909
[18,] 68.58776 164.6965 68.58776
[19,] 68.37293 175.6937 68.37293
[20,] 61.54072 166.3402 61.54072
```

```
library(zoo)
sapply(df2, function(x) ifelse(is.na(x), na.locf(x), x )) # carry forward
```

```
       weight   height  weight2
 [1,] 66.37812 170.1149 66.37812
```

```
 [2,] 66.37812 170.1149 68.58776
 [3,] 66.37812 169.3527 68.58776
 [4,] 58.37977 158.7822 58.37977
 [5,] 58.37977 158.7822 68.58776
 [6,] 60.62641 132.3578 60.62641
 [7,] 62.76482 152.2575 62.76482
 [8,] 72.92068 128.8460 72.92068
 [9,] 72.92068 204.5995 68.58776
[10,] 78.23465 174.2841 78.23465
[11,] 76.05072 177.9670 76.05072
[12,] 67.90822 168.6864 67.90822
[13,] 73.18265 190.9497 73.18265
[14,] 65.54547 166.6571 65.54547
[15,] 82.15976 169.2619 82.15976
[16,] 78.49235 138.3663 78.49235
[17,] 56.25909 155.3241 56.25909
[18,] 56.25909 155.3241 68.58776
[19,] 68.37293 175.6937 68.37293
[20,] 61.54072 166.3402 61.54072
```

```
sapply(df2, function(x) ifelse(is.na(x), na.locf(x,fromlast=TRUE), x ))
```

```
       weight   height  weight2
 [1,] 66.37812 170.1149 66.37812
 [2,] 66.37812 170.1149 68.58776
 [3,] 66.37812 169.3527 68.58776
 [4,] 58.37977 158.7822 58.37977
 [5,] 58.37977 158.7822 68.58776
 [6,] 60.62641 132.3578 60.62641
 [7,] 62.76482 152.2575 62.76482
 [8,] 72.92068 128.8460 72.92068
 [9,] 72.92068 204.5995 68.58776
[10,] 78.23465 174.2841 78.23465
[11,] 76.05072 177.9670 76.05072
[12,] 67.90822 168.6864 67.90822
[13,] 73.18265 190.9497 73.18265
[14,] 65.54547 166.6571 65.54547
[15,] 82.15976 169.2619 82.15976
[16,] 78.49235 138.3663 78.49235
[17,] 56.25909 155.3241 56.25909
[18,] 56.25909 155.3241 68.58776
```

```
[19,] 68.37293 175.6937 68.37293
[20,] 61.54072 166.3402 61.54072
```

```
sapply(df2, function(x) ifelse(is.na(x), na.approx(x), x )) # linear interpolation
```

```
       weight   height  weight2
 [1,] 66.37812 170.1149 66.37812
 [2,] 63.71200 169.7338 68.58776
 [3,] 61.04589 169.3527 68.58776
 [4,] 58.37977 158.7822 58.37977
 [5,] 59.50309 145.5700 68.58776
 [6,] 60.62641 132.3578 60.62641
 [7,] 62.76482 152.2575 62.76482
 [8,] 72.92068 128.8460 72.92068
 [9,] 75.57766 204.5995 68.58776
[10,] 78.23465 174.2841 78.23465
[11,] 76.05072 177.9670 76.05072
[12,] 67.90822 168.6864 67.90822
[13,] 73.18265 190.9497 73.18265
[14,] 65.54547 166.6571 65.54547
[15,] 82.15976 169.2619 82.15976
[16,] 78.49235 138.3663 78.49235
[17,] 56.25909 155.3241 56.25909
[18,] 62.31601 165.5089 68.58776
[19,] 68.37293 175.6937 68.37293
[20,] 61.54072 166.3402 61.54072
```

```
sapply(df2, function(x) ifelse(is.na(x), na.approx(x), x )) # cubic interpolation
```

```
       weight   height  weight2
 [1,] 66.37812 170.1149 66.37812
 [2,] 63.71200 169.7338 68.58776
 [3,] 61.04589 169.3527 68.58776
 [4,] 58.37977 158.7822 58.37977
 [5,] 59.50309 145.5700 68.58776
 [6,] 60.62641 132.3578 60.62641
 [7,] 62.76482 152.2575 62.76482
 [8,] 72.92068 128.8460 72.92068
 [9,] 75.57766 204.5995 68.58776
[10,] 78.23465 174.2841 78.23465
```

```
[11,] 76.05072 177.9670 76.05072
[12,] 67.90822 168.6864 67.90822
[13,] 73.18265 190.9497 73.18265
[14,] 65.54547 166.6571 65.54547
[15,] 82.15976 169.2619 82.15976
[16,] 78.49235 138.3663 78.49235
[17,] 56.25909 155.3241 56.25909
[18,] 62.31601 165.5089 68.58776
[19,] 68.37293 175.6937 68.37293
[20,] 61.54072 166.3402 61.54072
```

```r
# KNN (k-nearest neighbor) ile Değer Atama

library(DMwR2)
# airquality verisi
df_air <- tibble::as_tibble(airquality)
anyNA(df_air)
```

```
[1] TRUE
```

```r
# airquality verisindeki Wind değişkeninin bazı değerlerini NA yapalım
set.seed(1234)
row_num <- sample(1:nrow(airquality),5)
row_num # bu satırdaki değerlere NA atanacak
```

```
[1]  28  80 150 101 111
```

```r
airquality_2 <- airquality
airquality_2[row_num,"Wind"] <- NA
airquality_2[row_num,"Wind"]
```

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

```r
head(airquality_2,20)
```

```
   Ozone Solar.R Wind Temp Month Day
1     41     190  7.4   67     5   1
2     36     118  8.0   72     5   2
3     12     149 12.6   74     5   3
4     18     313 11.5   62     5   4
5     NA      NA 14.3   56     5   5
6     28      NA 14.9   66     5   6
7     23     299  8.6   65     5   7
8     19      99 13.8   59     5   8
9      8      19 20.1   61     5   9
10    NA     194  8.6   69     5  10
11     7      NA  6.9   74     5  11
12    16     256  9.7   69     5  12
13    11     290  9.2   66     5  13
14    14     274 10.9   68     5  14
15    18      65 13.2   58     5  15
16    14     334 11.5   64     5  16
17    34     307 12.0   66     5  17
18     6      78 18.4   57     5  18
19    30     322 11.5   68     5  19
20    11      44  9.7   62     5  20
```

```r
# k parametresi, verilen bir noktaya en yakın komşuların sayısıdır.
# Örneğin: k=5 olsun. Bu durumda mesafeye (öklit) göre en yakın 5 komşu belirlenir
# ve mesafenin ağırlıklı ortalaması hesaplanır.
# ağırlıklandırma, her komşuya 1 / d ağırlığının verilmesini içerir.
# burada d komşuya olan uzaklıktır.
knn_df_air <- knnImputation(airquality_2, k = 5) # k komşu sayısı

result <- data.frame(row=row_num,
                     orig=airquality[row_num,"Wind"],
                     knn=knn_df_air[row_num,"Wind"])
result
```

```
  row orig       knn
1  28 12.0 10.079819
2  80  5.1  8.765250
3 150 13.2  9.914454
4 101  8.0  6.807361
5 111 10.9 11.237192
```

```r
mean(result$orig-result$knn)
```

```
[1] 0.4791848
```

> Tavsiye
>
> Eksik verilerin analiz edilmesi ve imputasyon konusunda R içerisinde çeşitli kütühaneler bulunmaktadır. Bunlardan en çok bilinenleri **mice, VIM, missForest, imputation, mi, Amelia** ve **Hmisc** paketleridir.

Aykırı Değer Analizi

Aykırı değer, diğer gözlemlerden uzak olan, yani diğer veri noktalarından önemli ölçüde farklı olan bir veri noktası olan bir değer veya gözlemdir. Bu dokümanda, tanımlayıcı istatistikler (minimum, maksimum, histogram, kutu grafiği ve yüzdelikler dahil) gibi basit teknikler ve Z-Skoru ile aykırı değer analizi anlatılacaktır.

Minumum ve Maximum

```r
library(ggplot2)

# mpg verisindeki hwy değişkeni üzerinden inceleyelim
summary(mpg$hwy)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.00   18.00   24.00   23.44   27.00   44.00
```

```r
min(mpg$hwy)
```

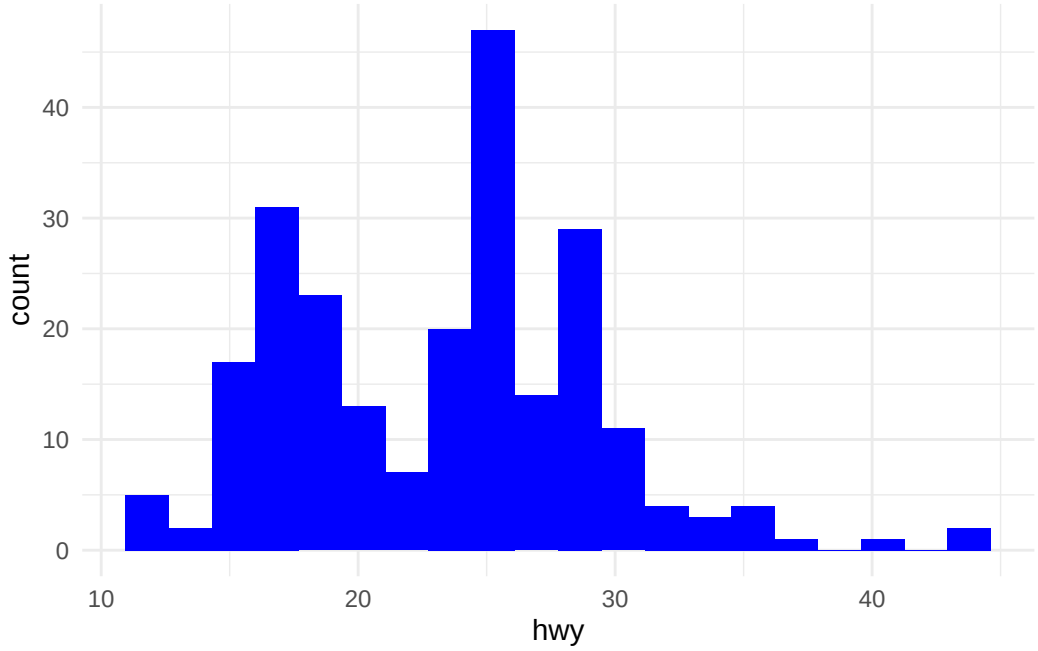
```
[1] 12
```

```r
max(mpg$hwy)
```

```
[1] 44
```

Histogram

```
ggplot(mpg) +
  aes(x = hwy) +
  geom_histogram(bins = 20, fill = "blue") +
  theme_minimal()
```

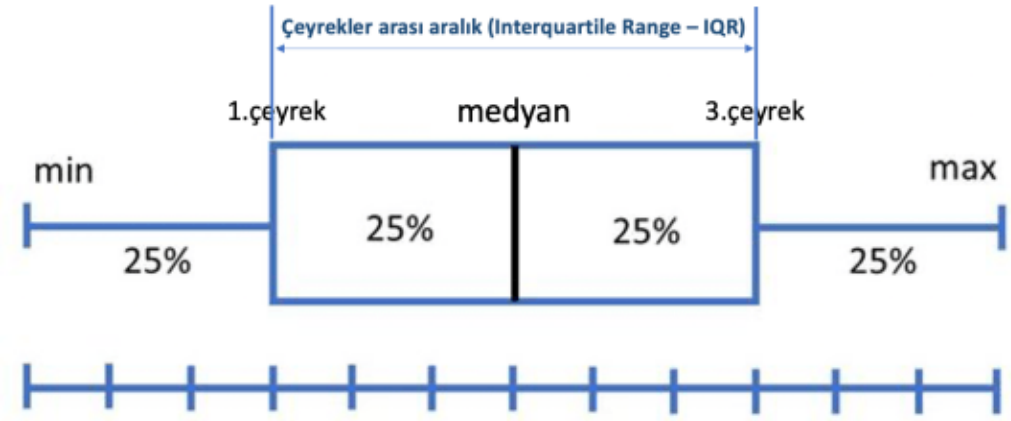



```
# grafiğiin sağ tarafında kalan gözlemler şüpheli görünüyor.
```

Boxplot

Boxplot, beş konum ölçüsü kullanarak verilerin grafiksel bir sunumunu verir: en küçük değer (min), birinci çeyreklik (Q_1) , medyan, üçüncü çeyreklik (Q_3) en büyük değer. Kutunun farklı bölümleri arasındaki boşluk, verilerdeki dağılım (yayılma) ve çarpıklık derecesini gösterir. Bir boxplot grafiği, çeyrekler arası aralık (IQR) kriteri kullanılarak şüpheli bir aykırı değer olarak sınıflandırılan herhangi bir gözlemi görüntüleyerek nicel bir değişkeni görselleştirmeye yardımcı olur.

$I = [Q_1 - 1.5 * IQR; Q_3 + 1.5 * IQR]$



IQR ise üçüncü ve birinci çeyrek arasındaki farktır. R içerisindeki `IQR()` fonksiyonu bu amaçla kullanılabilir.

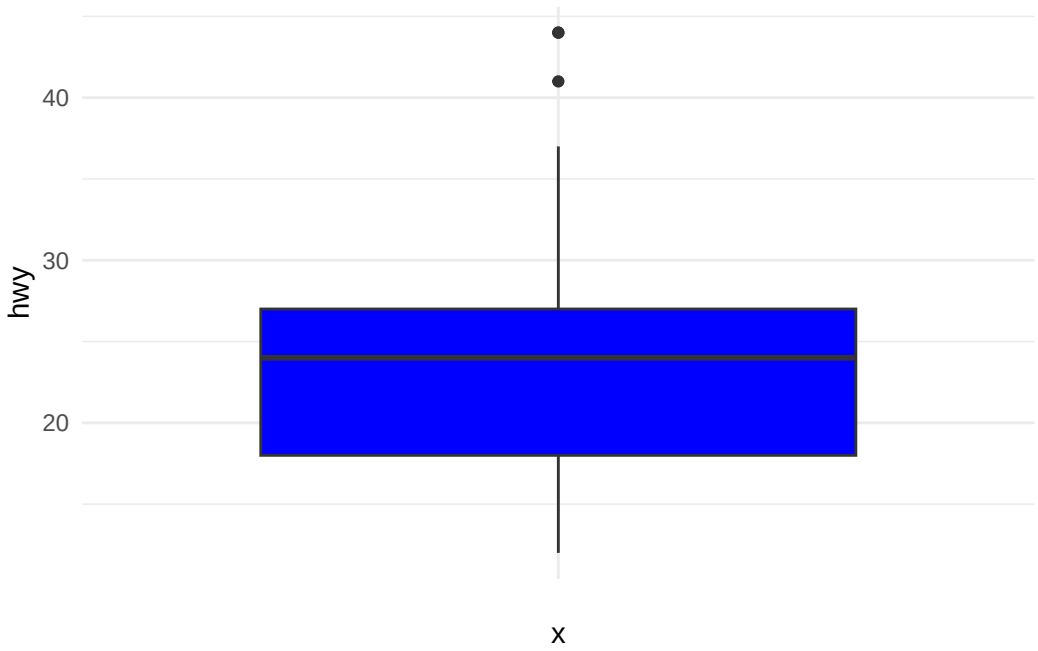
```
# temel istatistiklere erişim
summary(mpg$hwy)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.00   18.00   24.00   23.44   27.00   44.00
```

```
fivenum(mpg$hwy)
```

```
[1] 12 18 24 27 44
```

```
ggplot(mpg) +
  aes(x = "", y = hwy) +
  geom_boxplot(fill = "blue") +
  theme_minimal()
```




```
# outlier değerlerine erişim
boxplot.stats(mpg$hwy)$out
```

```
[1] 44 44 41
```

```
# outier olarak görülen değerlerin konumları
hwy_out <- boxplot.stats(mpg$hwy)$out
hwy_out_sira <- which(mpg$hwy %in% c(hwy_out))
hwy_out_sira
```

```
[1] 213 222 223
```

```
# outlier olarak görülen satırlar
mpg[hwy_out_sira, ]
```

```
# A tibble: 3 x 11
  manufacturer model      displ  year   cyl trans  drv     cty   hwy fl    class
  <chr>        <chr>      <dbl> <int> <int> <chr>  <chr> <int> <int> <chr> <chr>
1 volkswagen   jetta        1.9  1999     4 manua~ f        33    44 d     comp~
2 volkswagen   new beetle   1.9  1999     4 manua~ f        35    44 d     subc~
3 volkswagen   new beetle   1.9  1999     4 auto(~ f        29    41 d     subc~
```

Yüzdelikler (Percentiles)

Bu aykırı değer tespiti yöntemi, yüzdelik dilimlere dayalıdır. Yüzdelikler yöntemiyle, 2,5 ve 97,5 yüzdelik dilimlerin oluşturduğu aralığın dışında kalan tüm gözlemler potansiyel aykırı değerler olarak kabul edilecektir. Aralığı oluşturmak için 1 ve 99 veya 5 ve 95 yüzdelikler gibi diğer yüzdelikler de düşünülebilir.

```
alt_sinir <- quantile(mpg$hwy, 0.025)
alt_sinir
```

```
2.5%
  14
```

```
ust_sinir <- quantile(mpg$hwy, 0.975)
ust_sinir
```

```
 97.5%
35.175
```

```
# Bu yönteme göre, 14'ün altındaki ve 35.175'in üzerindeki tüm gözlemler,
# potansiyel aykırı değerler olarak kabul edilecektir.

outlier_sira <- which(mpg$hwy < alt_sinir | mpg$hwy > ust_sinir)
outlier_sira
```

```
 [1]  55  60  66  70 106 107 127 197 213 222 223
```

```
# Bu yönteme göre 11 adet outlier bulunmuştur.
mpg[outlier_sira,]
```

```
# A tibble: 11 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 dodge        dakota pi~   4.7  2008     8 auto~ 4         9    12 e     pick~
 2 dodge        durango 4~   4.7  2008     8 auto~ 4         9    12 e     suv
 3 dodge        ram 1500 ~   4.7  2008     8 auto~ 4         9    12 e     pick~
 4 dodge        ram 1500 ~   4.7  2008     8 manu~ 4         9    12 e     pick~
 5 honda        civic        1.8  2008     4 auto~ f        25    36 r     subc~
```

```
 6 honda        civic        1.8  2008     4 auto~ f        24    36 c     subc~
 7 jeep         grand che~   4.7  2008     8 auto~ 4         9    12 e     suv
 8 toyota       corolla      1.8  2008     4 manu~ f        28    37 r     comp~
 9 volkswagen   jetta        1.9  1999     4 manu~ f        33    44 d     comp~
10 volkswagen   new beetle   1.9  1999     4 manu~ f        35    44 d     subc~
11 volkswagen   new beetle   1.9  1999     4 auto~ f        29    41 d     subc~
```

```r
# Sınırları biraz daha küçültelim
alt_sinir <- quantile(mpg$hwy, 0.01)
ust_sinir <- quantile(mpg$hwy, 0.99)

outlier_sira <- which(mpg$hwy < alt_sinir | mpg$hwy > ust_sinir)

mpg[outlier_sira, ]
```

```
# A tibble: 3 x 11
  manufacturer model      displ  year   cyl trans  drv     cty   hwy fl    class
  <chr>        <chr>      <dbl> <int> <int> <chr>  <chr> <int> <int> <chr> <chr>
1 volkswagen   jetta        1.9  1999     4 manua~ f        33    44 d     comp~
2 volkswagen   new beetle   1.9  1999     4 manua~ f        35    44 d     subc~
3 volkswagen   new beetle   1.9  1999     4 auto(~ f        29    41 d     subc~
```

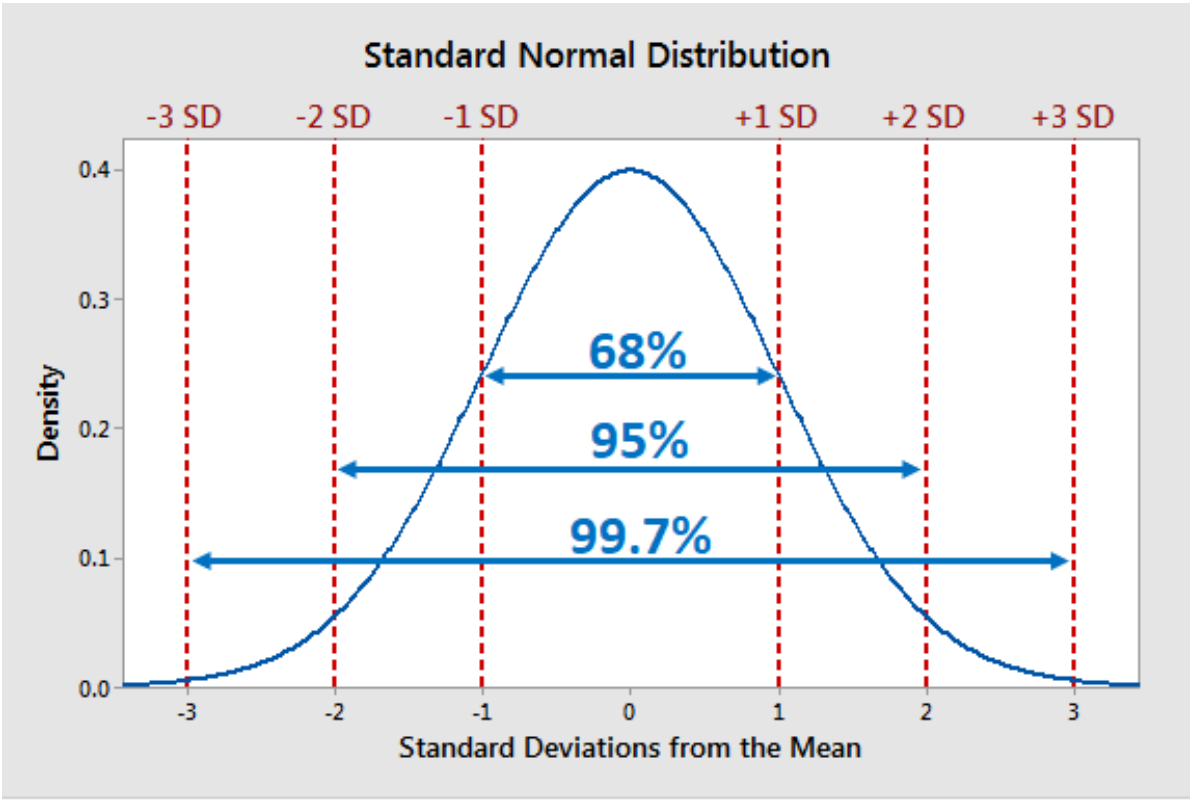
```r
# Buna göre IQR ile elde edildiği gibi 3 adet outlier bulundu.
```

Z-Skor Yöntemi

Aykırı değerlerin tespitinde ortalama ve standart sapmanın kulllanıldığı en bilinen yöntemlerdendir ve aşağıdaki şekilde hesaplanır.

$Z_i = \frac{(X_i - \mu)}{\sigma}$

```r
std_z <- function(x){

  z=(x-mean(x))/sd(x)
  return(z)
}

mpg$hwy_std <- std_z(mpg$hwy)
mpg[,c("hwy","hwy_std")]
```


Standard Normal Distribution
-3 SD
-2 SD
-1 SD
+1 SD
+2 SD
+3 SD
0.4
0.3
0.2
0.1
0.0
Density
68%
95%
99.7%
-3
-2
-1
0
1
2
3
Standard Deviations from the Mean

```
# A tibble: 234 x 2
     hwy hwy_std
   <int>   <dbl>
 1    29   0.934
 2    29   0.934
 3    31   1.27
 4    30   1.10
 5    26   0.430
 6    26   0.430
 7    27   0.598
 8    26   0.430
 9    25   0.262
10    28   0.766
# i 224 more rows
```

```
# -3 ve +3 sapma dışında kalanları aykırı değer olarak kabul ediyoruz.
outliers_zskor <- which(mpg$hwy_std < -3 | mpg$hwy_std > +3)
outliers_zskor
```

```
[1] 213 222
```

```
mpg[outliers_zskor,c() ]
```

```
# A tibble: 2 x 0
```

```
# bu yönteme göre 2 adet aykırı değer bulunmuştur.
```

Veri Normalleştirme

Değişkenler farklı ölçeklerde ölçüldüğünde, genellikle analize eşit katkıda bulunmazlar. Örneğin, bir değişkenin değerleri 0 ile 100.000 arasında ve başka bir değişkenin değerleri 0 ile 100 arasında değişiyorsa, daha büyük aralığa sahip değişkene analizde daha büyük bir ağırlık verilecektir. Değişkenleri normalleştirerek, her bir değişkenin analize eşit katkı sağladığından emin olabiliriz. Değişkenleri normalleştirmek için (veya ölçeklendirmek) genellikle min-max ya da z dönüşümü yöntemleri kullanılır.

```r
# min-max dönüşümleri

# 0 ile 1 arasi dönüşüm
std_0_1 <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

#-1 ile +1 arası dönüşüm
std_1_1 <- function(x) {
  ((x - mean(x)) / max(abs(x - mean(x))))
}

# a ile b arası dönüşüm
std_min_max <- function(x,a,b) {
  # a min değer
  # b max değer
  (a + ((x - min(x)) * (b - a)) / (max(x) - min(x)))
}

set.seed(12345)
dat <- data.frame(x = rnorm(20, 10, 3),
                  y = rnorm(20, 30, 8),
                  z = rnorm(20, 25, 5))
dat
```

```
          x        y        z
1  11.756586 36.23698 30.64255
2  12.128398 41.64628 13.09821
3   9.672090 24.84537 19.69867
4   8.639508 17.57490 29.68570
5  11.817662 17.21832 29.27226
6   4.546132 44.44078 32.30365
7  11.890296 26.14682 17.93451
8   9.171448 34.96304 27.83702
9   9.147521 34.89699 27.91594
10  7.242034 28.70151 18.46601
11  9.651257 36.49499 22.29807
12 15.451936 47.57467 34.73846
13 11.111884 46.39352 25.26795
14 11.560649 43.05957 26.75831
15  7.748404 32.03417 21.64512
16 12.450700 33.92951 26.38977
```

```
17  7.340927 27.40731 28.45586
18  9.005267 16.70360 29.11898
19 13.362138 44.14187 35.72533
20 10.896171 30.20641 13.26528
```

```r
summary(dat)
```

```
       x                y               z        
 Min.   : 4.546   Min.   :16.70   Min.   :13.10  
 1st Qu.: 8.914   1st Qu.:27.09   1st Qu.:21.16  
 Median :10.284   Median :34.41   Median :27.30  
 Mean   :10.230   Mean   :33.23   Mean   :25.53  
 3rd Qu.:11.836   3rd Qu.:42.00   3rd Qu.:29.38  
 Max.   :15.452   Max.   :47.57   Max.   :35.73  
```

```r
apply(dat, 2, std_0_1)
```

```
              x          y           z
 [1,] 0.6611575 0.63274053 0.775368144
 [2,] 0.6952505 0.80796300 0.000000000
 [3,] 0.4700211 0.26373477 0.291705877
 [4,] 0.3753393 0.02822392 0.733080320
 [5,] 0.6667578 0.01667340 0.714808256
 [6,] 0.0000000 0.89848463 0.848779748
 [7,] 0.6734179 0.30589231 0.213738973
 [8,] 0.4241150 0.59147416 0.651378062
 [9,] 0.4219211 0.58933460 0.654866001
[10,] 0.2471988 0.38864587 0.237228478
[11,] 0.4681108 0.64109819 0.406585628
[12,] 1.0000000 1.00000000 0.956385878
[13,] 0.6020419 0.96173940 0.537838847
[14,] 0.6431912 0.85374322 0.603705080
[15,] 0.2936301 0.49659993 0.377728555
[16,] 0.7248037 0.55799517 0.587417289
[17,] 0.2562668 0.34672297 0.678727553
[18,] 0.4088772 0.00000000 0.708033996
[19,] 0.8083774 0.88880212 1.000000000
[20,] 0.5822623 0.43739366 0.007383637
```

```
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
```

```
dat %>% mutate_all(std_0_1) %>% summary()
```

```
       x                y                z         
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.4005   1st Qu.:0.3365   1st Qu.:0.3562  
 Median :0.5261   Median :0.5737   Median :0.6275  
 Mean   :0.5211   Mean   :0.5354   Mean   :0.5492  
 3rd Qu.:0.6684   3rd Qu.:0.8194   3rd Qu.:0.7194  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

```
dat %>% mutate_all(std_1_1) %>% summary()
```

```
       x                    y                   z          
 Min.   :-1.000000   Min.   :-1.00000   Min.   :-1.0000  
 1st Qu.:-0.231502   1st Qu.:-0.37143   1st Qu.:-0.3514  
 Median : 0.009603   Median : 0.07154   Median : 0.1426  
 Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000  
 3rd Qu.: 0.282624   3rd Qu.: 0.53057   3rd Qu.: 0.3098  
 Max.   : 0.918881   Max.   : 0.86789   Max.   : 0.8207  
```

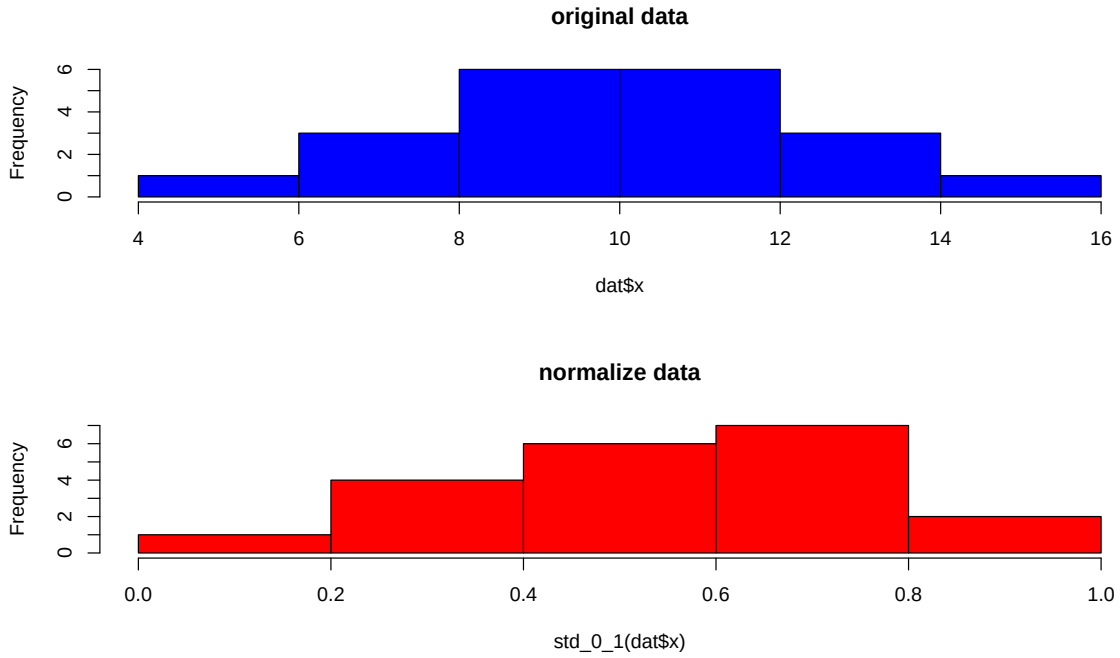
```
dat %>% mutate_all(std_min_max, a = -2, b = 2) %>% summary()
```

```
      x                   y                   z
Min.   :-2.00000   Min.   :-2.0000   Min.   :-2.0000
1st Qu.:-0.39803   1st Qu.:-0.6539   1st Qu.:-0.5751
Median : 0.10457   Median : 0.2947   Median : 0.5102
Mean   : 0.08455   Mean   : 0.1415   Mean   : 0.1970
3rd Qu.: 0.67369   3rd Qu.: 1.2776   3rd Qu.: 0.8775
Max.   : 2.00000   Max.   : 2.0000   Max.   : 2.0000
```

```
dat %>% mutate_all(std_z) %>% summary()
```

```
      x                   y                   z
Min.   :-2.27173   Min.   :-1.7088   Min.   :-1.9165
1st Qu.:-0.52591   1st Qu.:-0.6347   1st Qu.:-0.6735
Median : 0.02182   Median : 0.1223   Median : 0.2732
Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
3rd Qu.: 0.64204   3rd Qu.: 0.9067   3rd Qu.: 0.5937
Max.   : 2.08745   Max.   : 1.4831   Max.   : 1.5729
```

```
# Yapılan dönüşümler verinin dağılımını değiştirmemektedir.
par(mfrow=c(2,1))
hist(dat$x,main="original data",col="blue")
hist(std_0_1(dat$x),main="normalize data",col="red")
```


original data
Frequency
0
2
4
6
4
6
8
10
12
14
16
dat$x
normalize data
Frequency
0
2
4
6
0.0
0.2
0.4
0.6
0.8
1.0
std_0_1(dat$x)

R ile Temel İstatistik

İstatistik; amacın belirlenmesi, çalışmanın planlanması, verilerin toplanması, değerlendirilmesi ve karara varılması sürecini içeren bir bilim dalıdır. İstatistik bilimi içinde örneklemden elde edilen bilgileri kitlelere genelleme, tahminler yapma, değişkenler arasındaki ilişkileri ortaya çıkarma gibi konular yer almaktadır.

Uygulamalı istatistikler iki alana ayrılabilir: tanımlayıcı istatistikler ve çıkarımsal istatistikler. Tanımlayıcı istatistikler, tabloları, grafikleri ve özet ölçüleri kullanarak verileri düzenleme, görüntüleme ve tanımlama yöntemlerinden oluşur. Buna karşılık çıkarımsal istatistikler, bir popülasyon hakkında kararlar veya tahminler yapmak için örnek sonuçlarını kullanan yöntemlerden oluşur.

Tanımlayıcı istatistik, bir dizi değeri veya bir veri kümesini özetlemeyi, tanımlamayı ve sunmayı amaçlayan bir istatistik dalıdır. Tanımlayıcı istatistikler genellikle herhangi bir istatistiksel analizin ilk adımı ve önemli bir parçasıdır. Verilerin kalitesini kontrol etmeyi sağlar ve net bir genel bakışa sahip olarak verileri anlamaya yardımcı olur. Tanımlayıcı istatistikler, merkezi eğilim ölçüleri ve dağılım ölçüleri olmak üzere ikiye ayrılır.

Merkezi Eğilim Ölçüleri

Dağılımın konumu hakkında bilgi veren ölçümlerdir. Aritmetik ortalama, geometrik ortalama, harmonik ortalama, düzeltilmiş ortalama, ortanca, çeyrekler, yüzdelikler konum ölçülerine örnek olarak verilebilir.

Aritmetik Ortalama

- Günlük hayatta en sık kullanılan merkezi eğilim ölçüsüdür.
- Üzerinde inceleme yapılan veri setindeki elemanların toplanıp incelenen eleman sayısına bölünmesiyle elde edilir.
- Konum olarak verilerin en çok hangi değer etrafında toplandığının ya da yoğunlaştığının sayısal bir ölçüsüdür.
- Hem kitle hem de örneklem için hesaplanır.

- Dağılışların yerinin belirlenmesinde en çok kullanılan yer ölçüsü aritmetik ortalamadır; ve tek başına ortalama sözcüğünden aritmetik ortalama anlaşılır.
- Aritmetik ortalama bütün değerlerin ağırlığını eşit kabul ettiğinden dağılımı her zaman en iyi şekilde temsil etmeyebilir. Ayrıca aritmetik ortalama, veri kümesindeki aşırı değerlerden çok kolay etkilenir.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i$$

```
mean(airquality$Wind)
```

```
[1] 9.957516
```

```
mean(airquality$Ozone, na.rm = TRUE) # NA'ler kaldırılarak ortalama hesaplanır
```

```
[1] 42.12931
```

Geometrik Ortalama

- Periyodik artışlar veya azalmalar (değişim oranları) içeren enflasyon veya nüfus değişiklikleri gibi konuları incelerken, geometrik ortalama, incelenen tüm dönem boyunca ortalama değişikliği bulmak için daha uygundur.
- Eğer veriler sıfır ya da negatif değerler içeriyorsa geometrik ortalama hesaplanamaz.
- Geometrik ortalama, uç değerlerden aritmetik ortalamaya göre daha az etkilenmektedir.
- Geometrik Ortalama <= Aritmetik Ortalama

$$G.O. = \sqrt[n]{\prod_{i=1}^{n} X_i}$$

```
# R programında hazır geometrik ortalama fonksiyonu yoktur.
# 1. yol
geo_mean <- function(x){
  x <- na.omit(x)
  (prod(x))^(1/length(x))
}
```

```
round(geo_mean(airquality$Wind),3)
```

```
[1] 9.273
```

```
round(geo_mean(airquality$Ozone),3)
```

```
[1] 30.524
```

```
# 2. yol
library(psych)
round(geometric.mean(airquality$Wind),3)
```

```
[1] 9.273
```

```
round(geometric.mean(airquality$Ozone),3)
```

```
[1] 30.524
```

Medyan (Ortanca)

- Gözlem değerleri küçükten büyüğe sıralandığında ortada kalan gözlem değeridir.
- Bir seride yer alan gözlemlerin tümünün hesaba katılmadığı ortalamalardan biridir.
- Basit serilerde seri tek sayıda gözlemden oluşuyorsa serinin gözlem değerleri küçükten büyüğe sıralandığında tam ortada yer alan gözlem değeridir.
- Seri çift sayıda gözlemden oluşuyorsa ortada kalan iki gözlem değerinin aritmetik ortalaması medyandır.
- Medyan, ölçümlerin %50'sinin üzerinde, %50'sinin aşağısında yer aldığı merkezi değerdir.
- Dağılımdaki aşırı değerlerden etkilenmez.
- Aritmetik ortalamaya kıyasla daha tutarlı bir sonuç elde edilir.
- Her bir veri seti için bir tek medyan söz konusudur.
- Medyanın zayıf tarafı serideki bütün değerleri dikkate almaması sebebi ile matematik işlemlere elverişli değildir.

- Gözlem sayısı (n) tek ise , $\widetilde{X} = X_{\frac{n+1}{2}}$
- Gözlem sayısı (n) çift ise , $\widetilde{X} = \frac{X_{\frac{n}{2}} + X_{\frac{n+1}{2}}}{2}$

```
median(airquality$Wind)
```

```
[1] 9.7
```

```
median(airquality$Ozone,na.rm = TRUE)
```

```
[1] 31.5
```

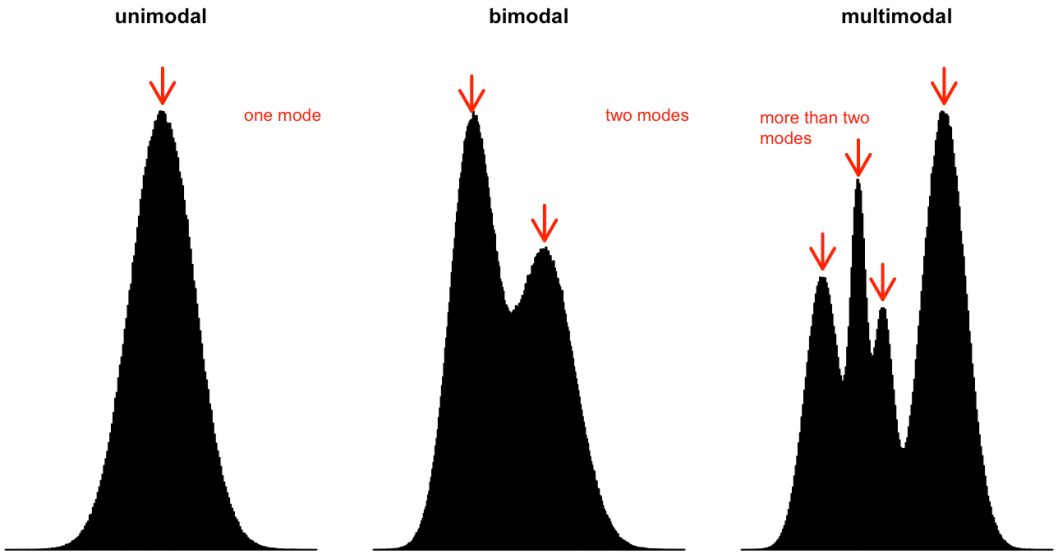
Mod (Tepe değeri)

- En sık ortaya çıkan (en yüksek frekanslı) ölçümdür.
- Dağılımdaki aşırı değerlerden etkilenmez
- Her dağılımda tepe değeri bulunmayabilir.
- Bazı dağılımlarda birden fazla tepe değeri bulunabilir.
- Tepe değeri aritmetik işlemler için elverişli değildir.
- Tüm veri değerlerini göz önünde bulundurmadığı için tutarlı olmayan bir merkezi eğilim ölçüsüdür.
- Gözlem sayısı az olduğunda tepe değer güvenilir bir ölçü değildir.

```
# R programında hazır mod fonksiyonu yoktur.

library(DescTools)
Mode(airquality$Wind)
```

```
[1] 11.5
attr(,"freq")
[1] 15
```

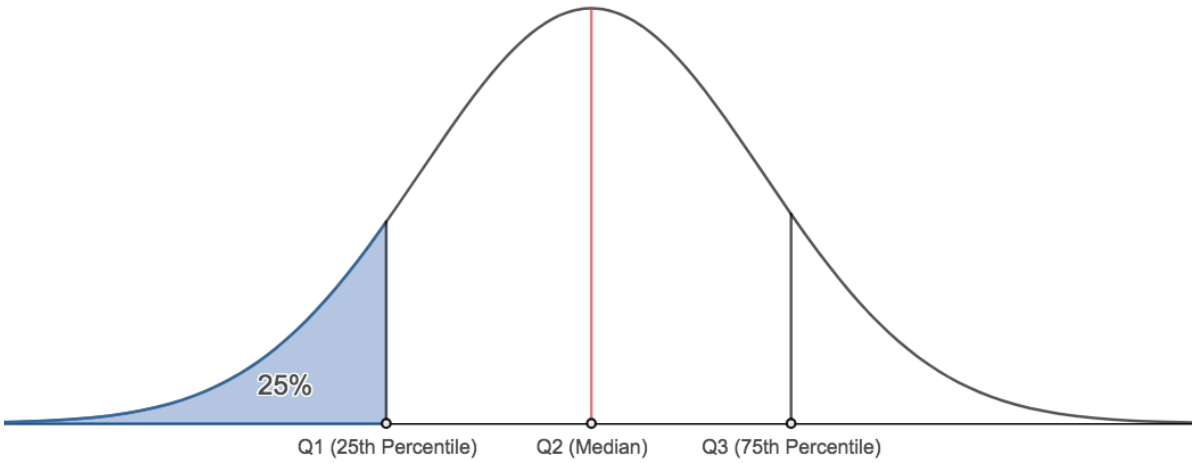
```
Mode(airquality$Solar.R,na.rm = TRUE)
```


unimodal
one mode
bimodal
two modes
multimodal
more than two
modes

```
[1] 238 259
attr(,"freq")
[1] 4
```

Çeyreklikler

- Birinci Bölen ilk yüzde 25 nci noktadır ve verinin ¼ kadarı birinci bölen içerisinde kalır.
- İkinci Bölen ilk yüzde 50 nci noktadır ve verinin yarısı bu noktanın altında kalır(½) aynı zamanda ikinci bölen medyan olarak ta bilinir.
- Üçüncü Bölen ilk yüzde 75 nci veri kümesidir ve bütün verinin ¾ kadarı bu noktanın altında kalır.
- Gözlem sayısı (n) tek ise , $Q_1 = X_{\frac{n+1}{4}}$
- Gözlem sayısı (n) çift ise , $Q_1 = \frac{X_{\frac{n}{4}} + X_{\frac{n}{4}+1}}{2}$
- Gözlem sayısı (n) tek ise , $Q_3 = X_{\frac{3(n+1)}{4}}$
- Gözlem sayısı (n) çift ise , $Q_3 = \frac{X_{\frac{3n}{4}} + X_{\frac{3n}{4}+1}}{2}$




```r
quantile(airquality$Wind,na.rm = TRUE)
```

```
  0%  25%  50%  75% 100% 
 1.7  7.4  9.7 11.5 20.7 
```

```r
median(airquality$Wind,na.rm = TRUE)
```

```
[1] 9.7
```

```r
quantile(airquality$Wind,na.rm = TRUE,probs = 0.75) #Q3
```

```
 75%
11.5
```

```r
quantile(airquality$Wind,na.rm = TRUE,probs = 0.25) #Q1
```

```
25%
7.4
```

```r
quantile(airquality$Wind,na.rm = TRUE,probs = c(0.20,0.50,0.80)) # %20,%50,%80
```

```
  20%   50%   80%
 6.90  9.70 12.96
```

```r
quantile(airquality$Solar.R,na.rm = TRUE)
```

```
  0%  25%  50%  75% 100%
 7.0 115.8 205.0 258.8 334.0
```

```r
median(airquality$Solar.R,na.rm = TRUE)
```

```
[1] 205
```

Dağılım Ölçüleri

Ortalama, medyan ve mod gibi merkezi eğilim ölçüleri, bir veri setinin dağılımının bütün resmini ortaya koymaz. Aynı ortalamaya sahip iki veri seti tamamen farklı yayılımlara sahip olabilir. Bir veri seti için gözlem değerleri arasındaki farklılık, diğer veri seti için olduğundan çok daha büyük veya daha küçük olabilir. Bu nedenle, ortalama, medyan veya mod tek başına genellikle bir veri kümesinin dağılımının şeklini ortaya çıkarmak için yeterli bir ölçü değildir. Bu yüzden veri değerleri arasındaki varyasyon hakkında bazı bilgiler sağlayabilecek bir ölçülere de ihtiyaç vardır. Bu ölçülere dağılım (yayılım) ölçüleri denir. Birlikte ele alınan merkezi eğilim ve dağılım ölçüleri, tek başına merkezi eğilim ölçülerinden ziyade bir veri setinin daha iyi bir resmini verir. Değişim aralığı, çeyrekler arası genişlik, varyans, standart sapma, basıklık, çarpıklık, min, max başlıca dağılım ölçüleri arasındadır.

Değişim Aralığı (Açıklık)

- Veri setindeki en büyük değer ile en küçük değer arasındaki farktır.
- En basit dağılım ölçüsü olmakla birlikte uç ve aykırı değerlerden etkilenmesi olumsuz yönüdür.
- Serinin sadece 2 gözlemine bağlı olarak hesaplanan bu ölçü değişkenliğin şekli hakkında çok fazla bilgi vermediğinden diğer değişkenlik ölçüleri kadar sık kullanılmaz.

$$D.A = max(X) - min(X)$$

```
# 1. yol
max(airquality$Ozone,na.rm = TRUE)-min(airquality$Ozone,na.rm = TRUE)
```

```
[1] 167
```

```
# 2. yol
range(airquality$Ozone,na.rm = TRUE)
```

```
[1]   1 168
```

```
range(airquality$Ozone,na.rm = TRUE)[2]-range(airquality$Ozone,na.rm = TRUE)[1]
```

```
[1] 167
```

Çeyrekler Arası Genişlik

- Dağılımdaki verilerin ortadaki % 50'sinin yer aldığı aralığı belirlemek için kullanılır.
- Aşırı uç değerlerden etkilenmez. Çünkü çeyreklikler arası genişlik dağılımdaki değerlerin merkezdeki %50'si ile ilgilenir.
- Çeyrekler arası bir genişlik, değerlerin büyük kısmının nerede olduğunu gösteren bir ölçüdür.
- Çeyrek Sapma 3. çeyrek ile 1. çeyrek arasındaki farktır.
- IQR (Interquartile Range) olarak ifade edilir.

$$IQR = Q_3 - Q1$$

```
# 1.yol
q3 <- quantile(airquality$Wind,na.rm = TRUE,probs = 0.75) #Q3
q1 <- quantile(airquality$Wind,na.rm = TRUE,probs = 0.25) #Q1
q3-q1
```

```
75%
4.1
```

```
# 2. yol
IQR(airquality$Wind,na.rm = TRUE)
```

```
[1] 4.1
```

Varyans ve Standart Sapma

Gözlem değerlerinin aritmetik ortalamadan sapmaları dikkate alınarak farklı değişkenlik ölçüleri geliştirilebilir. Ancak gözlemlerin aritmetik ortalamadan sapmalarının her zaman sıfıra eşittir. Bu sorunu ortadan kaldırmak için gözlemlerin aritmetik ortalamadan olan sapmalarının karelerinin toplamının gözlem sayısına oranı değişkenlik ölçüsü olarak yorumlanabilir. Bu ölçü varyans olarak adlandırılır.

- Bir dağılımda değerler aritmetik ortalamadan uzaklaştıkça dağılımın yaygınlığı artar.

- Varyansın karekökü standart sapmadır. Genel olarak, bir veri kümesi için standart sapmanın daha düşük bir değeri, o veri kümesinin değerlerinin ortalama etrafında nispeten daha küçük bir aralığa yayıldığını gösterir. Buna karşılık, bir veri kümesi için standart sapmanın daha büyük bir değeri, o veri kümesinin değerlerinin, ortalama etrafında nispeten daha geniş bir aralığa yayıldığını gösterir.
- Kitle varyansı σ^2 ile standart sapma ise σ ile gösterilmektedir. Örneklem standart sapması ise s ile ifade edilir.

$$s = \sqrt{\sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{n-1}}$$

```
var(airquality$Wind,na.rm=TRUE)
```

```
[1] 12.41154
```

```
sd(airquality$Wind,na.rm=TRUE)
```

```
[1] 3.523001
```

```
var(airquality$Solar.R,na.rm=TRUE)
```

```
[1] 8110.519
```

```
sd(airquality$Solar.R,na.rm=TRUE)
```

```
[1] 90.05842
```

Değişim Katsayısı

- Farklı serilerin değişkenliklerinin karşılaştırılmasında, farklı birimlerle ölçülmüş veri setleri söz konusu olduğundan standart sapma kullanışlı değildir.
- Bunun yerine ilgili serilerin standart sapmaları serilerin ortalama değerinin yüzdesi olarak ifade edilir ve gözlem değerlerinin büyüklüklerinden kaynaklanan farklılık ortadan kalkmış olur.

- Elde edilen bu yeni değişkenlik ölçüsü kullanılarak serilerin birbirlerine göre daha değişken ya da daha homojen oldukları konusunda yorum yapılabilir.
- Bu değer ne kadar küçükse dağılım o kadar homojendir, değişkenlik azdır. Yüzdesel olarak ifade edilir.
- Değişim Katsayısı standart sapmanın aritmetik ortalamaya bölünüp 100 ile çarpılmasıyla elde edilir.

$$D.K. = \frac{S}{\bar{X}} \times 100$$

```
dk_wind <- sd(airquality$Wind,na.rm=TRUE)/mean(airquality$Wind,na.rm=TRUE)
dk_wind
```

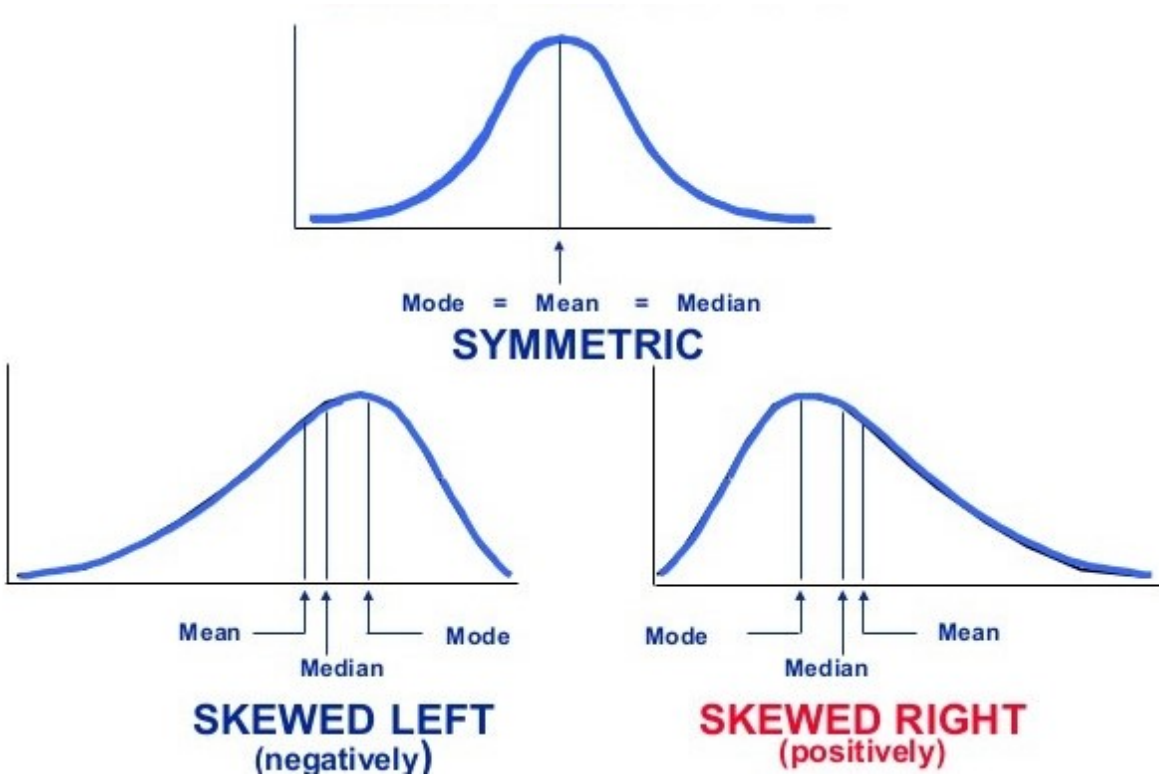
```
[1] 0.3538032
```

```
dk_solar <- sd(airquality$Solar.R,na.rm=TRUE)/mean(airquality$Solar.R,na.rm=TRUE)
dk_solar
```

```
[1] 0.4843634
```

Çarpıklık ve Basıklık

- Bir dağılımın normal dağılıma göre çarpık olup olmadığını belirlemede kullanılır. Simetrik dağılımlarda ortalama, ortanca ve tepe değeri birbirine eşittir.
- Çarpıklık katsayısı 0 ise dağılım simetriktir, 0'dan küçük ise sola çarpıktır (negatif çarpıklık), 0'dan büyük ise sağa çarpıktır (pozitif çarpıklık).
- Pozitif çarpıklıkta sağ kuyruk daha uzun iken negatif çarpıklıkta sol kuyruk daha uzundur.
- Aritmetik Ortalama, Medyan ve Mod arasındaki ilişkilere göre de çarpıklık belirlenebilir.
  - Mod $<$ Medyan $<$ Ortalama ise, dağılım sağa-çarpık yani (+) yöne eğilimli dağılımdır.
  - Ortalama $<$ Medyan $<$ Mod ise, dağılım sola-çarpık yani (-) yöne eğilimli dağılımdır.
  - Ortalama $=$ Mod $=$ Medyan ise, dağılım simetrik dağılımdır.
- Bir dağılımın normal dağılıma göre basık olup olmadığını belirlemede kullanılır.



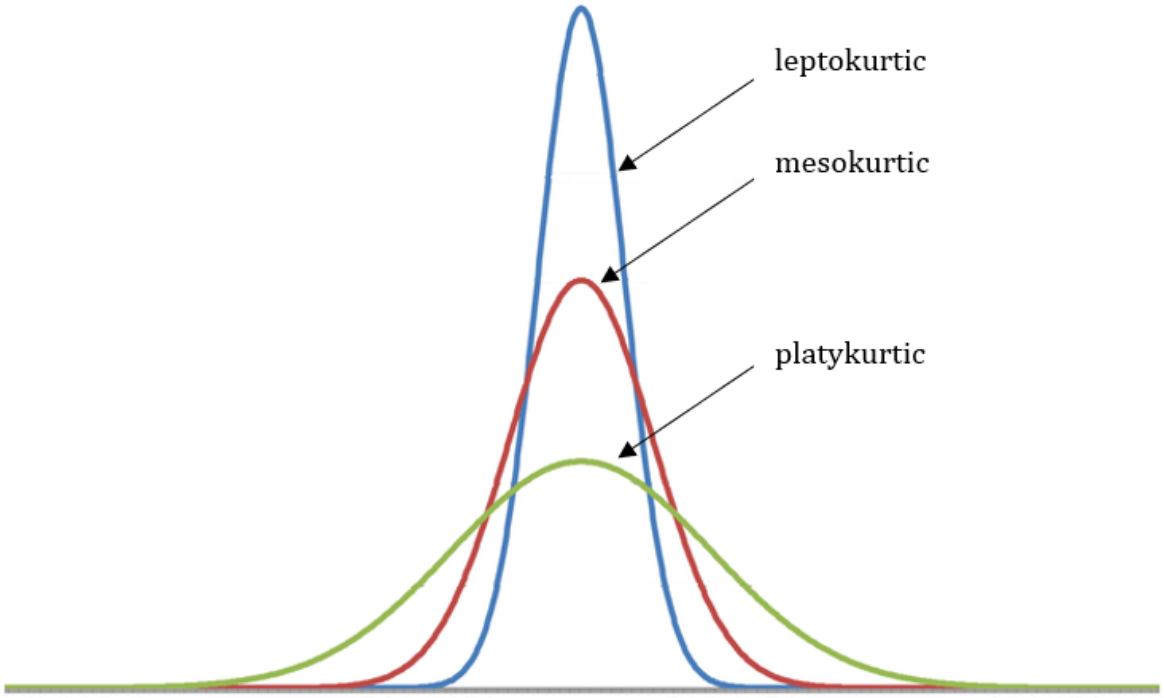
- Basıklık katsayısı sıfırdan büyükse normal dağılıma göre daha sivri, küçük ise daha basıktır.
- Basıklık katsayısı 3'e eşit ise seri normal dağılıma (mesokurtic) sahiptir. Eğer3'ten küçük ise, bir platykurtik dağılımı gösterir (daha kısa kuyruklu normal dağılımdan daha düz). Eğer 3'ten büyük ise, bir leptokurtik dağılımı gösterir (daha uzun kuyruklu normal dağılımdan daha doruğa).
- İki veya daha fazla simetrik dağılım karşılaştırıldığında aralarındaki fark basıklık ile incelenir.

```
library(moments)
skewness(airquality$Ozone,na.rm = TRUE) # sağa çarpık
```

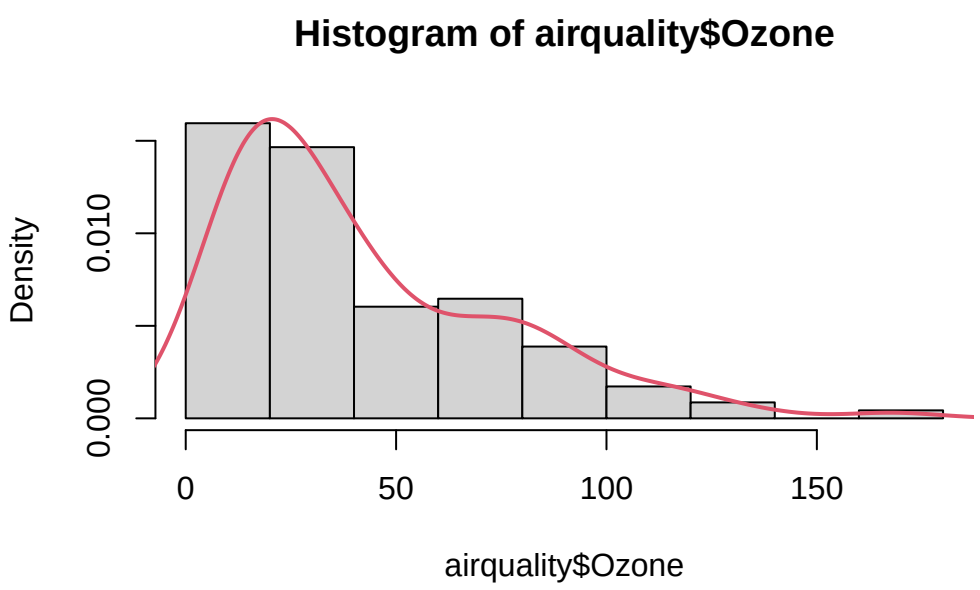
```
[1] 1.225681
```

```
kurtosis(airquality$Ozone,na.rm = TRUE) # sivri
```

```
[1] 4.184071
```




```
hist(airquality$Ozone,freq = FALSE)
lines(density(airquality$Ozone,na.rm = TRUE),col = 2, lwd = 2)
```



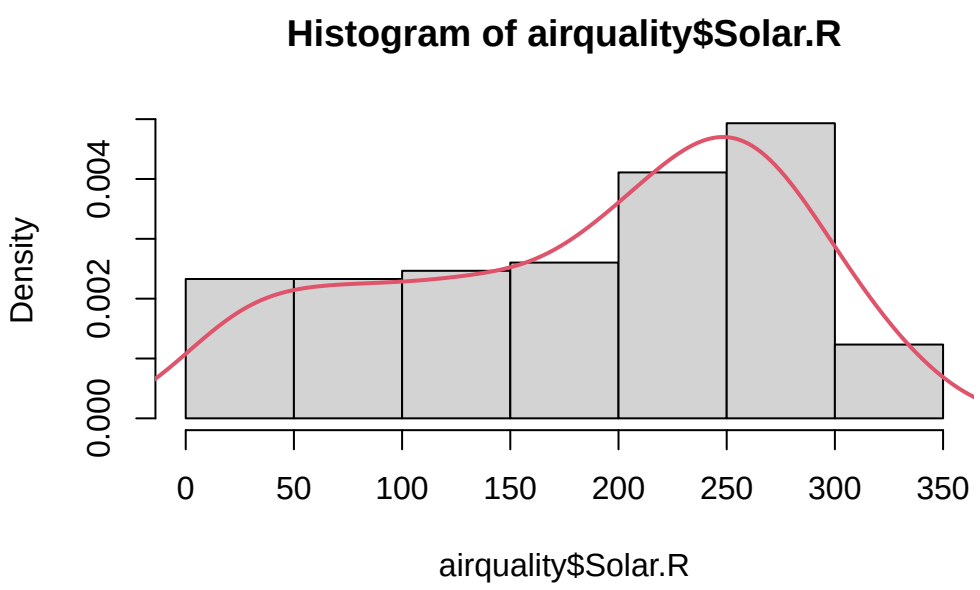

```
skewness(airquality$Solar.R,na.rm = TRUE) # sola çarpık
```

```
[1] -0.4236342
```

```
kurtosis(airquality$Solar.R,na.rm = TRUE) # sivri
```

```
[1] 2.023567
```

```
hist(airquality$Solar.R,freq = FALSE)
lines(density(airquality$Solar.R,na.rm = TRUE),col = 2, lwd = 2)
```



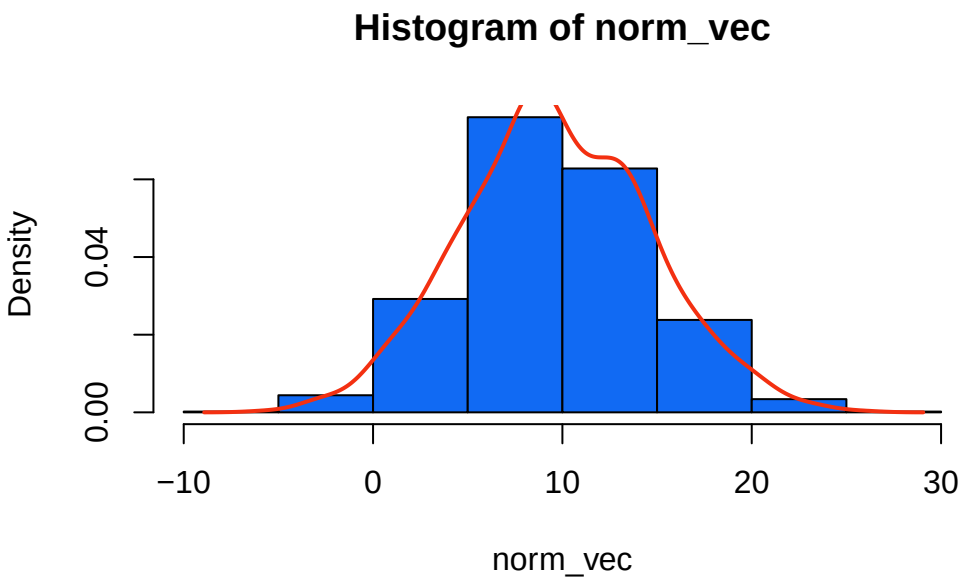

```r
# normal dağılımdan veri üretelim
norm_vec <- rnorm(1000,10,5)
skewness(norm_vec) # sola çarpık
```

```
[1] 0.06448269
```

```r
kurtosis(norm_vec) # sivri
```

```
[1] 2.897824
```

```r
hist(norm_vec,freq = FALSE,col="#116AF3") # renk kodları da kullanılabilir.
lines(density(norm_vec),col = "#F33011", lwd = 2)
```



İlişki Ölçüleri

Önceki bölümlerde, bir dağılımı tanımlayan ve özet istatistikleri hesaplayan tek bir değişkene odaklanmıştık. Tek bir değişkeni tanımlayan istatistiklere tek değişkenli istatistikler denir. İki değişken arasındaki ilişkiyi incelersek, iki değişkenli istatistiklere atıfta bulunuruz. Birkaç değişken arasındaki ilişkiler aynı anda incelenirse, çok değişkenli istatistiklere atıfta bulunuruz. İlişki ölçüleri, iki değişken arasındaki ilişkinin boyutunu özetlemek için araçlar sağlar.

İlişkiyi ölçmek için birçok araç türü olmasına rağmen, kovaryans ve Pearson korelasyon katsayıları "sayısal" veri türü için en bilinen ve yaygın araçlardır. Kovaryans ve korelasyon arasındaki temel fark, kovaryans, değerin işaretine (+'ve veya -'ve) bağlı olarak ilişkinin yönünü gösterir. Ancak korelasyon, değişkenler arasındaki "**doğrusal**" ilişkinin gücünü gösterir.

Kategorik veriler için ki-kare testi kullanılmkatadır. Spearman rho ve Kendall Tau korelasyon katsayıları da vardır ancak bunlar parametrik olmayan testlerdir ve yaygın olarak kullanılmazlar.

Değişkenler arasındaki ilişkiyi çizgi veya saçılım grafiği çizerek de incelenebilir. Ancak, bu grafiklere bakarak ilişkiden emin olmak her zaman mümkün olmayabilir. İstatistikte testler her zaman görsel araçlardan daha güçlüdür. Görsel araçlar fikir verir, testler ise fikirleri doğrular.

Kovaryans

Kovaryans, iki değişkenin ortak değişkenliğinin bir ölçüsüdür. Kovaryans $(-\infty,\infty)$ aralığında herhangi bir değer alabilir. Bir değişkenin büyük/küçük değerleri esas olarak diğer değişkenin daha büyük/küçük değerlerine karşılık geliyorsa kovaryans pozitiftir. Değişkenler zıt davranış gösterme eğilimindeyse kovaryans negatiftir. Kovaryans s_{xy} ile gösterilir ve aşağıdaki şekilde hesaplanır.

$$s_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

```
head(iris)
```

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

```
cov(iris$Sepal.Length,iris$Petal.Length) # pozitif ilişki var
```

```
[1] 1.274315
```

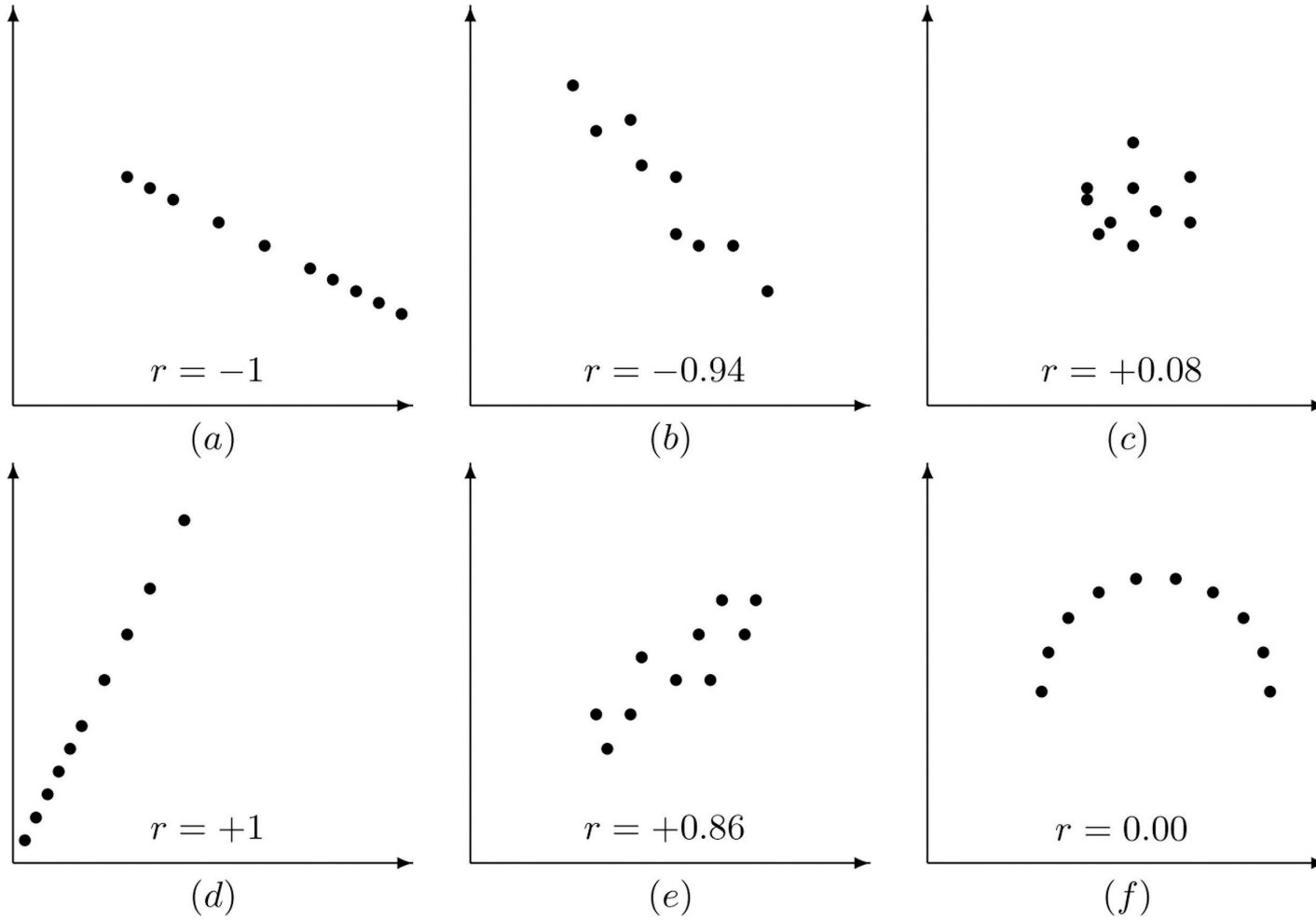
```
cov(iris$Sepal.Length,iris$Sepal.Length)
```

```
[1] 0.6856935
```

Korelasyon

Korelasyon, nicel değişkenler arasındaki ilişkiyi incelemek için yaygın olarak kullanılan bir yöntemdir. **Karl Pearson'ın** Pearson moment korelasyon katsayısı olarak da bilinen doğrusal korelasyon katsayısı **r**'dir. Doğrusal korelasyon katsayısı, iki değişken arasındaki doğrusal ilişkinin gücünü ölçer.

- Korelasyon, kovaryansın standartlaştırılmış halidir.
- Standartlaştırmadan kaynaklanan bilgi kaybı vardır.


$r = -1$
(a)
$r = -0.94$
(b)
$r = +0.08$
(c)
$r = +1$
(d)
$r = +0.86$
(e)
$r = 0.00$
(f)

- Standartlaştırılmış olduğu için korelasyonun birimi yoktur, birimsizdir.
- Korelasyon -1 ve +1 arasında değer alır.
- Korelasyon , ± 1'e yakınsa, iki değişken yüksek oranda ilişkilidir ve bir saçılım grafiği üzerinde çizilirse, veri noktaları bir çizgi etrafında kümelenir.
- Korelasyon , ± 1'den uzaksa, veri noktaları daha geniş bir alana dağılır.
- Korelasyon 0'a yakınsa, veri noktaları esasen yatay bir çizgi etrafında dağılır ve bu, değişkenler arasında neredeyse hiçbir doğrusal ilişki olmadığını gösterir.
- r=1 ise değişkenler arasında pozitif yönlü tam bir doğrusal ilişki vardır.
- r=-1 ise değişkenler arasında negatif (ters) yönlü tam bir doğrusal ilişki vardır.
- r=0 ise değişkenler arasında doğrusal ilişki yoktur.
- Korelasyon nedensel ilişki değildir.
- Korelasyon değişkenler arasındaki sebep sonuç ilişkilerini açıklamaz.
- Korelasyon matematiksel ilişkidir.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{s_{xy}}{s_x s_y}$$

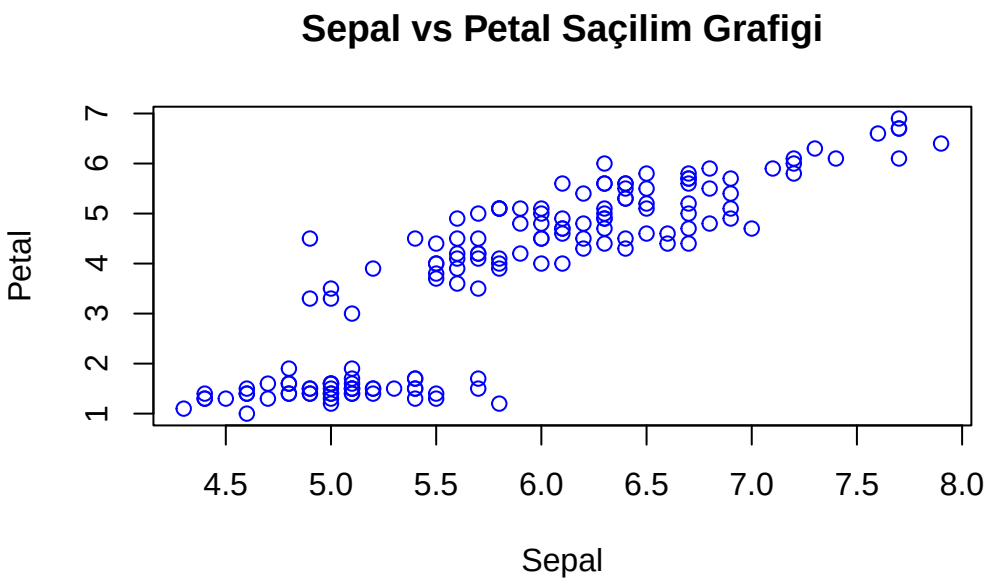
İki değişken arasındaki doğrusal ilişkinin miktarı için açık bir sınıflandırma kuralı yoktur. Bununla birlikte, aşağıdaki tablo, Pearson çarpım momenti korelasyon katsayısının sayısal değerlerinin nasıl ele alınacağı konusunda temel bir fikir verebilir.

Korelasyon Katsayısı (r)	İlişkinin Derecesi
$r > 0.90$	Çok kuvvetli
$0.70 < r \leq 0.90$	Kuvvetli
$0.50 < r \leq 0.70$	Orta
$0.30 < r \leq 0.50$	Düşük
$r < 0.30$	Zayıf

```
cor(iris$Sepal.Length,iris$Petal.Length) # kuvvetli ilişki vardır.
```

```
[1] 0.8717538
```

```r
plot(iris$Sepal.Length,iris$Petal.Length,
     col="blue",
     xlab = "Sepal",
     ylab = "Petal",
     main = "Sepal vs Petal Saçılım Grafiği")
```



Kontenjans Katsayısı

Kontenjans katsayısı C, kategorik veriler için $\chi 2$ tabanlı bir ilişki ölçüsüdür. Bağımsızlık için $\chi 2$ testine dayanır. $\chi 2$ istatistiği, kontenjans durum tablolarındaki (iki yönlü tablo, çapraz tablo tablosu veya çapraz tablolar olarak da bilinir) değişkenler arasında istatistiksel bir ilişki olup olmadığını değerlendirmeyi sağlar. Bu tür tablolarda değişkenlerin dağılımı matris formatında gösterilir. İki nominal (kategorik) değişken arasında anlamlı bir ilişki olup olmadığını belirlemek için kullanılır.

$$\chi 2 = \sum \frac{(G-B)^2}{B}$$

Burada G gözlemlenen frekansı ve B ise beklenen frekansı temsil eder . Ki-kare test istatistiği ile iki kategorik değişken arasında ilişki olup olmadığı araştırılır. Hipotez aşağıdaki gibi kurulur:

H_0 : Değişkenler arasında ilişki yoktur.

H_1 : Değişkenler arasında ilişki vardır.

Kontenjans katsayısı ise şu şekilde elde edilir:

$$C = \sqrt{\frac{\chi 2}{n + \chi 2}}$$

Burada n satır ve sütun toplamlarını ifade eder. C katsayısı 0 ile 1 arasında bir değer alır. C=0 olması iki değişken arasında ilişki olmadığına, C=1 olması ile tam ilişkili olduğu anlamına gelir.

```
# öğrencilerin sigara içme alışkanlığının egzersiz düzeyi ile ilişkili
# olup olmadığını inceleyelim.

library(MASS)
head(survey)
```

```
     Sex Wr.Hnd NW.Hnd W.Hnd    Fold Pulse    Clap Exer Smoke Height      M.I
1 Female   18.5   18.0 Right  R on L    92    Left Some Never 173.00   Metric
2   Male   19.5   20.5  Left  R on L   104    Left None Regul 177.80 Imperial
3   Male   18.0   13.3 Right  L on R    87 Neither None Occas     NA     <NA>
4   Male   18.8   18.9 Right  R on L    NA Neither None Never 160.00   Metric
5   Male   20.0   20.0 Right Neither    35   Right Some Never 165.00   Metric
6 Female   18.0   17.7 Right  L on R    64   Right Some Never 172.72 Imperial
     Age
1 18.250
2 17.583
3 16.917
4 20.333
5 23.667
6 21.000
```

```
nrow(survey)
```

```
[1] 237
```

```
tbl <-  table(survey$Smoke, survey$Exer)
tbl
```

```
       Freq None Some
  Heavy    7    1    3
  Never   87   18   84
  Occas   12    3    4
  Regul    9    1    7
```

```r
# 1.yol
chisq.test(tbl)
```

```
Warning in chisq.test(tbl): Chi-squared approximation may be incorrect
```

```

    Pearson's Chi-squared test

data:  tbl
X-squared = 5.4885, df = 6, p-value = 0.4828
```

```r
# 0.4828 p değeri .05 anlamlılık düzeyinden büyük olduğu için sigara
# içme alışkanlığının öğrencilerin egzersiz düzeyinden bağımsız olduğu
# sıfır hipotezini reddedemeyiz.

# 2.yol
summary(tbl)
```

```
Number of cases in table: 236
Number of factors: 2
Test for independence of all factors:
    Chisq = 5.489, df = 6, p-value = 0.4828
    Chi-squared approximation may be incorrect
```

Doğrusal Regresyon

Basit doğrusal regresyon, iki nicel değişken arasındaki doğrusal ilişkiyi değerlendirmeye izin veren istatistiksel bir yaklaşımdır. Daha doğrusu, ilişkinin nicelleştirilmesini ve öneminin değerlendirilmesini sağlar. Çoklu doğrusal regresyon, bu yaklaşımın bir yanıt değişkeni (nicel) ile birkaç açıklayıcı değişken (nicel veya nitel) arasındaki doğrusal ilişkileri değerlendirmeyi mümkün kılması anlamında, basit doğrusal regresyonun bir genellemesidir.

Gerçek dünyada, çoklu doğrusal regresyon, basit doğrusal regresyondan daha sık kullanılır. Bu çoğunlukla böyledir çünkü, Çoklu doğrusal regresyon, diğer değişkenlerin etkisini kontrol ederken (yani etkiyi ortadan kaldırırken) iki değişken arasındaki ilişkiyi değerlendirmeye izin verir. Veri toplamanın da kolaylaşmasıyla, veriler analiz edilirken daha fazla değişken dahil edilebilir ve dikkate alınabilir.

Basit doğrusal regresyon, iki değişken arasında doğrusal bir ilişkinin varlığını değerlendirmeye ve bu bağlantıyı nicelleştirmeye izin verir. Doğrusallığın, iki değişkenin doğrusal olarak bağımlı olup olmadığını test etmesi ve ölçmesi anlamında doğrusal regresyonda güçlü bir varsayım olduğuna dikkat etmek gerekmektedir.

Doğrusal regresyonu güçlü bir istatistiksel araç yapan şey, açıklayıcı/bağımsız değişken bir birim arttığında yanıtın/bağımlı değişkenin hangi nicelikle değiştiğini ölçmeye izin vermesidir. Bu kavram doğrusal regresyonda anahtardır ve aşağıda verilen türde soruları yanıtlamaya yardımcı olur:

- Reklama harcanan miktar ile belirli bir dönemdeki satışlar arasında bir bağlantı var mı?
- Tütün vergilerindeki artış tüketimini azaltır mı?
- Bölgeye bağlı olarak bir konutun en olası fiyatı nedir?
- Bir kişinin bir uyarana tepki verme süresi cinsiyete bağlı mıdır?

Basit doğrusal regresyon analizinde, bağımlı değişken y ile bağımsız değişken x arasındaki ilişki doğrusal bir denklem şeklinde verilir.

$$y = \beta_0 + \beta_1 x$$

Burada, β_0 sayısına kesme noktası denir ve regresyon doğrusu ile y ekseninin (x=0) kesişme noktasını tanımlar. β_1 sayısına regresyon katsayısı denir. Regresyon doğrusu eğiminin bir

ölçüsüdür. Böylece β_1, x değeri 1 birim arttığında y değerinin ne kadar değiştiğini gösterir. Model, x ve y arasında kesin bir ilişki verdiği için deterministik bir model olarak kabul edilir.

Ancak birçok durumda, iki değişken x ve y arasındaki ilişki kesin değildir. Bunun nedeni, bağımlı değişken y'nin, tahmin değişkeni x tarafından tam olarak yakalanmayan diğer bilinmeyen ve/veya rastgele süreçlerden etkilenmesidir. Böyle bir durumda veri noktaları düz bir çizgi üzerinde sıralanmaz. Bununla birlikte, veriler hala temeldeki doğrusal bir ilişkiyi takip edebilir. Bu bilinmeyenleri dikkate almak için lineer model denklemine ε ile gösterilen rastgele bir hata terimi eklenir, böylece yukarıdaki deterministik modelin aksine olasılıklı bir model elde edilir.

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Burada hata terimi ε_i'nin bağımsız normal dağılımlı değerlerden oluştuğu varsayılır, $e_i \sim N(0, \sigma^2)$.

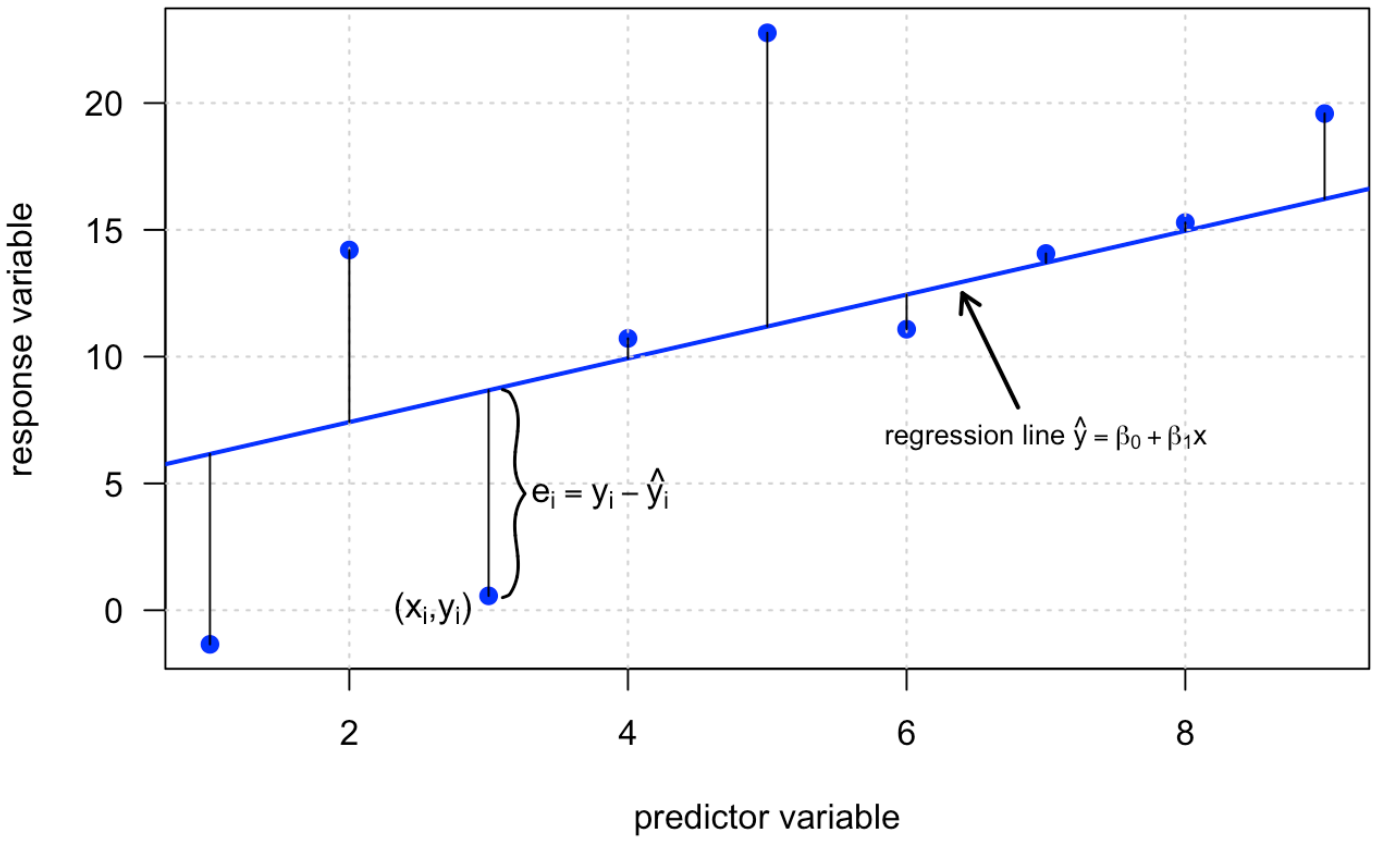
Doğrusal regresyon modeli hakkında aşağıdaki varsayımlar yapılır:

- Bağımlı değişken tesadüfi bir değişkendir ve normal dağılım göstermektedir.
- Tahmin hataları tesadüfidir ve normal dağılım gösterirler.
- Hatalar birbirinden bağımsızdır (otokorelasyon yoktur).
- Hata varyansı sabittir ve veriler arasında hiç değişmediği varsayılır (eşit varyanslılık-homoscedasticity).
- Eğer çoklu regresyon analizi yapılıyorsa, bağımsız değişkenlerin birbirleri ile bağlantısının olmaması gereklidir. Buna çoklu bağlantı (multicollinearity) olmaması varsayımı adı verilir.
- Bağımlı değişken ile bağımsız değişkenler arasında doğrusal bir ilişki olmalıdır.
- Gözlem sayısı parametre sayısından büyük olmalıdır.

```
library(gapminder)
library(dplyr)
```

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

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

    filter, lag
```


response variable
predictor variable
20
15
10
5
0
2
4
6
8
$e_i = y_i - \hat{y}_i$
(x_i, y_i)
regression line $\hat{y} = \beta_0 + \beta_1 x$

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

    intersect, setdiff, setequal, union
```

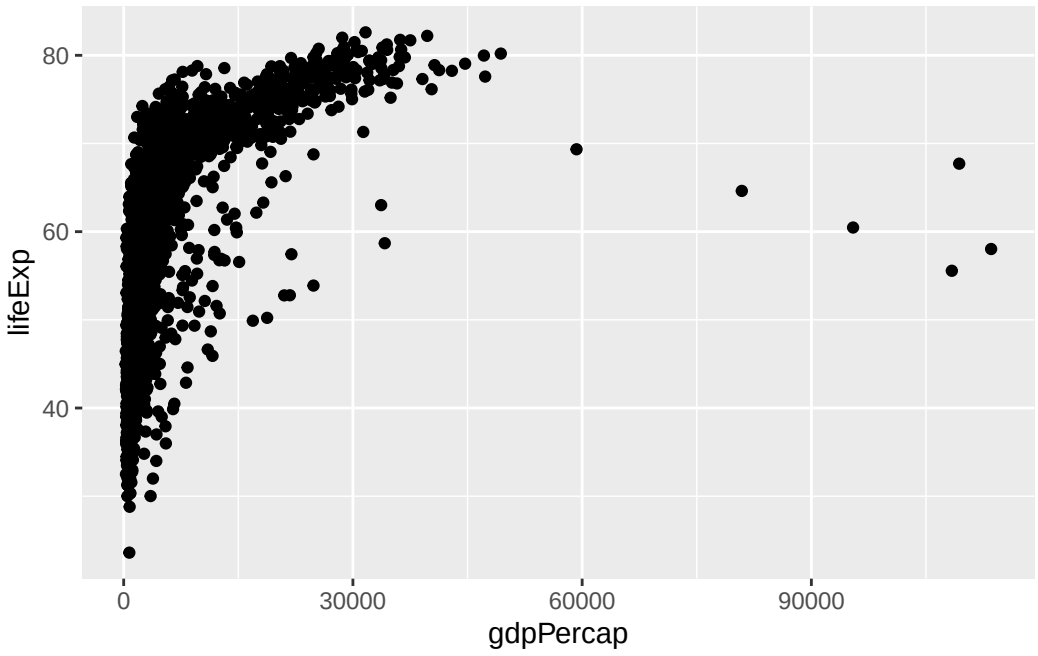
```
library(ggplot2)

# gapminder veri setine bakalım

glimpse(gapminder)
```

```
Rows: 1,704
Columns: 6
$ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

```
summary(gapminder)
```

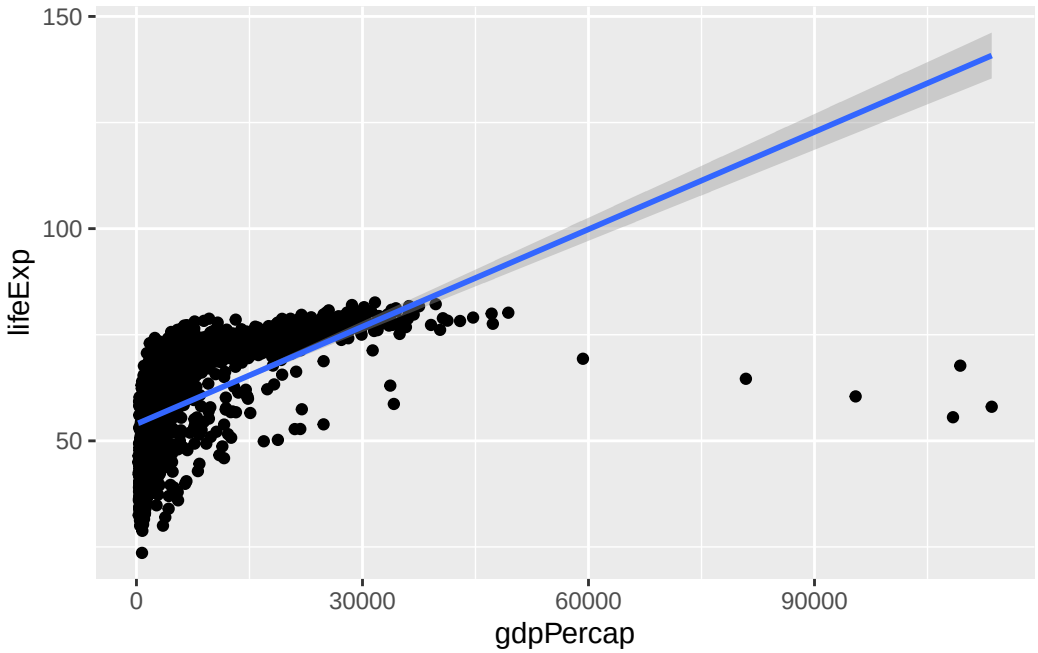
```
       country        continent        year         lifeExp     
 Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60  
 Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20  
 Algeria    :  12   Asia    :396   Median :1980   Median :60.71  
 Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47  
 Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85  
 Australia  :  12                  Max.   :2007   Max.   :82.60  
 (Other)    :1632                                                
      pop              gdpPercap       
 Min.   :6.001e+04   Min.   :   241.2  
 1st Qu.:2.794e+06   1st Qu.:  1202.1  
 Median :7.024e+06   Median :  3531.8  
 Mean   :2.960e+07   Mean   :  7215.3  
 3rd Qu.:1.959e+07   3rd Qu.:  9325.5  
 Max.   :1.319e+09   Max.   :113523.1  
```

```
# kişi başına milli gelir ile yaşam beklentisi değişkenlerini görselleştirelim.

ggplot(gapminder, aes(gdpPercap, lifeExp)) +
  geom_point()
```




```
ggplot(gapminder, aes(gdpPercap, lifeExp)) +
  geom_point() +
  geom_smooth(method = "lm",se=TRUE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
# regresyon modeli kuralım

model1 <- lm(lifeExp ~ gdpPercap, data = gapminder)
model1
```

```
Call:
lm(formula = lifeExp ~ gdpPercap, data = gapminder)

Coefficients:
(Intercept)    gdpPercap
  5.396e+01    7.649e-04
```

i Yorum

Yani burada söyleyebileceğimiz şey, GSYİH'daki her 1 artış için, yaşam beklentisinde 0.0007649 yıllık bir artış görmeyi bekleyebiliriz. Modelimizi daha iyi anlayabilmek için model üzerinde **`summary()`** fonksiyonunu kullanabiliriz. Ayrıca artıkların normalliğini de bakmak da fayda var.

summary fonksiyonu ile modelimizin verilere ne kadar iyi uyduğu hakkında biraz daha bilgi alıyoruz. Genel modelimiz ve her değişken için p-değerlerini görebiliriz. R^2 değeri, veri kümenizdeki varyansın ne kadarının modeliniz tarafından açıklanabileceğini temel olarak, modelinizin verilere ne kadar iyi uyduğunu gösterir. Bu değer 0 ile 1 arasında değişir ve büyük olması beklenir. Genel olarak, modelinizde kaç değişken kullandığınızı telafi eden düzeltilmiş R^2'yi kullanırız. Aksi halde başka bir değişken eklemek her zaman R^2'yi artırır.

```
summary(model1)
```

```
Call:
lm(formula = lifeExp ~ gdpPercap, data = gapminder)

Residuals:
    Min      1Q  Median      3Q     Max
-82.754  -7.758   2.176   8.225  18.426

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.396e+01  3.150e-01  171.29   <2e-16 ***
gdpPercap   7.649e-04  2.579e-05   29.66   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.49 on 1702 degrees of freedom
Multiple R-squared:  0.3407,    Adjusted R-squared:  0.3403
F-statistic: 879.6 on 1 and 1702 DF,  p-value: < 2.2e-16
```

Modele gdp değişkenin logaritmasını alarak ve continent (kıta) ve year (yıl) değişkenlerini de ekleyerek çoklu regresyon analizi sonuçlarına bakalım.

```
model2 <- lm(lifeExp ~ log(gdpPercap) + continent + year, data = gapminder)
summary(model2)
```

```
Call:
lm(formula = lifeExp ~ log(gdpPercap) + continent + year, data = gapminder)

Residuals:
     Min       1Q   Median       3Q      Max
-25.0433  -3.2175   0.3482   3.6657  15.1321
```

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)       -4.659e+02  1.667e+01  -27.94   <2e-16 ***
log(gdpPercap)     5.024e+00  1.595e-01   31.50   <2e-16 ***
continentAmericas  8.926e+00  4.630e-01   19.28   <2e-16 ***
continentAsia      7.063e+00  3.959e-01   17.84   <2e-16 ***
continentEurope    1.251e+01  5.097e-01   24.54   <2e-16 ***
continentOceania   1.275e+01  1.275e+00   10.00   <2e-16 ***
year               2.416e-01  8.586e-03   28.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.813 on 1697 degrees of freedom
Multiple R-squared:  0.7982,    Adjusted R-squared:  0.7975
F-statistic:  1119 on 6 and 1697 DF,  p-value: < 2.2e-16
```

Yorum

Bu sonuçlara göre R^2 değeri 0.79'a yükselmiştir. Değişken sayısını artırmak model başarısını artırmış görünüyor. Ayrıca katsayıların hepsinin de anlamlı çıktığı göz ardı edilmemelidir.
Afrika kıtası haricinde, veri kümemizdeki kıtaların her biri için bir satır var. Bunun sebebi Afrika kıtası referans kıta olarak burada belirlenmesinden kaynaklanmaktadır. Yani kıtalara göre verileri yorumlarken Afirika kıtasına göre değerlendirme yapılacaktır. Örneğin Avrupa'da olmak ortalama olarak, Afrika'da olmaktan 12.51 yıl daha fazla yaşam beklentisine sahip olmak anlamına gelmektedir.

Tavsiye

Model sonuçlarının daha güzel ve temiz (tidy) bir formatta görünmesi için **broom** paketi kullanılabilir.

```r
library(broom)

# gözlem düzeyinde sonuçlar
augment(model2)
```

```
# A tibble: 1,704 x 10
   lifeExp `log(gdpPercap)` continent  year .fitted .resid    .hat .sigma
     <dbl>            <dbl> <fct>     <int>   <dbl>  <dbl>   <dbl>  <dbl>
```

```
 1    28.8            6.66 Asia       1952    46.3  -17.5 0.00470   5.80
 2    30.3            6.71 Asia       1957    47.8  -17.5 0.00425   5.80
 3    32.0            6.75 Asia       1962    49.2  -17.2 0.00393   5.80
 4    34.0            6.73 Asia       1967    50.3  -16.3 0.00380   5.80
 5    36.1            6.61 Asia       1972    50.9  -14.8 0.00399   5.80
 6    38.4            6.67 Asia       1977    52.4  -14.0 0.00393   5.81
 7    39.9            6.89 Asia       1982    54.7  -14.9 0.00367   5.80
 8    40.8            6.75 Asia       1987    55.2  -14.4 0.00422   5.80
 9    41.7            6.48 Asia       1992    55.1  -13.4 0.00529   5.81
10    41.8            6.45 Asia       1997    56.2  -14.4 0.00588   5.80
# i 1,694 more rows
# i 2 more variables: .cooksd <dbl>, .std.resid <dbl>
```

```r
#model düzeyinde sonuçlar
glance(model2)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik    AIC    BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1     0.798         0.797  5.81     1119.       0     6 -5414. 10843. 10887.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```