R ile İstatistiğe Giriş ve Veri Dönüşümü

Servet Çetin

İçindekiler

Önsöz

Devlet yardımları, ekonomik ve sosyal kalkınmayı desteklemek amacıyla yürütülen önemli araçlardır. Ancak bu yardımların etkinliğinin değerlendirilmesi, kaynakların verimli kullanımı ve doğru politikaların oluşturulması açısından kritik bir öneme sahiptir. Etki değerlendirme yöntemleri, bu süreçte bilimsel bir temel sağlayarak karar alma mekanizmalarına rehberlik eder.

Etki değerlendirme süreçlerinin güvenilir sonuçlar verebilmesi, kullanılan yöntemlerin sağlamlığı kadar, bu süreçlerde kullanılan verilerin niteliğiyle de doğrudan ilişkilidir. R programlama dili, analiz ve modelleme çalışmalarında sağladığı esneklik ve güç ile etki değerlendirme yöntemlerinin uygulanmasında kritik bir rol oynamaktadır. Bu çalışmada, devlet yardımlarının etki değerlendirme süreçlerinde kullanılabilecek yöntemlere teorik ve uygulamalı bir temel oluşturulması amaçlanmıştır.

Etki değerlendirme süreçlerinde doğru ve anlamlı sonuçlara ulaşmak, yalnızca kullanılan yöntemlerin etkinliği ile değil, aynı zamanda güvenilir ve tutarlı verilere dayanılarak gerçekleştirilen analizlerle mümkündür. Bu nedenle, istatistiksel analiz ve veri dönüşüm süreçleri, çalışmanın odak noktalarından biri olarak ele alınmıştır. Bu süreçler, devlet yardımlarının etkilerinin doğru bir şekilde ölçülmesinin ve daha etkili politikaların geliştirilmesinin temelini oluşturmaktadır. Bu çalışmanın, Başkanlığımız ve diğer bakanlık ve kurumlarda yürütülen faaliyetler için yardımcı bir rehber olabileceği değerlendirilmektedir.

Çalışma, R programı ile uyum içinde çalışan Quarto platformu kullanılarak hazırlanmıştır. Bu platformun sunduğu dinamik ve modüler yapı, kitabın etkili bir şekilde yapılandırılmasına olanak sağlamıştır. Kitap, Quarto'nun kitap formatı özelliğiyle derlenerek, hem kolay okunabilir bir kaynak hem de uygulamalı bir rehber olarak tasarlanmıştır.

Çalışmanın, ülkemizin kalkınma hedeflerine katkı sağlaması ve bu alanda çalışan uzmanlara yardımcı olmasını dileriz.

Strateji ve Bütçe Başkanlığı Devlet Yardımları Genel Müdürlüğü

Giriş

Bu çalışma, devlet yardımlarının etkilerinin ölçülmesinde kullanılan istatistiksel analiz ve veri dönüşüm süreçlerine odaklanmaktadır. Çalışma, temel istatistiksel kavramlardan başlayarak, R programlama dili kullanılarak gerçekleştirilen veri manipülasyonu ve analitik yöntemlere kadar geniş bir yelpazeyi kapsamaktadır.

Çalışmanın ilk bölümü, istatistiksel analizlerde temel teşkil eden kavramların açıklanmasıyla başlamaktadır. Bu kapsamda, veri tipleri ele alınarak sayısal (nicel) ve kategorik (nitel) verilerin sınıflandırılmasına ve bu sınıfların alt gruplarına (örneğin, sürekli, kesikli, nominal, ordinal) değinilmiştir. Her bir veri tipi, özellikleri ve kullanım alanları açısından örneklerle desteklenmiştir. Bu açıklamalar, analiz süreçlerinde verilerin doğru şekilde sınıflandırılması ve uygun yöntemlerin seçilmesi açısından önem taşımaktadır.

Merkezi eğilim ölçüleri (ortalama, medyan ve mod) ve yayılım ölçüleri (aralık, varyans, standart sapma, çeyrekler açıklığı), veri kümesinin genel özelliklerini ve dağılımını anlamak için kullanılan temel araçlar olarak ele alınmıştır. Bu ölçülerin avantajları ve sınırlamaları açıklanmış, farklı veri yapılarında nasıl kullanılabilecekleri örneklerle gösterilmiştir. Dağılım türleri ve merkezi limit teoremi gibi istatistikte sıkça kullanılan kavramlara da yer verilmiş, bu kavramların istatistiksel analiz süreçlerinde taşıdığı önem vurgulanmıştır.

Çalışmanın ikinci bölümü, veri manipülasyonu ve dönüşüm tekniklerine odaklanmaktadır. Veri çerçeveleri üzerinde yapılan seçim, sıralama, filtreleme, dönüştürme ve özetleme gibi işlemler, R programlama dili kullanılarak uygulamalı örneklerle ele alınmıştır. Özellikle dplyr paketi ile veri manipülasyonu süreçleri sade bir şekilde anlatılmış ve kullanılan yöntemlerin adımları açıklanmıştır. Bu yöntemler, veri analizi sürecinin başlangıcında sıkça karşılaşılan zorluklara çözüm sunmayı amaçlamaktadır.

Veri temizleme işlemleri kapsamında eksik verilerle çalışmaya yönelik detaylı bir anlatı sunulmaktadır. Eksik verilerin tespiti, analiz sürecindeki etkileri ve bu durumların yönetimi için kullanılan yöntemler, teorik açıklamalar ve R programlama diliyle gerçekleştirilen uygulamalarla desteklenmiştir. Eksik veri oranlarının hesaplanması, bu verilerin görselleştirilmesi ve eksik değerlerin doldurulmasına yönelik çeşitli yaklaşımlar kitapta yer bulmuştur. Özellikle MICE ve dlookr gibi R paketlerinin bu alandaki işlevselliği, uygulamalı örneklerle açıklanmıştır. Eksik verilerin uygun yöntemlerle işlenmesi, analiz sonuçlarının doğruluğunu artırmak ve bu verilerin veri setindeki diğer bilgilerle uyumlu hale gelmesini sağlamak açısından önem taşımaktadır.

Kitabın bu iki bölümü, okuyucunun hem istatistiksel analiz hem de veri manipülasyonu konularında sağlam bir altyapı kazanmasını sağlamayı hedeflemektedir. Teorik bilgiler ve uygulamalı örneklerin bir

arada sunulması ile veri analizi süreçlerinde karşılaşabilecek zorluklara çözüm üretilebilmesi amaçlanmıştır.

Part I

Temel İstatistik

Çalışmanın *Temel İstatistik* bölümü, istatistiksel analizlerin yapı taşlarını oluşturan temel kavram ve yöntemlere odaklanmaktadır. İlk olarak, veri tipleri ele alınmış ve sayısal (nicel) verilerin sürekli ve kesikli, kategorik (nitel) verilerin ise nominal ve ordinal olarak sınıflandırılması yapılmıştır. Bu sınıflandırma, farklı veri türlerinin analiz süreçlerinde nasıl ele alınması gerektiğini açıklamaktadır. Ayrıca, merkezi eğilim ölçüleri (ortalama, medyan, mod) ve yayılım ölçüleri (aralık, varyans, standart sapma) gibi istatistiksel araçlar ayrıntılı bir şekilde incelenmiş ve bunların veri setlerini özetlemedeki rolleri vurgulanmıştır. Merkezi limit teoremi, dağılım türleri ve olasılık dağılımları (örneğin, binom, Poisson, normal dağılım) gibi istatistikte önemli kavramlara da bölümde yer verilmiş, bu kavramların temel prensipleri uygulamalı örneklerle desteklenmiştir.

Bölümün devamında, istatistiksel analizlerde sıklıkla kullanılan güven aralıkları ve hipotez testleri ayrıntılı bir şekilde ele alınmaktadır. Güven aralıklarının nasıl hesaplandığı ve bu aralıkların sonuçların yorumlanmasındaki önemi açıklanmıştır. Hipotez testleri kapsamında, sıfır hipotezi ve alternatif hipotez kavramları, p-değeri ve anlamlılık seviyelerinin istatistiksel çıkarımlar üzerindeki etkisi ele alınmıştır. Ayrıca, farklı hipotez testi türleri ve bu testlerin hangi durumlarda kullanılması gerektiği açıklanmıştır.

1 İstatistikte Veri Tipleri

İstatistikte veri tipleri, analiz edilecek verilerin özelliklerine ve ölçüm yöntemlerine göre sınıflandırılır. İki ana gruba ayrılırlar: **Sayısal (Nicel)** ve **Kategorik (Nitel)**. Bu veri tiplerini anlamak, doğru analiz yöntemlerini seçmek için kritik öneme sahiptir.

1.1 Sayısal (Nicel) Veriler

Sayısal veriler, sayılarla ifade edilen ve genellikle miktar veya ölçüm sonucu olan verilerdir. Nicel veriler, sürekli veya kesikli olmak üzere iki alt gruba ayrılır.

1.1.1 Sürekli (Measured)

- **Tanım:** Belirli bir aralıkta herhangi bir değer alabilen, ölçümle elde edilen verilerdir. Bu tür veriler genellikle ölçü aletleriyle elde edilir (metre, kilogram, saat vb.).
- **Özellikler:**
  - Sonsuz sayıda değer alabilir.
  - Genellikle ondalıklı değerler içerir.
- **Örnekler:**
  - Bir kişinin boyu: 175.4 cm
  - Bir suyun sıcaklığı: 23.7°C
  - Bir koşucunun 100 metreyi tamamlama süresi: 10.53 saniye

1.1.2 Kesikli (Counted)

- **Tanım:** Sayılabilir ve yalnızca tam sayı değerleri alabilen verilerdir. Bu tür veriler genellikle bir şeyin sayısını ifade eder.
- **Özellikler:**
  - Sayılar tam sayıdır; ondalıklı değer almaz.
- **Örnekler:**
  - Bir sınıfta bulunan öğrenci sayısı: 25

  - Bir mağazada satılan ürün adedi: 120
  - Bir kişinin aylık kitap okuma sayısı: 3

1.2 Kategorik (Nitel) Veriler

Kategorik veriler, sayısal olmayan ve sınıflar veya gruplar halinde sınıflandırılan verilerdir. Kategorik veriler nominal veya ordinal olarak ikiye ayrılır.

1.2.1 Nominal (Sırasız)

- **Tanım:** Belirli bir sıralama içermeyen, yalnızca kategorileri veya sınıfları ifade eden verilerdir.
- **Özellikler:**
  - Kategoriler arasında bir sıralama yoktur.
  - Genellikle metinsel olarak ifade edilir.
- **Örnekler:**
  - Cinsiyet: Erkek, Kadın
  - Kan grubu: A, B, AB, 0
  - Göz rengi: Kahverengi, Mavi, Yeşil

1.2.2 Ordinal (Sıralı)

- **Tanım:** Kategoriler arasında belirli bir sıralama olan verilerdir. Ancak bu sıralama arasındaki farklar eşit veya kesin değildir.
- **Özellikler:**
  - Sıralama içerir; ancak farkların büyüklüğü ölçülemez.
- **Örnekler:**
  - Eğitim düzeyi: İlkokul, Ortaokul, Lise, Üniversite
  - Müşteri memnuniyeti: Çok memnun, Memnun, Nötr, Memnun Değil
  - Yarış sıralaması: 1., 2., 3.

Veri Tiplerinin Seçimi ve Önemi

1. **Sayısal Veriler:** İstatistiksel analizlerde genellikle aritmetik işlemlere uygundur. Örneğin, bir grup insanın boy ortalamasını hesaplamak.
2. **Kategorik Veriler:** Genellikle gruplama ve sınıflandırma için kullanılır. Örneğin, farklı kan gruplarının bir popülasyondaki dağılımını analiz etmek.

1.3 Özet Tablo

Veri Tipi	Tanım	Örnekler
Sürekli	Ölçülen, belirli bir aralıkta değer alır	Boy: 175.4 cm, Sıcaklık: 23.7°C
Kesikli	Sayılan, yalnızca tam sayı değer alır	Öğrenci sayısı: 25, Ürün adedi: 120
Nominal	Sırasız kategoriler	Cinsiyet: Erkek, Kadın; Göz rengi: Mavi, Yeşil
Ordinal	Sıralı kategoriler	Eğitim düzeyi: Lise, Üniversite; Yarış sıralaması: 1., 2.

Veri türlerinin anlaşılması, doğru istatistiksel analiz yöntemlerini seçmek ve sonuçları daha iyi yorumlamak için önemlidir.

2 R'da Veri Tipleri ve Veri Yapıları

2.1 Veri Tipleri

R'daki veri tipleri, bir değişkenin bellekte nasıl saklandığını ve ne tür işlemler yapılabileceğini belirleyen temel yapı taşlarıdır. Bu veri tiplerini anlamak, R'da verilerle etkin bir şekilde çalışmanın anahtarıdır. Veri tipleri, hangi tür veriyi nasıl sakladığımızı ve bu veriyle ne tür işlemler yapabileceğimizi belirler. Aşağıda, R'da kullanılan temel veri tiplerini ve kullanım alanlarını bulabilirsiniz:

2.1.1 `Numeric` (Sayısal)

- **Tanım:** Ondalıklı veya tam sayıları ifade eder. Bu tip, en yaygın kullanılan veri türüdür.
- **Özellikler:**
  - Hem tam sayılar hem de ondalıklı sayılar bu kategoridedir.
  - Varsayılan olarak, sayılar `numeric` olarak tanımlanır.

```
# Sayısal bir tam sayı tanımlama
sayi <- 42

# Ondalıklı bir sayı tanımlama
ondalikli <- 3.14

# Değişkenlerin sınıflarını kontrol etme
class(sayi)        # "numeric"
```

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

```
class(ondalikli)   # "numeric"
```

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

- **Kullanım Alanı:** Matematiksel işlemler, istatistiksel hesaplamalar ve ölçüm verileri.

2.1.2 `Integer` (Tamsayı)

- **Tanım:** Tam sayıları temsil eder. Numeric tipine benzer, ancak yalnızca tam sayı değerleri içerir.
- **Özellikler:** Tamsayı olarak bir değer belirtmek için `L` eklenir (örneğin, `5L`).

```
# Tamsayı bir değişken tanımlama
tamsayi <- 100L

# Değişkenin sınıfını kontrol etme
class(tamsayi)  # "integer"
```

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

- **Kullanım Alanı:** İndeksleme, sayaçlar veya tam sayı gerektiren işlemler.

2.1.3 `Character` (Karakter)

- **Tanım:** Metinsel verileri temsil eder. Kelimeler, cümleler veya herhangi bir metin bilgisi için kullanılır.
- **Özellikler:** Değerler çift tırnak (`"`) veya tek tırnak (`'`) ile tanımlanır.

```
# Metin türünde bir değişken tanımlama
metin <- "Merhaba R"

# Değişkenin sınıfını kontrol etme
class(metin)  # "character"
```

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

- **Kullanım Alanı:** İsimler, açıklamalar, kategorik veriler veya etiketler.

2.1.4 `Logical` (Mantıksal)

- **Tanım:** Doğru (`TRUE`) veya yanlış (`FALSE`) durumlarını ifade eder.
- **Özellikler:** Mantıksal veri tipleri, genellikle koşullu ifadelerde kullanılır.

```r
# Mantıksal değişkenler tanımlama
dogru_mu <- TRUE
yanlis_mi <- FALSE

# Değişkenin sınıfını kontrol etme
class(dogru_mu)  # "logical"
```

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

- **Kullanım Alanı:** Koşullar, filtreleme ve kontrol akışları.

2.1.5 `Complex` (Karmaşık)

- **Tanım:** Karmaşık sayılar, reel ve sanal bileşenleri içeren sayılardır.
- **Özellikler:** Karmaşık sayıların biçimi: `a + bi` (örneğin, `2 + 3i`).

```r
# Karmaşık sayı tanımlama
karmasik <- 1 + 2i

# Değişkenin sınıfını kontrol etme
class(karmasik)  # "complex"
```

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

- **Kullanım Alanı:** Matematikte, özellikle ileri düzey hesaplamalarda.

2.1.6 `Factor` (Faktör)

- **Tanım:** Faktörler, kategorik verilerin saklanması için kullanılır. Bu tip, sınıflandırma ve seviyelendirme için idealdir.
- **Özellikler:**
  - Faktörler, seviyeler (`levels`) adı verilen kategorileri içerir.
  - Bellek kullanımı açısından verimlidir.

```r
# Faktör türünde bir değişken tanımlama
cinsiyet <- factor(c("Erkek", "Kadın", "Kadın"))

# Değişkenin sınıfını kontrol etme
class(cinsiyet)      # "factor"
```

```
[1] "factor"
```

```r
# Faktör seviyelerini görüntüleme
levels(cinsiyet)     # "Erkek" "Kadın"
```

```
[1] "Erkek" "Kadın"
```

- **Kullanım Alanı:** Anket yanıtları, kategorik değişkenler ve gruplama.

> **i** Veri Tiplerini Bilmek Neden Önemlidir?
>
> 1. **Bellek Yönetimi:** Doğru veri tipini kullanmak, bellek verimliliği sağlar.
> 2. **Doğru Hesaplamalar:** Veri tipi yanlış seçilirse, beklenmeyen sonuçlar veya hata mesajları alınabilir.
> 3. **Dönüşüm:** Veri tiplerini bilmek, gerekli dönüşümleri yaparak (örneğin, sayısaldan karaktere) veriyle daha esnek çalışmayı mümkün kılar.
>
> **Örnek Dönüşüm:**
>
> ```r
> # Numeric bir değişken tanımlama
> sayi <- 123
>
> # Numeric'ten karaktere dönüşüm
> karakter <- as.character(sayi)
>
> # Dönüştürülen değişkenin sınıfını kontrol etme
> class(karakter)  # "character"
> ```
>
> ```
> [1] "character"
> ```
>
> Veri tiplerini anlamak, R'da verileri işlemek için temel bir adımdır ve verilerle daha etkili bir şekilde çalışmanızı sağlar.

2.2 Veri Yapıları

R'daki veri yapıları, verilerin nasıl organize edildiğini ve işlendiğini belirler. Farklı boyut ve türdeki veri kümelerini temsil etmek için çeşitli veri yapıları kullanılır. Aşağıda, R'da kullanılan temel veri yapıları ve bu yapıların nasıl kullanılabileceği açıklanmıştır:

2.2.1 `Vector` (Vektör)

- **Tanım:** Homojen bir veri yapısıdır; yalnızca bir tür veri içerir (örneğin, tümü sayısal veya tümü karakter).
- **Özellikler:** Bir vektör, `c()` fonksiyonu ile oluşturulur.

```
# Sayısal bir vektör tanımlama
sayisal_vektor <- c(1, 2, 3, 4, 5)

# Karakter bir vektör tanımlama
karakter_vektor <- c("Ali", "Asya", "Mehmet")

# Vektörlerin sınıfını kontrol etme
class(sayisal_vektor)    # "numeric"
```

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

```
class(karakter_vektor)   # "character"
```

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

- **Kullanım Alanı:** Basit veri listeleri, bir boyutlu veriler.

2.2.2 `Matrix` (Matris)

- **Tanım:** İki boyutlu ve homojen bir veri yapısıdır; yalnızca bir tür veri içerir.
- **Özellikler:** Matris, `matrix()` fonksiyonu ile oluşturulur.

```
# Bir matris tanımlama
matris <- matrix(1:6, nrow = 2, ncol = 3)

# Matrisi ekrana yazdırma
print(matris)
```

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

```r
# Matrisin sınıfını kontrol etme
class(matris)  # "matrix"
```

```
[1] "matrix" "array"
```

- **Kullanım Alanı:** Matematiksel hesaplamalar ve matris işlemleri.

2.2.3 `Array` (Dizi)

- **Tanım:** Çok boyutlu ve homojen bir veri yapısıdır.
- **Özellikler:** Bir dizi, `array()` fonksiyonu ile oluşturulur.

```r
# Bir dizi (array) tanımlama
dizi <- array(1:12, dim = c(2, 3, 2))

# Diziyi ekrana yazdırma
print(dizi)
```

```
, , 1

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

, , 2

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

- **Kullanım Alanı:** Daha yüksek boyutlu verilerle çalışmak.

2.2.4 `Data Frame` (Veri Çerçevesi)

- **Tanım:** Heterojen bir veri yapısıdır; farklı türde veriler içerebilir (örneğin, sayısal ve karakter birlikte).
- **Özellikler:** Bir veri çerçevesi, `data.frame()` fonksiyonu ile oluşturulur.

```r
# Bir veri çerçevesi (data frame) tanımlama
veri_cercevesi <- data.frame(
    ID = c(1, 2, 3),              # Kimlik numarası sütunu
    Ad = c("Ahmet", "Asya", "Mehmet"), # İsim sütunu
    Yas = c(25, 30, 22)           # Yaş sütunu
)

# Veri çerçevesini ekrana yazdırma
print(veri_cercevesi)
```

```
  ID     Ad Yas
1  1  Ahmet  25
2  2   Asya  30
3  3 Mehmet  22
```

- **Kullanım Alanı:** Tablo formatında veriler.

> Neden data.frame() yerine tibble() tercih edilebilir?
>
> - **Daha kullanıcı dostudur:** Tibble, büyük veri setlerinde sadece ilk birkaç satırı göstererek konsol çıktısını okunabilir hale getirir.
> - **Modern veri analizi için optimize edilmiştir:** `tidyverse` ekosistemiyle tam uyumlu çalışır ve pipeline (%>%) kullanımıyla veri manipülasyonunu kolaylaştırır.
> - **Varsayılan davranışları daha sezgiseldir:** Karakter vektörlerini `factor`'a dönüştürmez, sütun isimlerinde esneklik sunar ve veri işleme sürecinde sık yapılan hataları önler.

```r
# Gerekli kütüphanenin yüklenmesi
library(tidyverse)

# Bir tibble tanımlama
tb <- tibble(
    isim = c("Ahmet", "Asya", "Mehmet"),  # İsim sütunu (character)
    yas = c(25, 30, 22),                  # Yaş sütunu (numeric)
    evli = c(TRUE, FALSE, FALSE)          # Evli durumu sütunu (logical)
)

# Tibble'ı ekrana yazdırma
print(tb)
```

```
# A tibble: 3 x 3
  isim     yas evli
  <chr>  <dbl> <lgl>
```

```
1 Ahmet     25 TRUE
2 Asya      30 FALSE
3 Mehmet    22 FALSE
```

> **Veri Yapılarının Seçimi**
>
> Veri yapılarının doğru seçimi, veri analizi ve modelleme işlemlerini daha verimli hale getirir.

2.2.5 `List` (Liste)

- **Tanım:** Heterojen bir veri yapısıdır; herhangi bir veri türünü içerebilir.
- **Özellikler:** Bir liste, `list()` fonksiyonu ile oluşturulur.
- **Örnekler:**

```r
# Bir liste tanımlama
liste <- list(
    isim = "Ayşe",              # Karakter eleman
    yas = 30,                   # Sayısal eleman
    evli = TRUE,                # Mantıksal eleman
    sayilar = c(1, 2, 3)        # Vektör eleman
)

# Listeyi ekrana yazdırma
print(liste)
```

```
$isim
[1] "Ayşe"

$yas
[1] 30

$evli
[1] TRUE

$sayilar
[1] 1 2 3
```

- **Kullanım Alanı:** Birden fazla veri türünü aynı yapıda saklamak.

Veri Yapılarının Seçimi

1. **Boyut ve Tür:** Verinin boyutu (tek boyutlu, iki boyutlu, çok boyutlu) ve türü (homojen, heterojen) veri yapısını seçerken belirleyicidir.
2. **Hafıza Kullanımı:** Daha büyük veri yapıları için belleği etkili kullanmak önemlidir.
3. **Uygulama:** Veri yapıları, analiz türüne göre seçilir. Örneğin:
    - Basit veriler için vektör.
    - Tablo formatı için veri çerçevesi.
    - İleri düzey hesaplamalar için matris veya diziler.

Veri yapılarının doğru seçimi, veri analizi ve modelleme işlemlerini daha verimli hale getirir.

Referanslar

https://www.w3schools.com/r/r_variables.asp https://www.modernstatisticswithr.com/

https://openintro-ims.netlify.app/exploratory-data-analysis

https://trevorfrench.github.io/R-for-Data-Analysis/p1c3-data-types.html

https://www.w3schools.com/r/r_data_types.asp

https://app.datacamp.com/learn/courses/introduction-to-statistics-in-r

3 Merkezi Eğilim ve Yayılım Ölçüleri

Merkezi eğilim ve yayılım ölçüleri, bir veri kümesinin genel özelliklerini özetleyen istatistiksel araçlardır. Bu araçlar, veri kümesinin merkezini, dağılımını ve çeşitliliğini anlamak için kullanılır.

3.1 Merkezi Eğilim Ölçüleri

Merkezi eğilim ölçüleri, bir veri kümesindeki "merkezi değeri" tanımlamak için kullanılır. Temel merkezi eğilim ölçüleri ortalama, medyan ve moddur.

3.1.1 Ortalama

- **Tanım:** Bir veri kümesindeki tüm değerlerin toplamının, veri sayısına bölünmesiyle elde edilen değerdir. mean() fonksyonu ile hesaplanır.

```
# Bir vektör tanımlama
veri <- c(12, 15, 20, 25, 30, 35, 40, 50)

# Vektörün ortalamasını hesaplama
ortalama <- mean(veri)

# Ortalamayı ekrana yazdırma
print(ortalama)
```

```
[1] 28.375
```

Aykırı değerler, ortalamayı büyük ölçüde etkileyebilir

Aykırı değerler, bir veri kümesindeki diğer verilere kıyasla çok büyük veya çok küçük olan sıra dışı değerlerdir. Bu değerler, veri kümesinin genel eğilimine uymadığı için ortalamayı önemli ölçüde etkileyebilir.
Neden Ortalamayı Etkiler? Ortalama, tüm değerlerin toplamının veri sayısına bölünmesiyle hesaplanır. Bu nedenle, aşırı büyük veya küçük bir değer toplam üzerinde büyük bir ağırlık oluşturabilir ve ortalamayı bu uç değere doğru kaydırabilir.

```r
# Aykırı değer içermeyen bir veri kümesi
veri_normal <- c(10, 15, 20, 25, 30)
ortalama_normal <- mean(veri_normal)
print(ortalama_normal) # Çıktı: 20
```

```
[1] 20
```

```r
# Aykırı değer içeren bir veri kümesi
veri_aykiri <- c(10, 15, 20, 25, 1000) # 1000 bir aykırı değerdir
ortalama_aykiri <- mean(veri_aykiri)
print(ortalama_aykiri) # Çıktı: 214
```

```
[1] 214
```

3.1.2 Medyan (Ortanca)

- **Tanım:** Bir veri kümesindeki sıralanmış verilerin ortanca değeridir. Eğer veri sayısı tek ise ortanca değeri, çift ise ortanca iki değerin ortalamasıdır. median() fonksyonu ile hesaplanır.

```r
# Bir veri kümesi tanımlama
veri <- c(3, 5, 7, 9, 11, 13)

# Verinin medyanını hesaplama
medyan <- median(veri)

# Medyanı ekrana yazdırma
print(medyan)
```

```
[1] 8
```

! Medyanın Avantajı: Aykırı Değerlerden Etkilenmeme

Medyan, veri kümesindeki sıralanmış değerlerin ortanca noktasını temsil ettiği için, çok büyük ya da çok küçük uç değerler (aykırı değerler) medyanı etkilemez. Bu, medyanı özellikle aykırı değerlere sahip veri setlerinde güvenilir bir merkezi eğilim ölçüsü haline getirir.
Neden Etkilenmez? Medyan, sadece sıralanmış verilerin ortanca noktasını kullanır. Veri kümesindeki en küçük veya en büyük değer, sıralamayı değiştirse bile ortanca değeri değiştirmez.

```r
# Aykırı değer içermeyen bir veri kümesi
veri_normal <- c(10, 15, 20, 25, 30)
medyan_normal <- median(veri_normal)
print(medyan_normal) # Çıktı: 20
```

```
[1] 20
```

```r
# Aykırı değer içeren bir veri kümesi
veri_aykiri <- c(10, 15, 20, 25, 1000)
medyan_aykiri <- median(veri_aykiri)
print(medyan_aykiri) # Çıktı: 20
```

```
[1] 20
```

Medyan, hem aykırı değer olmayan hem de aykırı değer içeren veri setlerinde aynı sonucu vermiştir. Bu, medyanın aykırı değerlere karşı dayanıklı olduğunu gösterir.

Medyan, özellikle uç değerlerin bulunduğu durumlarda daha güvenilir bir merkezi eğilim ölçüsü sağlar. Örneğin, gelir dağılımı gibi büyük uç değerlere sahip veri kümelerinde medyan, genellikle ortalamadan daha anlamlı bir ölçü olur.

3.1.3 Mod

- **Tanım:** Bir veri kümesinde en sık görülen değerdir. Mod, özellikle kategorik verilerde veya sayısal verilerde tekrarlanan değerleri anlamak için kullanılır.
- **Özellikler:**
  - Tekrarlanan bir değer yoksa mod yoktur.
  - Birden fazla değerin aynı sıklıkla görülmesi durumunda, veri kümesi "çok modlu" (multimodal) olarak adlandırılır.

```r
# Gerekli kütüphanenin yüklenmesi
# install.packages("modeest") # Eğer yüklenmemişse, bu satırı çalıştırın
library(modeest)

# Bir veri kümesi tanımlama
veri <- c(1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6)

# Verinin modunu hesaplama (en sık görülen değer)
mod <- mfv(veri)
```

```r
# Modu ekrana yazdırma
print(mod) # En sık görülen değer: 4
```

```
[1] 4
```

Kategorik Veri için Mod

Mod, kategorik verilerde en sık görülen kategoriyi belirlemek için de kullanılabilir.

```r
# Gerekli kütüphanenin yüklenmesi
# install.packages("modeest") # Eğer yüklenmemişse bu satırı çalıştırın
library(modeest)

# Bir kategorik veri kümesi tanımlama
kategorik_veri <- c("Kırmızı", "Mavi", "Kırmızı", "Yeşil", "Mavi", "Mavi")

# Kategorik verinin modunu hesaplama (en sık görülen kategori)
mod_kategorik <- mfv(kategorik_veri)

# Modu ekrana yazdırma
print(mod_kategorik) # En sık görülen kategori: "Mavi"
```

```
[1] "Mavi"
```

> **i** Mod Uygulama Alanı ve Açıklamalar
>
> **Uygulama Alanı:** Mod, özellikle kategorik verilerde en sık görülen grubu veya sınıfı anlamak için yararlıdır. Sayısal verilerde de merkezi eğilimi gösterir, ancak tüm veri setini tam olarak temsil etmeyebilir.
> **Eksiklikler:** Mod her zaman var olmayabilir (tekrarlanan bir değer yoksa). Çok modlu veri setlerinde tek bir merkezi eğilim ölçüsü sağlamak zordur.

Mod, veri setindeki tekrarlanan veya en yaygın değeri anlamak için kullanışlı bir araçtır. Ancak veri setinin niteliğine bağlı olarak diğer merkezi eğilim ölçüleri (örneğin, ortalama veya medyan) ile birlikte değerlendirilmesi daha kapsamlı bir analiz sağlar.

3.2 Yayılım Ölçüleri

Yayılım ölçüleri, veri kümesindeki değerlerin çeşitliliğini veya veri setinin ne kadar yayıldığını ölçmek için kullanılır.

3.2.1 Aralık

- **Tanım:** Bir veri kümesindeki en büyük değer ile en küçük değer arasındaki farktır. Aralık, veri setindeki yayılımın en temel ölçüsüdür.

```
# Bir veri kümesi tanımlama
veri <- c(4, 7, 10, 15)

# Verinin aralığını hesaplama
aralik <- max(veri) - min(veri)

# Aralığı ekrana yazdırma
print(aralik) # Çıktı: 11
```

```
[1] 11
```

> **Aykırı Değerlerin Etkisi**
>
> Aralık, yalnızca iki değere (en büyük ve en küçük) bağlıdır. Veri kümesindeki diğer değerleri dikkate almaz. Aykırı değerler varsa yayılımı olduğundan fazla gösterebilir. Dolayısıyla Aykırı değerler (veri kümesindeki diğer değerlere göre çok büyük veya çok küçük olan değerler), aralığı büyük ölçüde etkileyebilir.

3.2.2 Varyans

- **Tanım:** Verilerin ortalamadan ne kadar uzaklaştığını ölçen bir yayılım ölçüsüdür. Her bir veri noktasının ortalamadan farkının karesi alınarak bu farkların ortalaması hesaplanır. Karesi alındığı için varyans, tüm sapmaları pozitif hale getirir ve yayılımın büyüklüğünü anlamamızı sağlar. var() fonksyonu ile hesaplanır.
- **Popülasyon varyansı**: $\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$
- **Örneklem varyansı**: $s^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$

```
# Bir veri kümesi tanımlama
veri <- c(2, 4, 6)

# Verinin ortalamasını hesaplama
ortalama <- mean(veri)
print(ortalama) # Çıktı: 4
```

```
[1] 4
```

```r
# Verinin varyansını hesaplama
varyans <- var(veri)
print(varyans) # Çıktı: 4
```

```
[1] 4
```

Hesaplama Adımları

1. Ortalama: $\bar{x} = \frac{(2+4+6)}{3} = 4$
2. Ortalamadan sapmalar: $(2-4), (4-4), (6-4) = -2, 0, 2$
3. Sapmaların kareleri: $(-2)^2, (0)^2, (2)^2 = 4, 0, 4$
4. Varyans: $\frac{(4+0+4)}{3-1} = \frac{8}{2} = 4$

```r
# Daha geniş bir veri kümesi tanımlama
veri_genis <- c(5, 10, 15, 20, 25)

# Verinin ortalamasını hesaplama
ortalama_genis <- mean(veri_genis)
print(ortalama_genis) # Çıktı: 15
```

```
[1] 15
```

```r
# Verinin varyansını hesaplama
varyans_genis <- var(veri_genis)
print(varyans_genis) # Çıktı: 62.5
```

```
[1] 62.5
```

Hesaplama Adımları

1. Ortalama: $\bar{x} = \frac{(5+10+15+20+25)}{5} = 15$
2. Ortalamadan sapmalar: $(-10,\ -5,\ 0,\ 5,\ 10)$
3. Sapmaların kareleri: $(100,\ 25,\ 0,\ 25,\ 100)$
4. Varyans: $\frac{(100+25+0+25+100)}{5-1} = \frac{250}{4} = 62.5$

Dikkat Edilmesi Gereken Hususlar

1. **Birimin Kare Farklılığı:**

   - Varyansın birimi, orijinal veri biriminin karesidir. Örneğin, veri "cm" birimindeyse, varyans "cm²" biriminde olur.
   - Bu nedenle, varyansın yorumu bazen zor olabilir. Standart sapma, birimi orijinal veri birimiyle aynı olduğu için daha sık tercih edilir.

2. **Varyansın Yorumlanması:**

   - Varyans büyüdükçe, verilerin ortalamadan daha fazla sapma gösterdiği anlamına gelir.
   - Küçük bir varyans, verilerin ortalamaya yakın olduğunu ifade eder.

Varyans, veri kümesindeki yayılımı anlamak için güçlü bir araçtır. Ancak birimi farklı olduğu için doğrudan yorumlamak yerine genellikle standart sapma ile birlikte değerlendirilir. Daha büyük veri kümelerinde varyans, verilerdeki farklılıkları daha doğru bir şekilde yansıtır.

3.2.3 Standart Sapma

- **Tanım:** Standart sapma, veri değerlerinin ortalamadan ne kadar saptığını gösteren bir yayılım ölçüsüdür. Varyansın karekökü olarak hesaplanır ve birimi, veri birimiyle aynıdır. Bu özelliği, standart sapmayı yorumlamayı daha kolay hale getirir. sd() fonksyonu ile hesaplanır.
- **Popülasyon Standart Sapması:** $\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$
- **Örneklem Standart Sapması:** $s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}}$

```r
# Bir veri kümesi tanımlama
veri <- c(2, 4, 6)

# Verinin standart sapmasını hesaplama
std_sapma <- sd(veri)

# Standart sapmayı ekrana yazdırma
print(std_sapma) # Çıktı: 2
```

```
[1] 2
```

Hesaplama Adımları:

1. **Ortalama:** $\overline{x} = \frac{(2+4+6)}{3} = 4$
2. **Sapmalar:** $(2-4), (4-4), (6-4) = -2, 0, 2)$
3. **Sapmaların Kareleri:** $(-2)^2, (0)^2, (2)^2 = 4, 0, 4$
4. **Varyans:** $\frac{(4+0+4)}{3-1} = \frac{8}{2} = 4$
5. **Standart Sapma:** $\sqrt{4} = 2$

i Veri Yapılarının Seçimi

1. **Birimin Korunması:**

- Standart sapmanın birimi, orijinal veri birimiyle aynıdır. Örneğin, veri "metre" ise, standart sapma da "metre" birimindedir.
- Bu özellik, standart sapmayı varyanstan daha kolay yorumlanabilir hale getirir.

2. **Veri Dağılımını Anlama:**

- Standart sapma küçükse, veri değerleri ortalamaya yakın demektir.
- Standart sapma büyükse, veri değerleri ortalamadan uzaklaşır, yani yayılım artar.

Aykırı değerler ve standart sapma

- Standart sapma, aykırı değerlerden etkilenir. Eğer veri setinde uç noktalar varsa, standart sapma olduğundan daha büyük görünebilir.
- Aykırı değerlerin etkisini azaltmak için çeyrekler açıklığı (IQR) gibi diğer yayılım ölçüleriyle birlikte kullanılabilir.

Standart sapma, verilerin yayılımını ve dağılımını anlamak için güçlü bir araçtır. Yorumlanması kolaydır ve birimi koruduğu için veri analizi sırasında sıkça tercih edilir. Ancak, aykırı değerlere duyarlılığı nedeniyle dikkatli bir şekilde değerlendirilmelidir.

3.2.4 Çeyrekler Açıklığı (Interquartile Range (IQR))

- **Tanım:** Çeyrekler açıklığı, verilerin dağılımını anlamak için kullanılan bir yayılım ölçüsüdür. Veriler, sıralandıktan sonra dört eşit parçaya bölünür ve çeyrekler açıklığı, üçüncü çeyrek (Q3) ile birinci çeyrek (Q1) arasındaki fark olarak hesaplanır. Bu ölçü, veri setindeki orta yüzde 50'lik dilimdeki yayılımı gösterir ve aykırı değerlere karşı daha dayanıklıdır.

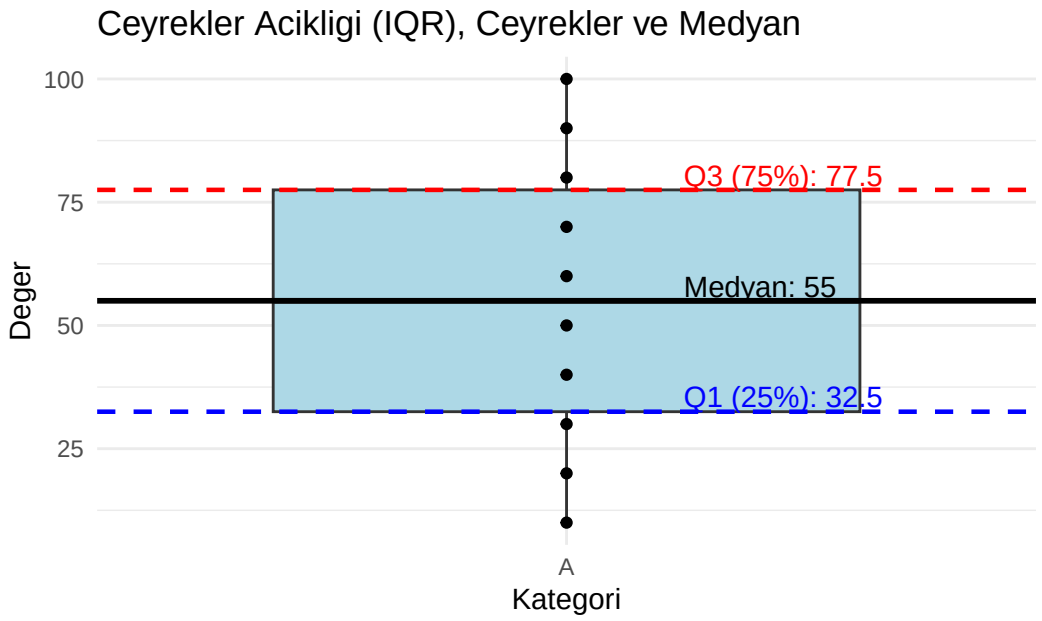
Çeyrekler açıklığı (IQR) şu şekilde hesaplanır: $IQR = Q3 - Q1$

Burada $Q1$: Birinci çeyrek, verilerin en küçük %25'lik kısmının üst sınırıdır. $Q3$: Üçüncü çeyrek ise, verilerin en küçük %75'lik kısmının üst sınırıdır.

```r
# Daha büyük bir veri kümesi tanımlama
veri_genis <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)

# Çeyreklerin hesaplanması
q1_genis <- quantile(veri_genis, 0.25) # Birinci çeyrek (Q1)
q3_genis <- quantile(veri_genis, 0.75) # Üçüncü çeyrek (Q3)

# Çeyrekler açıklığı
cayrekler_acikligi_genis <- q3_genis - q1_genis

# Çeyrekler açıklığını ekrana yazdırma
print(cayrekler_acikligi_genis) # Çıktı: 50
```

```
75%
 45
```

Açıklama: 1. **Veri kümesi:** $(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)$
2. **Birinci çeyrek (Q1):** (35) 3. **Üçüncü çeyrek (Q3):** (85) 4. **Çeyrekler açıklığı:** $Q3 - Q1 = 85 - 35 = 50$



Siyah cizgi medyani, mavi cizgi Q1'i, kirmizi cizgi Q3'u temsil eder.

Çeyrekler açıklığı

Aykırı Değerlere Dayanıklılık: Çeyrekler açıklığı, yalnızca orta yüzde 50'lik dilimi dikkate alır. Bu nedenle, aykırı değerlerden etkilenmez.
Dağılımın Anlaşılması: Veri setinin yoğunluğu ve yayılımı hakkında bilgi verir.

Çeyrekler açıklığı

Çeyrekler açıklığı, yalnızca veri setinin orta kısmını dikkate alır. Eğer veri setinin tümüne dair yayılım bilgisi gerekiyorsa, standart sapma gibi ölçülerle birlikte kullanılması daha yararlı olur.

Çeyrekler açıklığı, aykırı değerlere dayanıklı bir yayılım ölçüsüdür ve özellikle veri setinin merkezi yayılımını anlamak için güçlü bir araçtır. Veri setindeki yoğunluğu ve merkezi eğilimi analiz etmek için sıkça kullanılır.

Özet

Merkezi eğilim ölçüleri (ortalama, medyan, mod), verilerin merkezi bir noktada nasıl toplandığını anlamaya yardımcı olurken, yayılım ölçüleri (aralık, varyans, standart sapma, çeyrekler açıklığı), verilerin çeşitliliğini ve dağılımını anlamayı sağlar. Aykırı değerler, analiz sonuçlarını etkileyebileceği için dikkatli bir şekilde değerlendirilmelidir.

Referanslar

https://www.modernstatisticswithr.com/modchapter.html

https://app.datacamp.com/learn/courses/introduction-to-statistics-in-r

4 Dağılım Türleri ve Merkezi Limit Teoremi

4.1 Ayrık Olasılık Dağılımları (Discrete Probability Distributions)

Ayrık olasılık dağılımları, yalnızca belirli ve sayılabilir değerler üzerinde tanımlanan olasılık dağılımlarıdır. Bu, değerlerin kesintili olduğu ve arada başka olasılıkların bulunmadığı anlamına gelir. Örneğin, bir zarın atılması durumunda sonuçlar yalnızca {1, 2, 3, 4, 5, 6} şeklindedir. Bir zarın 3.5 gibi bir sonuç göstermesi mümkün değildir, çünkü bu değer ayrık olasılıkların tanımı dışındadır.

Bu tür dağılımlar, bir olayın belirli bir sayıda gerçekleşme olasılığını modellemek ve tahmin etmek için kullanılır. Ayrık olasılık dağılımları, genellikle bir olayın "kaç kez" veya "hangi durumda" gerçekleştiğini anlamak için kullanılır.

4.1.1 Binom Dağılımı (Binomial Distribution)

Binom dağılımı, belirli bir sayıda bağımsız denemede bir olayın kaç kez başarılı olduğunu modellemek için kullanılır. Örneğin, 10 kez madeni para atıldığında, yazının kaç kez geleceğini tahmin etmek binom dağılımıyla modellenebilir.

Binom dağılımı, her bir denemenin birbirinden bağımsız olduğu ve her birinde başarı veya başarısızlık gibi iki olası sonuç bulunduğu durumları ifade eder. Başarı olasılığı sabittir ve her bir deneme sonucu diğerlerinden bağımsızdır. Bu nedenle, belirli bir deneme sayısı için başarı sayısının olasılıklarını modellemek için oldukça kullanışlıdır.

```
# dbinom() fonksiyonu ile olasılık hesaplama

dbinom( # dbinom() fonksiyonu
  x = 3, # x: Başarı sayısı
  size = 10, # size: Toplam deneme sayısı
  prob = 0.5# prob: Her bir denemede başarı olasılığı
)
```

```
[1] 0.1171875
```

```
# 10 bağımsız denemede, başarı olasılığı 0.5 olan bir olayın tam 3 kez
# gerçekleşme olasılığını hesaplar.
```

i set.seed() Fonksiyonu

`set.seed()` fonksiyonu, rastgele sayı üretim süreçlerinde kullanılan bir kontrol mekanizmasıdır ve R'de rastgelelik ile çalışırken sonuçların yeniden üretilebilir olmasını sağlar.
Rastgelelik ve Sorunlar: Rastgele sayı üretimi aslında "sözde rastgele sayı üretimi" (pseudo-random number generation) olarak çalışır. Bu işlem bir başlangıç noktasına (seed) dayanır. Eğer başlangıç noktası aynıysa, üretilen rastgele sayılar da aynı olur.
Eğer `set.seed()` **kullanılmazsa**: R her çalıştırıldığında farklı rastgele sayılar üretir. Bu durum, analizlerin yeniden üretilebilirliğini zorlaştırır.

```
# Rastgele sayılar üretme
runif(5)
```

```
[1] 0.8884064 0.4331525 0.8160797 0.4841483 0.6209260
```

```
# 0 ile 1 arasında 5 rastgele sayı üretir. Her çalışmada sayılar da değişir.
```

Eğer `set.seed()` **kullanılırsa**: Kod her çalıştırıldığında aynı başlangıç noktası kullanılır ve aynı rastgele sayılar üretilir. Bu, kodun tutarlı ve yeniden üretilebilir olmasını sağlar.

```
# Rastgelelik için sabit bir başlangıç noktası belirleme
set.seed(123)
# set.seed() kullanıldığında, rastgele sayı üretimi kontrol altına alınır.

# 0 ile 1 arasında 5 rastgele sayı üretme
runif(5)  # Kod her çalıştırıldığında aynı sayılar üretilir. Değişmez.
```

```
[1] 0.2875775 0.7883051 0.4089769 0.8830174 0.9404673
```

set.seed(123) ile runif(5) kodu çalıştırıldığında, aynı yazılım ortamı ve aynı R sürümünde çalışan dünyadaki tüm bilgisayarlarda aynı sonucu verir.

4.1.2 Poisson Dağılımı (Poisson Distribution)

Poisson dağılımı, belirli bir zaman aralığında veya belirli bir alanda bir olayın kaç kez gerçekleştiğini modellemek için kullanılır. Örneğin, bir çağrı merkezine bir saatte gelen çağrıların sayısını tahmin etmek Poisson dağılımı ile modellenebilir.

Poisson dağılımı, olayların sabit bir ortalama hızla meydana geldiği durumları ifade eder. Bu olaylar birbirinden bağımsızdır ve herhangi bir zaman veya alanda iki olayın aynı anda gerçekleşme olasılığı yok denecek kadar azdır. Özellikle nadir olayların sayısını modellemek için oldukça kullanışlıdır.

```
# Poisson dağılımında belirli bir olayın gerçekleşme olasılığını hesaplama

# dpois() fonksiyonu
dpois( # dpois() fonksiyonu
  x = 3, # x: Olay sayısı (kaç kez gerçekleştiği)
  lambda = 5 # lambda: Ortalama olay sayısı (beklenen değer)
)
```

```
[1] 0.1403739
```

```
# Ortalama 5 olay gerçekleşen bir durumda, 3 olayın gerçekleşme olasılığı
```

4.1.3 Geometrik Dağılım (Geometric Distribution)

Geometrik dağılım, bir denemede başarıdan önceki başarısızlıkların sayısını modellemek için kullanılır. Örneğin, bir zar atışında ilk kez 6 gelene kadar kaç başarısızlık (6 dışında başka bir sayı) yaşandığını tahmin etmek bu dağılım ile modellenebilir.

Geometrik dağılım, bir olayın başarıyla sonuçlanana kadar tekrarlanmasını ifade eder. Başarıya ulaşma olasılığı, denemeler ilerledikçe giderek azalır. Teorik olarak olasılıklar sonsuza kadar devam eder; yani başarının hiçbir zaman gerçekleşmeme olasılığı sıfır değildir, ancak bu olasılık çok küçüktür. Bu nedenle geometrik dağılım, nadir olayların kaç deneme sonra gerçekleşebileceğini modellemek için oldukça kullanışlıdır.

```
# Geometrik dağılımda olasılık hesaplama
dgeom(x = 3, # x: Başarıdan önceki başarısızlık sayısı
      prob = 0.2 # prob: Her bir denemede başarı olasılığı
      )
```

```
[1] 0.1024
```

```
# Başarı olasılığı 0.2 olan bir deneyde,
# ilk başarıdan önce tam 3 başarısızlık gerçekleşme olasılığını hesaplar.
```

4.1.4 Hipergeometrik Dağılım (Hypergeometric Distribution)

Hipergeometrik dağılım, belirli bir özelliğe sahip nesnelerin bir örneklemde kaç kez seçileceğini modellemek için kullanılır. Örneğin, bir torbada 10 beyaz ve 5 siyah bilye varsa, rastgele seçilen 5 bilye içindeki beyaz bilye sayısını tahmin etmek için kullanılır.

Hipergeometrik dağılım, belirli bir popülasyondan yapılan örneklemin, geri koymadan seçilmesi durumunda belirli bir özelliğe sahip nesnelerin sayısını modellemek için kullanılır. Bu dağılımda, seçim yapıldıkça popülasyondaki nesnelerin oranı değiştiği için her seçimin olasılığı bağımsız değildir. Bu, örneklemler arası bağımlılığın olduğu durumları modellemek için oldukça kullanışlıdır.

```
# Hipergeometrik dağılımda olasılık hesaplama

dhyper( # dhyper() fonksiyonu
  x = 3, # x: Örneklemde istenen özelliğe sahip nesne sayısı
  m = 10, # m: Popülasyondaki istenen özelliğe sahip nesne sayısı
  n = 5, # n: Popülasyondaki diğer nesne sayısı
  k = 5 # k: Örneklem büyüklüğü
)
```

```
[1] 0.3996004
```

```
# Popülasyonda 10 beyaz, 5 siyah bilye bulunan bir torbadan, rastgele
# seçilen 5 bilyede tam 3 beyaz bilye bulunma olasılığını hesaplar.
```

4.2 Sürekli Olasılık Dağılımları (Continuous Probability Distributions)

Sürekli Olasılık Dağılımları (Continuous Probability Distributions) Sürekli olasılık dağılımları, bir aralıktaki değerler üzerinde tanımlanan olasılık dağılımlarıdır. Bu, değerlerin kesintisiz olduğu ve belirli bir değer yerine bir aralık için olasılık hesaplandığı anlamına gelir. Örneğin, bir kişinin boyu 170 cm olabilir, ancak 170.1 cm veya 170.25 cm gibi sonsuz sayıda ara değer de mümkündür. Bu nedenle, sürekli dağılımlarda belirli bir değerin olasılığı sıfırdır, ancak bir aralık içerisindeki olasılık hesaplanabilir.

Bu tür dağılımlar, bir olayın belirli bir aralıktaki değerlerde gerçekleşme olasılığını modellemek ve tahmin etmek için kullanılır. Sürekli olasılık dağılımları, genellikle bir değişkenin “hangi aralıkta” ya da “ne kadar” gerçekleştiğini anlamak için kullanılır. Normal dağılım, üstel dağılım ve tekdüze dağılım gibi modeller sürekli olasılık dağılımlarına örnektir.

4.2.1 Normal Dağılım (Normal Distribution)

Normal dağılım, doğal olayların çoğunu modellemek için kullanılan çan şeklinde bir sürekli olasılık dağılımıdır. Örneğin, bir sınıftaki öğrencilerin boylarının ortalamalarının etrafında simetrik bir şekilde dağıldığını gözlemlemek, normal dağılımın bir örneğidir. (Çan Eğrisi)

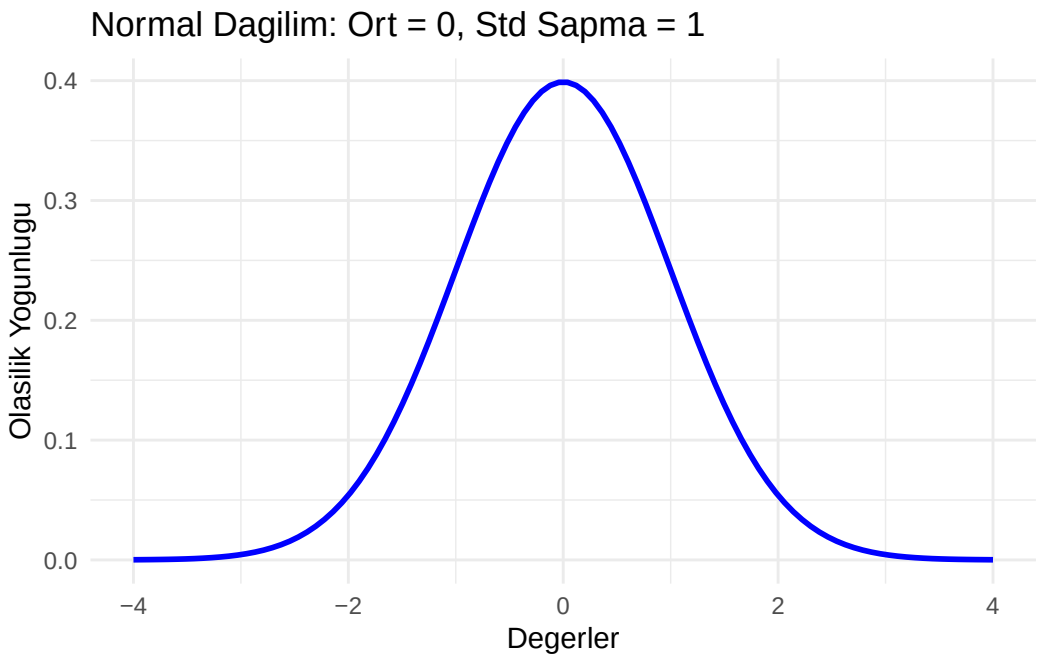
Normal dağılım, sürekli bir veri setinin ortalama etrafında yoğunlaşması ve uçlara doğru gidildikçe olasılıkların azalması durumunu modellemek için kullanılır. Normal dağılım simetriktir ve ortalama, medyan ve mod aynı noktada bulunur. Doğadaki birçok olayın normal dağılıma uyduğu gözlemlenmiştir ve bu nedenle istatistikte sıkça kullanılır.

```
# Normal dağılımda olasılık yoğunluğunu hesaplama
dnorm( # dnorm() fonksiyonu
  x = 0,# x: Değerler
  mean = 0, # mean: Ortalama
  sd = 1 # sd: Standart sapma
  )
```

```
[1] 0.3989423
```

```
# Ortalama 0 ve standart sapma 1 olan bir normal dağılımda,
# tam olarak 0 değerinin olasılık yoğunluğunu hesaplar.
```

Normal Dağılım Grafik Görünümü



Normal Dağılım Neden Önemli

Normal dağılım, istatistik biliminde temel bir yere sahiptir, çünkü birçok doğal olay ve sosyal fenomen normal dağılım özellikleri sergilemektedir. Özellikle parametrik testler (örneğin, t-testi ve ANOVA) ve regresyon analizleri, verilerin normal dağılıma uygun olduğu varsayımı üzerine inşa edilmiştir. Normal dağılımın bu denli önemli olmasının temel nedenlerinden biri, Merkezi Limit Teoremi'dir.
Etki değerlendirme yöntemlerinde, özellikle yarı deneysel yaklaşımlar (farkların farkı, eğilim skoru eşleştirme, regresyon süreksizliği gibi), analizlerin güvenilirliği ve geçerliliği büyük ölçüde normal dağılım varsayımına bağlıdır. Normal dağılım, hata terimlerinin ve sonuç değişkenlerinin istatistiksel gücünü ve analiz sonuçlarının doğruluğunu artırır. Parametrik testlerin (örneğin, t-testi, ANOVA) veya regresyon modellerinin etkin bir şekilde kullanılabilmesi için bu varsayım kritik bir öneme sahiptir. Normal dağılım sağlanmadığında, analiz sonuçları yanıltıcı olabilir ve etkilerin doğru tahmini güçleşir. Bu nedenle, yarı deneysel yöntemler uygulanmadan önce verilerin dağılımını kontrol etmek ve gerekirse gerekli dönüşümleri uygulamak, güvenilir ve anlamlı sonuçlar elde etmek için vazgeçilmez bir adımdır.

Merkezi Limit Teoremi

Merkezi Limit Teoremi, bağımsız ve aynı dağılıma sahip rastgele değişkenlerin aritmetik ortalamasının, örneklem büyüklüğü yeterince büyük olduğunda yaklaşık olarak normal dağılım göstereceğini ifade eder. Bu teorem, rastgele değişkenlerin varyansının sonlu olması durumunda geçerlidir ve örneklem büyüklüğü arttıkça, toplam veya ortalamanın dağılımının normal bir şekil alacağını belirtir. Merkezi Limit Teoremi, başlangıçta normal dağılıma uymayan verilerin bile, büyük örneklem boyutlarında ortalamalarının normal dağılıma yakınsama eğiliminde olduğunu söyler. Bu, istatistiksel analizlerde normal dağılımın neden bu kadar sık kullanıldığını açıklayan temel bir kavramdır.

4.2.2 Standart Normal Dağılım

Standart normal dağılım, ortalaması 0 ve standart sapması 1 olan özel bir normal dağılım türüdür. Normal dağılımların standartlaştırılmış hali olarak da düşünülebilir. Bu dağılımda, herhangi bir veri noktasının z-skoru, o noktanın ortalamadan kaç standart sapma uzaklıkta olduğunu gösterir.

Standart normal dağılım, eğrisinin toplam alanı 1 olacak şekilde çan şeklindedir ve ortalama etrafında simetriktir. Eğrinin en yüksek noktası, ortalama olan 0 değerindedir ve değerler uçlara doğru gittikçe olasılık yoğunluğu azalır. Tüm normal dağılımlar, bu dağılıma dönüştürülebilir, böylece farklı birimlerdeki veriler karşılaştırılabilir ve analiz edilebilir hale gelir.

Bu dağılım, istatistikte sıklıkla kullanılır çünkü normal dağılım ile ilgili tüm olasılık hesaplamalarını kolaylaştırır. Örneğin, **z-skoru** kullanılarak bir veri noktasının hangi yüzdelik dilimde olduğu veya bir değerin olasılığı standart normal dağılım tabloları yardımıyla hesaplanabilir. Standart normal dağılım,

istatistiksel analizlerde merkezi bir role sahiptir ve parametrik testler gibi birçok yöntemin temelini oluşturur.

Z-Skoru

Z-skoru, bir veri noktasının popülasyon ortalamasından kaç standart sapma uzaklıkta olduğunu gösteren bir ölçüdür. Z-skoru şu formülle hesaplanır:
$Z = \frac{X-\mu}{\sigma}$

- X: Veri noktası,
- μ: Popülasyon ortalaması,
- σ : Popülasyon standart sapmasıdır.

Z-skoru, verilerin standartlaştırılmasını sağlar ve farklı dağılımların karşılaştırılmasına olanak tanır. Örneğin, $Z = 2$ bir veri noktasının ortalamadan 2 standart sapma uzaklıkta olduğunu ifade eder.

4.2.3 t-Dağılımı (t-Distribution)

t-Dağılımı, örnekleme dağılımlarında kullanılan ve küçük örneklem boyutlarında popülasyon varyansının bilinmediği durumlar için tasarlanmış bir sürekli olasılık dağılımıdır. William Sealy Gosset tarafından geliştirilmiş ve başlangıçta Student's t-distribution olarak adlandırılmıştır. t-dağılımı, normal dağılıma benzer bir yapıya sahiptir ancak daha geniş kuyruklara sahiptir. Bu, küçük örneklemlerle çalışırken uç değerlerin daha yüksek bir olasılıkla gerçekleştiği anlamına gelir.

Matematiksel Açıklama

Bir örneklemden elde edilen ortalama ile popülasyon ortalamasının **standart hata** üzerinden farkını ölçmek için kullanılır.

Standart Hata - Standart Error (SE)

Standart hata, bir örneklem istatistiğinin (örneğin, ortalama) örnekleme dağılımındaki değişkenliğini özetleyen bir ölçüttür. Standart hata, örneklemin standart sapması (s) ve örneklem büyüklüğü (n) kullanılarak tahmin edilir ve şu formülle hesaplanır: $SE = \frac{s}{\sqrt{n}}$
Standart hata, örneklem büyüklüğü arttıkça azalır. Bu, daha büyük örneklemlerle tahminlerin daha kesin hale geldiği anlamına gelir. Bu ilişki, genellikle "***n'in karekökü kuralı***" olarak adlandırılır; standart hatayı yarıya indirmek için örneklem büyüklüğünün dört katına çıkarılması gerekir. Standart hata kavramı, **Merkezi Limit Teoremi** ile ilişkilidir, çünkü bu teorem, yeterince büyük bir örneklemde, örneklem istatistiklerinin yaklaşık olarak normal dağılım göstereceğini ifade eder.

Karıştırmayın: Standard Sapma vs. Standart Hata
Standart sapma ve standart hata farklı kavramlardır ve genellikle birbirine karıştırılır. **Standart sapma**, bireysel veri noktalarının yayılımını, yani veri setindeki değişkenliği ölçer. Öte yandan, **standart hata**, bir örneklemden hesaplanan bir istatistiğin (örneğin, örneklem ortalaması) farklı örneklemler arasında nasıl değiştiğini ölçer. Kısaca:

- **Standart Sapma (SD)**: Tek bir veri setinin yayılımını ölçer.
- **Standart Hata (SE)**: Örneklem ortalamalarının dağılımını ölçer.

Bu nedenle, **standart hata** bir tahminin hassasiyetini değerlendirirken kullanılırken, **standart sapma**, verilerin çeşitliliği hakkında bilgi sağlar.

t-dağılımının temel formülü:

$t = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}}$

Burada:

- $\bar{X}$: Örneklem ortalaması,
- μ: Popülasyon ortalaması,
- s: Örneklem standart sapması,
- n: Örneklem büyüklüğüdür.

t- Dağılımının Özellikleri

1. **Simetrik ve Çan Şeklindedir**: t-dağılımı, normal dağılım gibi çan şeklinde ve simetriktir.
2. **Serbestlik Derecesine (df - degrees of freedom) Bağlıdır**: t-dağılımı, örneklem büyüklüğü azaldıkça daha geniş kuyruklara sahiptir. Serbestlik derecesi arttıkça (örneklem büyüdükçe), t-dağılımı normal dağılıma yaklaşır.

Serbestlik Derecesi - Degrees of Freedom (df)

Serbestlik derecesi, bir istatistiksel hesaplamada serbestçe değişebilen bağımsız veri noktalarının sayısını ifade eder. Bir parametrenin tahmininde kullanılan bağımsız veri sayısından, bu tahminde kullanılan ara hesaplamalar (örneğin, ortalama gibi) çıkarılarak hesaplanır. Örneğin, bir örneklemde varyans hesaplanırken, ortalama sabit bir değer olduğu için serbestlik derecesi $N-1$ olarak belirlenir. Serbestlik derecesi, özellikle t-dağılımı ve ki-kare dağılımı gibi istatistiksel testlerde, dağılımın şekli ve genişliği üzerinde önemli bir etkiye sahiptir. Matematiksel olarak, serbestlik derecesi, bir rastgele vektörün kaç bileşeni bilindiğinde tam olarak belirlenebileceğini ifade eder.

3. **Popülasyon Standart Sapması Bilinmediğinde Kullanılır**: Normal dağılım yerine t-dağılımı, popülasyon standart sapmasının bilinmediği ve küçük örneklem boyutlarıyla çalışıldığı durumlarda kullanılır.
4. **Geniş Kuyruklar**: t-dağılımı, uç değerlerin daha yüksek bir olasılıkla gerçekleştiğini varsayar.

Kullanım Alanları

- **t-Testleri**: Küçük örneklemlerde iki grup arasındaki farkın anlamlılığını test etmek için kullanılır.

> **t-Testleri**
>
> **t-testi**, iki grup arasındaki ortalamaların istatistiksel olarak anlamlı bir fark gösterip göstermediğini belirlemek için kullanılan bir istatistiksel testtir. Küçük örneklemlerle çalışırken veya popülasyon varyansı bilinmediğinde kullanılır. t-testi, grup ortalamalarının farkını standart hata ile karşılaştırarak bir t-istatistiği hesaplar ve bu istatistiğin t-dağılımına göre anlamlılığını değerlendirir. Eğer t-istatistiği belirli bir eşik değerini (örneğin, $p < 0.05$) aşarsa, gruplar arasında anlamlı bir fark olduğu sonucuna varılır.

```r
# Örnek veri seti oluşturma
set.seed(123)  # Rastgelelik kontrolü
group1 <- rnorm(15, mean = 100, sd = 10)  # Grup 1 için rastgele değerler
group2 <- rnorm(15, mean = 110, sd = 10)  # Grup 2 için rastgele değerler

# Grupları görüntüleme
print(group1)
```

```
 [1]  94.39524  97.69823 115.58708 100.70508 101.29288 117.15065 104.60916
 [8]  87.34939  93.13147  95.54338 112.24082 103.59814 104.00771 101.10683
[15]  94.44159
```

```r
print(group2)
```

```
 [1] 127.86913 114.97850  90.33383 117.01356 105.27209  99.32176 107.82025
 [8]  99.73996 102.71109 103.74961  93.13307 118.37787 111.53373  98.61863
[15] 122.53815
```

```r
# Bağımsız iki örneklem t-testi
t_test_result <- t.test(group1, group2, var.equal = TRUE)
#t.test() Fonksiyonu: İki grubun ortalamalarının eşit olup olmadığını test eder.

# Test sonuçlarını görüntüleme
print(t_test_result)
```

```
    Two Sample t-test
```

```
data:  group1 and group2
t = -1.685, df = 28, p-value = 0.1031
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.316499   1.296023
sample estimates:
mean of x mean of y
 101.5238  107.5341
```

t-testi Sonuçlarının Anlamı:

1. **t = -1.685**: t-istatistiği, iki grup arasındaki farkın standart hata cinsinden ne kadar büyük olduğunu gösterir. Negatif değer, Grup 1'in ortalamasının Grup 2'nin ortalamasından daha düşük olduğunu ifade eder, ancak bu fark istatistiksel olarak anlamlı olmayabilir.
2. **df = 28**: Serbestlik derecesi (degrees of freedom), veri setinin büyüklüğüne ve kullanılan modelin özelliklerine bağlıdır. Burada serbestlik derecesi 28'dir.
3. **p-value = 0.1031**: p-değeri, gruplar arasındaki farkın şansa bağlı olarak oluşma olasılığını ifade eder. Burada p=0.1031p = 0.1031p=0.1031, genellikle kabul edilen $\alpha = 0.05$ anlamlılık seviyesinden büyüktür. Bu, iki grup arasında istatistiksel olarak anlamlı bir fark olmadığını gösterir.
4. **95 percent confidence interval: [-13.316499, 1.296023]**: Bu güven aralığı, iki grup arasındaki gerçek ortalama farkın hangi aralıkta olabileceğini ifade eder. Güven aralığı 0'ı içerdiği için (örneğin, -13.32 ile 1.30 arasında), iki grup arasında anlamlı bir fark olduğu söylenemez.
5. **mean of x = 101.5238, mean of y = 107.5341**: Grup 1'in (x) ortalaması 101.52, Grup 2'nin (y) ortalaması 107.53'tür. Grup 2'nin ortalaması daha yüksek, ancak bu fark istatistiksel olarak **anlamlı değildir**.

- **t-Dağılımının Regresyon Analizindeki Rolü:**

1. **Katsayıların Anlamlılığını Test Etmek**: Regresyon analizinde, bağımsız değişkenlerin katsayılarının istatistiksel olarak anlamlı olup olmadığını değerlendirmek için **t-testi** kullanılır. t-testi, katsayı tahmininin standart hatası ile katsayının değerini karşılaştırarak bir **t-istatistiği** hesaplar: $t = \frac{\hat{\beta} - \beta_0}{SE(\hat{\beta})}$. Bu t-istatistiği, t-dağılımına göre **p-değeri** hesaplamada kullanılır.

> **i** p-değeri
>
> **p-değeri**, bir istatistiksel test sonucunda, gözlenen verilerin veya daha uç sonuçların, test edilen dağılım altında tesadüfen ortaya çıkma olasılığını ifade eder.
> Yukarıdaki bağlamda, t-istatistiği, t-dağılımında hesaplanır ve p-değeri bu istatistiğin t-dağılımında daha uç değerlere düşme olasılığını temsil eder. Bu, iki grup arasındaki farkın rast-

lantısal olma ihtimalini değerlendirmek için kullanılır. Küçük bir p-değeri (genellikle $p < 0.05$), farkın tesadüfi olmaktan çok anlamlı olabileceğini işaret eder.

2. **Hata Terimlerinin Dağılımı:** Regresyon analizinde, hata terimlerinin normal dağıldığı varsayılır. Ancak örneklem boyutu küçükse, katsayı tahminlerinin dağılımı normal değil, t-dağılımına uyar.
3. **Küçük Örneklemlerde Doğruluk:** Eğer örneklem boyutu küçükse $n<30$, katsayı tahminleri t-dağılımına göre değerlendirilir, çünkü küçük örneklemlerde t-dağılımı daha geniş kuyruklara sahiptir ve bu da uç değerleri daha iyi hesaba katar.

- **Örneklem Dağılımı**: Küçük örneklem boyutları için ortalamaların dağılımını modellemek amacıyla t-dağılımı kullanılır.

4.2.4 Üstel Dağılım (Exponential Distribution)

Üstel dağılım, olaylar arasındaki bekleme sürelerini modellemek için kullanılan bir sürekli olasılık dağılımıdır. Örneğin, bir ATM'ye gelen müşteriler arasındaki işlem süreleri, üstel dağılımın tipik bir örneğidir.

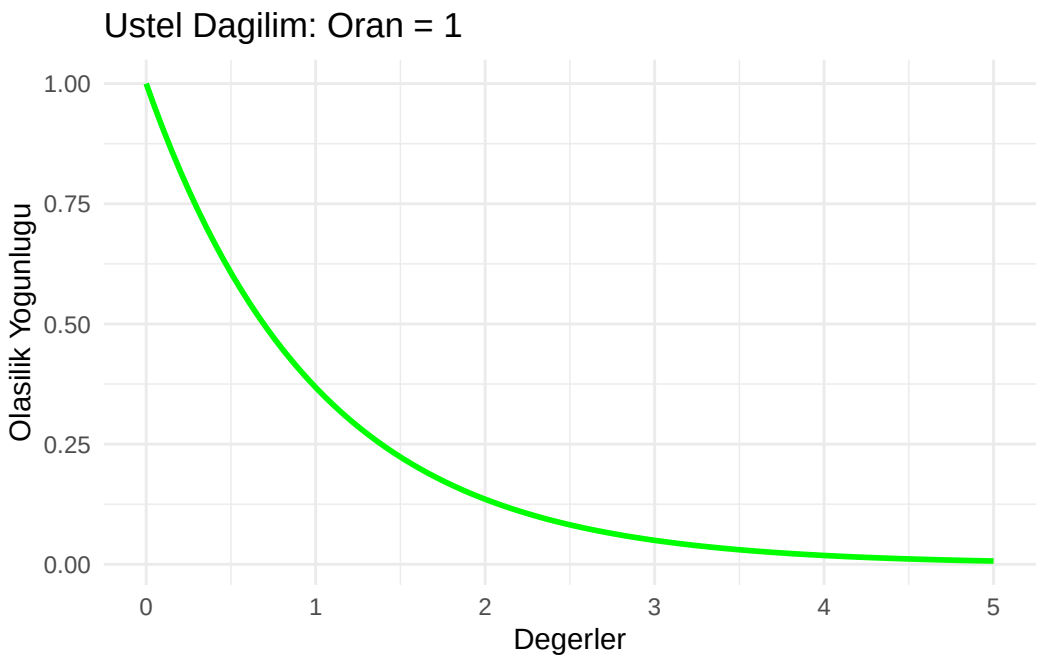
Üstel dağılım, bir olayın sabit bir hızla meydana geldiği durumlarda, olaylar arasındaki sürelerin dağılımını modellemek için kullanılır. Bu dağılım, sıfırdan başlar ve asimetrik bir şekilde sağa doğru uzanır. Olasılıklar, değerler arttıkça azalır, bu nedenle kısa sürelerde olay gerçekleşme olasılığı daha yüksektir. Üstel dağılım, genellikle bir olayın gerçekleşme hızını veya bekleme sürelerini analiz etmek için kullanılır.

```
# Üstel dağılımda olasılık yoğunluğunu hesaplama
dexp(
  x = 2, # x: Değerler (süre)
  rate = 1 # rate: Oran (olayın gerçekleşme hızı)
  )
```

```
[1] 0.1353353
```

```
# Olayın gerçekleşme hızının 1 olduğu bir üstel dağılımda,
# 2 birimlik bir sürede olayın gerçekleşme olasılığını hesaplar.
```

Üstel Dağılım Grafik Görünümü



4.2.5 Tekdüze Dağılım (Uniform Distribution)

Tekdüze dağılım, belirli bir aralıkta tüm değerlerin eşit olasılıkla meydana geldiği bir sürekli olasılık dağılımıdır. Örneğin, 1 ile 100 arasında rastgele bir sayı seçildiğinde her sayının eşit olasılıkla seçilebileceği bir durum, tekdüze dağılımın bir örneğidir.

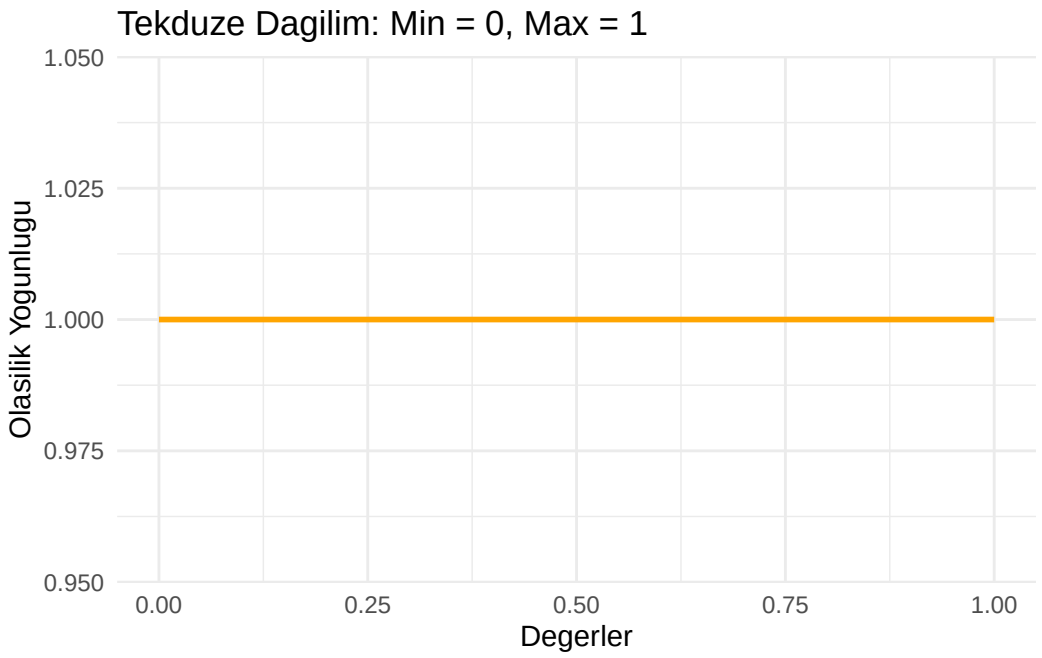
Tekdüze dağılım, belirli bir alt ve üst sınır arasında sürekli bir veri setinin her değerinin eşit olasılıkla seçilme durumunu modellemek için kullanılır. Bu dağılımın olasılık yoğunluğu, verilen aralığın dışındaki tüm değerler için sıfırdır. Tekdüze dağılım, rastgele olayların eşit olasılıkla gerçekleştiği durumları analiz etmek için sıklıkla kullanılır.

```
# Tekdüze dağılımda olasılık yoğunluğunu hesaplama
dunif(
  x = 0.5, # x: Değerler
  min = 0, # min: Alt sınır
  max = 1 # max: Üst sınır
  )
```

```
[1] 1
```

```
# 0 ile 1 arasında bir tekdüze dağılımda,
# 0.5 değerinin olasılık yoğunluğunu hesaplar.
```

Uniform Dağılım Grafik Görünümü



4.2.6 Gamma Dağılımı (Gamma Distribution)

Gamma dağılımı, sürekli bir değişkenin pozitif değerler üzerinde nasıl dağıldığını modellemek için kullanılır. Örneğin, bir çağrı merkezinde müşteriler arasındaki bekleme sürelerinin toplamını analiz etmek, gamma dağılımının bir örneğidir.

Gamma dağılımı, belirli bir olayın toplam süresini veya bir olayın gerçekleşmesi için gereken süreyi modellemek için uygundur. Şekil ve ölçek parametrelerine bağlı olarak, dağılımın formu değişebilir. Bu dağılım, sağa çarpık olabilir ve sıfırdan başlayarak pozitif değerlere doğru genişler. Gamma dağılımı, bekleme süreleri, toplam zamanlama ve sigorta risk analizi gibi alanlarda yaygın olarak kullanılır.

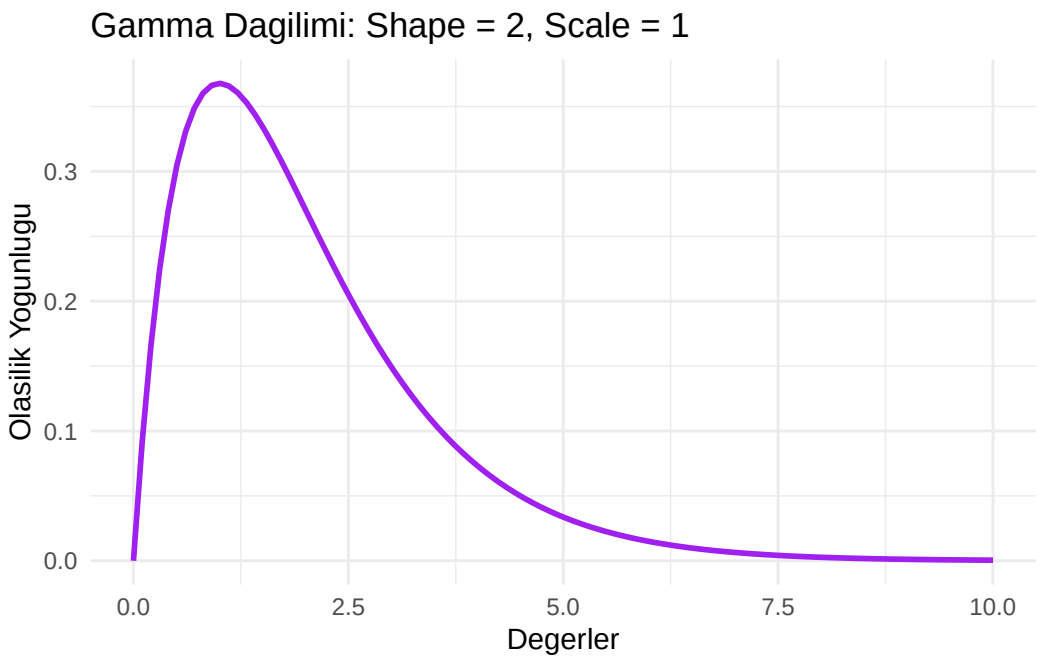
```
# Gamma dağılımında olasılık yoğunluğunu hesaplama
dgamma(
  x = 2, # x: Değerler (süre)
  shape = 2, # shape: Şekil parametresi
```

```
  scale = 1 # scale: Ölçek parametresi
  )
```

```
[1] 0.2706706
```

```
# Şekil parametresi 2 ve ölçek parametresi 1 olan bir gamma dağılımında,
# 2 değerinin olasılık yoğunluğunu hesaplar.
```

Gamma Dağılım Grafik Görünümü



Referanslar

https://bookdown.org/pbaumgartner/swr-harris/04-probability-distributions.html

https://app.datacamp.com/learn/courses/introduction-to-statistics-in-r

https://github.com/gedeck/practical-statistics-for-data-scientists/blob/master/R/code/Chapter%202%20-%20Data%20and%20sampling%20distributions.R

5 Güven Aralıkları ve Hipotez Testleri

Güven aralıkları ve hipotez testleri, çıkarımsal istatistikte karar verme süreçlerinin temel araçlarıdır. Güven aralıkları, bir anakütle parametresi için tahmin edilen aralığı sunarak belirsizliği açıkça ifade eder ve tahminin ne kadar güvenilir olduğunu gösterir. Hipotez testleri ise belirli bir iddianın (örneğin, bir ortalamanın veya farkın anlamlılığı) istatistiksel olarak desteklenip desteklenmediğini değerlendirir. Her iki yöntem de, sınırlı örneklem verilerinden anakütle hakkında genelleme yapmaya olanak tanır ve araştırmacıların veriye dayalı, anlamlı ve doğru sonuçlar çıkarmasına yardımcı olur. Bu araçlar olmadan, çıkarımsal analizlerin güvenilirliği ve bilimselliği büyük ölçüde azalır.

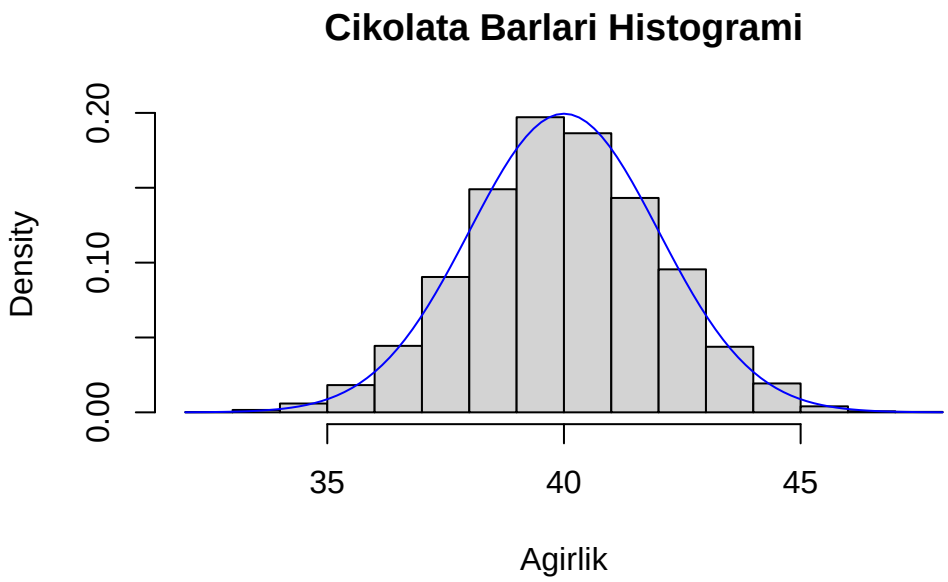
- **Örnek veri seti: Çikolata Barları**

Ankara merkezli bir çikolata üreticisi olan **Ankara Çikolatacısı**, çikolata barlarının ağırlığını (gram cinsinden) ölçmüş ve üretim sürecindeki değişkenliği anlamamızı istemiştir. Bu verileri Normal dağılımdan simüle edeceğiz. Ortalama ($\mu = 40$) ve satandart sapma ($\sigma = 2$). Çikolata bar ağırlıklarının gerçek ortalamasını ve varyansını biliyoruz, ancak ilerleyen bölümlerde (örn. t-testi yaparken) bu parametrelerin bilinmediğini varsayabiliriz.

- **Veri Simülasyonu ve İnceleme**

```
# Rastgelelik için sabit bir başlangıç noktası belirleme
set.seed(1)

# Normal dağılımdan rastgele veri üretme
choc.ankara <- rnorm(
  10000,  # Üretilecek rastgele sayıların toplam miktarı
  mean = 40,  # Normal dağılımın ortalama değeri
  sd = 2  # Normal dağılımın standart sapması
)

# İlk birkaç veriyi görüntüleme
head(choc.ankara)
```

```
[1] 38.74709 40.36729 38.32874 43.19056 40.65902 38.35906
```



5.1 Güven Aralığı (Confidence Interval - CI)

Güven aralığı, istatistikte bir anakütle parametresi için bir aralık kestirimi olup, çıkarımsal istatistikte önemli bir araçtır. Tek bir noktadan tahmin yerine, parametrenin belirli bir olasılıkla içinde bulunabileceği alt ve üst sınırlarla tanımlanan bir aralık sunar. Güven aralıkları, tahminin ne kadar güvenilir olduğunu ifade eder ve bu güvenilirlik, seçilen güven düzeyiyle (%90, %95 veya %99 gibi) belirtilir. Yüksek bir güven düzeyi seçildiğinde güven aralığı genişler, bu da parametreyi kapsama olasılığını artırır. Güven aralıkları genellikle, tahmin yönteminin normal dağılım gibi belirli varsayımları karşılaması durumunda hesaplanır. Çıkarımsal istatistikte, anakütle parametreleri hakkında daha anlamlı ve güvenilir sonuçlara ulaşmayı sağlar ve hipotez testleriyle birlikte kullanıldığında analizlerin istatistiksel gücünü artırır.

Güven Aralıklarının Hesaplanması

R'de güven aralıklarını hesaplamak oldukça basittir. Temel R paketinde doğrudan güven aralığı hesaplayan bir fonksiyon bulunmasa da, `z.test()` ve `t.test()` fonksiyonlarını kullanarak ortalamalar için güven aralıklarını hesaplayabiliriz (bu yöntemler oranlar için kullanılamaz).

5.1.1 Popülasyon Standart Sapmasını Biliniyorsa: `z.test()` Fonksiyonu

Eğer popülasyon standart sapmasını biliyorsak, `z.test()` fonksiyonunu (BSDA paketinden) kullanabiliriz. Bu fonksiyona popülasyon standart sapmasını (**sigma.x**) ve güven düzeyini (**conf.level**) sağla-

mamız gerekir. Örnek bir kullanım aşağıda verilmiştir:

```
# Gerekli paketin yüklenmesi
# install.packages("BSDA")
library("BSDA")

# z.test fonksiyonu ile güven aralığı hesaplama
z_test1 <- z.test(
  choc.ankara,       # Analiz edilecek veri seti
  sigma.x = 2,       # Popülasyonun bilinen standart sapması
  conf.level = 0.95  # Güven düzeyi (%95)
)

# Güven aralığını döndürme
z_test1$conf.int
```

```
[1] 39.94773 40.02613
attr(,"conf.level")
[1] 0.95
```

```
# Hesaplanan güven aralığını döndürür. Bu, belirlenen %95 güven düzeyi ile
# çikolata barlarının ağırlık ortalaması için tahmin edilen aralıktır.
```

Bu sonuç, çikolata barlarının ağırlık ortalamasının %95 güven düzeyiyle 39.94773 gram ile 40.02613 gram arasında olduğunu göstermektedir. Başka bir deyişle, elimizdeki verilere dayanarak, gerçek ortalama ağırlığın bu aralıkta yer alması oldukça olasıdır. Ancak, bu güven aralığı %100 kesinlik sunmaz; %5'lik bir hata payı mevcuttur.

Güven düzeyi %95 olarak belirlenmiştir. Bu da, benzer şekilde birçok örneklem üzerinden analiz yapılsa, bu örneklemlerin %95'inin gerçek ortalamayı belirtilen aralıkta içerme olasılığını ifade eder. Dolayısıyla, çikolata barlarının üretim sürecinin genel olarak tutarlı ve standartlara uygun olduğunu söyleyebiliriz. Üretimdeki varyasyonun düşük olması, kalite kontrol süreçlerinin etkili bir şekilde işlediğini göstermektedir.

> **i** Neden "%95" Güven Aralığı
>
> **Güven aralıkları**nı hesaplarken %95 güven düzeyinin önemi, istatistikte bir denge noktası olarak kabul edilmesinden kaynaklanır. %95 güven düzeyi, bir tahminin güvenilirliğini makul bir kesinlik düzeyinde ifade ederken, aynı zamanda hata payını (%5) da kontrol edilebilir seviyede tutar. Bunun birkaç temel nedeni vardır:
>
> 1. **Pratik Denge:** %95 güven düzeyi, güvenilirlik ve belirsizlik arasında dengeli bir nokta sağlar. Daha yüksek bir güven düzeyi (%99) aralığı genişletir, bu da sonuçları daha az

hassas hale getirebilir. Daha düşük bir güven düzeyi (%90) ise daha dar aralık sağlar ancak güvenilirlik azalır. %95, bu ikisi arasında ideal bir denge olarak kabul edilir.

2. **Geleneksel Kabul:** İstatistiksel analizde %95 güven düzeyi, literatürde ve uygulamada yaygın olarak kabul edilmiş bir standarttır. Bu standartlaşma, farklı çalışmalar arasında karşılaştırma yapmayı kolaylaştırır.
3. **Hata Payı (%5):** %95 güven düzeyi, tahmin edilen aralığın gerçek parametreyi kapsama olasılığının %95 olduğunu, yani yalnızca %5 hata payı olduğunu ifade eder. Bu hata payı, birçok durumda bilimsel kabul için yeterince düşük bulunur.
4. **Normatif Kullanım:** Pek çok disiplin ve uygulamada (örneğin, biyoloji, sosyal bilimler, ekonomi) %95 güven düzeyi yaygın olarak kullanılır ve bu düzeyin ötesinde bir anlam çıkarma genellikle "istatistiksel olarak anlamlı" kabul edilir.
5. **Z Değeri:** %95 güven düzeyi, standart normal dağılımda $z = \pm 1.96$ sınırına karşılık gelir. Bu, istatistiksel analizlerde kolaylıkla hesaplanabilen ve yorumlanabilen bir değerdir.

Sonuç olarak, %95 güven aralığı, tek bir tahminin doğruluğunu ifade etmez. Bunun yerine, birçok örneklem alınarak yapılan hesaplamalarda, bu aralıkların çoğunluğunun (%95) gerçek değeri kapsayacağını garanti eder.

5.1.2 Popülasyon Standart Sapmasını Bilinmiyorsa: `t.test()` Fonksiyonu

Pratikte çoğu zaman popülasyon standart sapmasını bilmediğimiz durumlarla karşılaşırız. Bu durumda `t.test()` fonksiyonunu kullanabiliriz. Burada yalnızca güven düzeyini (**conf.level**) belirteriz ve fonksiyon, standart hatayı kullanarak güven aralığını oluşturur.

```
# T-testi ile güven aralığı hesaplama
t_test1 <- t.test(
  choc.ankara,       # Analiz edilecek veri seti
  conf.level = 0.95  # Güven düzeyi (%95)
)

# Hesaplanan güven aralığını döndürme
t_test1$conf.int
```

```
[1] 39.94724 40.02661
attr(,"conf.level")
[1] 0.95
```

5.2 Hipotez Testleri (Hypothesis Tests)

Hipotez testleri, istatistiksel analizlerde belirli bir iddianın doğruluğunu değerlendirmek için kullanılan

yöntemlerdir. Hipotez testi, bir null hipoteze (yokluk hipotezi) karşı kanıtların gücünü değerlendirmek için kullanılan bir istatistiksel yöntemdir. Null hipotez, popülasyonda herhangi bir fark, ilişki veya etkinin olmadığını varsayar. Bu yöntem, null hipotezin doğru olduğunu kabul ederek, örneklem verilerinde gözlemlenen veya daha uç sonuçların olasılığını hesaplar. Bu olasılık, p-değeri ile ifade edilir.

Eğer p-değeri yeterince küçükse (örneğin, genellikle $p < 0.05$), null hipotez reddedilir ve alternatif hipotezin doğru olabileceği sonucuna varılır. Ancak, null hipotez asla "**kabul edilmez**" veya "**kanıtlanmaz**." Sadece mevcut verilerle **reddedilemeyecek kadar güçlü** olduğu düşünülür.

Bu süreç, bir ceza davasına benzetilebilir: Sanık suçsuz kabul edilir (null hipotez reddedilmez) ancak suçluluğu yeterince güçlü kanıtlarla gösterildiğinde (istatistiksel olarak anlamlı sonuçlar) sanık suçlu ilan edilir (null hipotez reddedilir).

Bu yöntemlerde iki ana hipotez tanımlanır:

- **Null hipotez** (H_0): Varsayılan durum ya da iddia (örneğin, iki grup arasında fark yoktur).
- **Alternatif hipotez** (H_1): Test edilmek istenen yeni iddia (örneğin, iki grup arasında fark vardır).

Veriler toplandıktan sonra, bu hipotezler istatistiksel araçlarla değerlendirilir. Hipotez testinin sonucunda elde edilen p-değeri, H_0'ın doğru olduğu varsayımı altında gözlemlenen sonuçların olasılığını ifade eder. Eğer p-değeri, önceden belirlenmiş anlamlılık düzeyinden (örneğin, $\alpha = 0.05$) küçükse, null hipotez reddedilir ve alternatif hipotezin doğru olabileceği kabul edilir.

Adım 1. Araştırma Sorusu Belirlenir

Bir palet çikolata barımız olduğunu ve bunların **Ankara Çikolatacısı**'na ait olup olmadığından emin olmadığımızı varsayalım. Bu çikolata barlarının ağırlıkları, bu barların Ankara Çikolatacısı'na ait olma olasılığı hakkında bize ne söyleyebilir?

```
# Rastgelelik için sabit bir başlangıç noktası belirleme
set.seed(20)

# Normal dağılımdan rastgele veri üretme
choc.palet <- rnorm(
  20,       # Üretilecek rastgele sayıların toplam miktarı
  mean = 42, # Normal dağılımın ortalama değeri
  sd = 2     # Normal dağılımın standart sapması
)

# Veri özet istatistiklerini görüntüleme
summary(choc.palet)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  36.22   40.81   41.41   41.62   42.87   45.57
```

Araştırma sorusu: "Bu çikolata barlarının ağırlıkları, onların Ankara Çikolatacısı'na ait olduğunu gösteriyor mu?"

Adım 2. Null Hipotezi (H_0), ve Alternatif Hipotez (H_1) Belirlenir

Araştırma sorusuna dayanarak, null ve alternatif hipotezler oluşturulur. Örneğin, Adım 1'deki araştırma sorusunu cevaplamak için, iki grup arasında bir karşılaştırma yapmamız gerekir: Ankara Çikolataları'ndan olduğu varsayılan çikolata barlarının ağırlıkları ve diğer çikolata barlarının ağırlıkları. Bu durumda, her iki grubun popülasyon ortalamalarını sırasıyla μ_1 ve μ_2 olarak ifade edilir.

> **i** Yunan Alfabesi ve istatistik Sembolleri
>
> Yunan alfabesindeki μ (*mu*) harfi istatistikte popülasyon ortalamasını bembolize etmektedir. Buna benzer olarak, σ (sigma), α (*alfa*) gibi semboller de kullanılmaktadır. Sembollerle alakalı daha fazla bilgi edinmek için: https://mathvault.ca/hub/higher-math/math-symbols/probability-statistics-symbols/

Null hipotez H_0, genellikle "**etki yok**" ya da "**fark yok**" ifadesini içeren bir parametre durumu olmalıdır. Bu örnekte, null hipotez şu şekilde ifade edilir:

H_0: $\mu_1 = \mu_2$ ya da $\mu_1 - \mu_2 = 0$

> Bu, çikolata barlarının ağırlıklarının Ankara Çikolataları'nın ortalama ağırlığı olan ($\mu_1 = 40$) 40 gramdan **farklı olmadığını** varsayar.

Alternatif hipotez (H_1), genellikle doğru olduğunu düşündüğümüz ya da test etmek istediğimiz iddiayı ifade eder. Bu durumda, alternatif hipotez şu şekilde olabilir:

H_1: $\mu_1 \neq \mu_2$ ya da $\mu_1 - \mu_2 \neq 0$

> Bu, çikolata barlarının ağırlıklarının Ankara Çikolataları'nın ortalama ağırlığından ($\mu_1 = 40$) **farklı** olduğunu ifade eder.

Adım 3. Test İstatistiği ve p-değeri Hesaplanır

5.2.1 Ortalamalar için Hipotez Testleri (Hypothesis Tests for Means)

5.2.1.1 Standart Sapma Biliniyorsa (Known Standard Deviation)

Bir palet çikolata üzerinde, bu çikolataların Ankara Çikolataları'na ait olup olmadığını test etmek isteniyor. Bu amaçla, popülasyon standart sapmasının bilindiği durumda kullanılan **z-testi** yöntemi kullanılır. Test sırasında şu hipotezler tanımlanır:

- **Null hipotez (H_0):** Çikolata paletinde yer alan çikolataların ağırlık ortalaması Ankara Çikolataları'nın ortalaması olan 40 gramdır ($\mu_1 = \mu_2 = 40$).

- **Alternatif hipotez** (H_1): Çikolata paletinde yer alan çikolataların ağırlık ortalaması 40 gramdan farklıdır ($\mu_1 \neq \mu_2 \neq 40$).

Popülasyon standart sapmasının $\sigma = 2$ olduğu bilindiğinden (*adım 1.'de `choc.palet` veri setinde standart sapma 2 (`sd = 2`) olarak belirlenmişti.*), bu değer **z-testi** sırasında kullanılacaktır. Ayrıca, güven düzeyini $1 - \alpha = 0.95$ (yani %95) olarak belirledik. Testi gerçekleştirmek için **BSDA** paketindeki `z.test()` fonksiyonunu kullanılır. Test sonucunda elde edilen **p-değeri**, null hipotezin doğru olduğu varsayımı altında gözlemlenen verilerin veya daha uç sonuçların olasılığını gösterecektir.

Eğer hesaplanan p-değeri, belirlenen anlamlılık düzeyinden ($1 - \alpha = 0.95$, yani yüzde 95 güven aralığı) küçükse, null hipotezi reddedilir. Bu durumda, çikolataların Ankara Çikolataları'na ait olmadığı sonucuna varabilir. Eğer p-değeri büyükse, null hipotezi reddedemeyiz ve çikolataların ağırlıklarının Ankara Çikolataları'na ait olduğu varsayımını destekleyen bir sonuç elde edilmiş olur.

```r
# Gerekli kütüphanenin yüklenmesi
library(BSDA)

# Z-testi ile hipotez testi
z_test2 <- z.test(
  x = choc.palet,      # Test edilecek veri: Çikolata barlarının ağırlıkları
  mu = 40,             # Null hipotezdeki popülasyon ortalaması: 40 gram
  sigma.x = 2,         # Popülasyon standart sapması: 2 gram
  conf.level = 0.95    # Güven düzeyi: %95
)

# Hesaplanan p-değerini görüntüleme
z_test2$p.value
```

```
[1] 0.0002807644
```

Test sonucunda elde edilen **p-değeri = 0.0002807644**, genellikle kabul edilen anlamlılık düzeyi olan $\alpha = 0.05$'ten (hatta $\alpha = 0.01$'den) oldukça küçüktür. Bu, gözlemlenen verilerin null hipotez ($H_0 = \mu_2 = 40$) doğru olduğu varsayımı altında ortaya çıkma olasılığının son derece düşük olduğunu göstermektedir. Bu nedenle, **null hipotez reddedilmektedir**.

Bu sonuca göre, çikolata paletindeki çikolataların ağırlık ortalamasının Ankara Çikolataları'nın standart ağırlık ortalaması olan 40 gramdan **anlamlı derecede farklı** olduğu söylenebilir. Dolayısıyla, bu çikolataların Anakara Çikolataları'na ait olmadığını gözlemlenen farkın tesadüfen oluşma olasılığının çok düşük olduğunu ifade etmektedir.

Sonuç olarak bu test sonuçlarına dayanarak, çikolata paletindeki çikolataların Ankara Çikolataları'na ait olmadığı yönünde güçlü bir kanıya varılabilir.

5.2.1.2 Standart Sapma Bilinmiyorsa (Unknown Standard Deviation)

Bir palet çikolata üzerinde, bu çikolataların Ankara Çikolataları'na ait olup olmadığını test etmek isteniyor. Ancak bu durumda, popülasyon standart sapmasının bilinmediği varsayılmaktadır. Standart sapmanın bilinmediği durumlarda, **t-testi** yöntemi kullanılır ve bu yöntem R'ın temel fonksiyonlarından olan `t.test()` ile kolaylıkla uygulanabilir. Bu yöntem, aşağıdaki formül ile tanımlanan dağılıma dayanır:

$\frac{\bar{X}-\mu}{\frac{s_x}{\sqrt{n}}} \sim t_{n-1}$

Burada t_{n-1} , $n-1$ serbestlik derecesine sahip **Student's t-dağılımı**nı ifade eder. Testi gerçekleştirmek için yalnızca güven düzeyini belirtmek yeterlidir. Örneğin, güven düzeyi $1-\alpha = 0.95$ olarak belirlenebilir. Bu durumda, çikolata paletindeki çikolataların Ankara Çikolataları'na ait olup olmadığı, popülasyon standart sapması bilinmeden değerlendirilmiş olur.

```
# T-testi ile hipotez testi
t_test2 <- t.test(
  x = choc.palet,      # Test edilecek veri: Çikolata paleti ağırlıkları
  mu = 40,             # Null hipotezdeki popülasyon ortalaması: 40 gram
  conf.level = 0.95    # Güven düzeyi: %95
)

# Hesaplanan p-değerini görüntüleme
t_test2$p.value
```

```
[1] 0.003109143
```

Test sonucunda elde edilen **p-değeri = 0.0031**, genellikle kabul edilen anlamlılık düzeyi olan $\alpha = 0.05$'ten küçük bir değerdir. Bu durum, null hipotezin ($H_0 = \mu_2 = 40$) doğru olduğu varsayımı altında, gözlemlenen verilerin veya daha uç sonuçların ortaya çıkma olasılığının oldukça düşük olduğunu göstermektedir. Bu nedenle, **null hipotez reddedilmektedir**.

Bu sonuca göre, çikolata paletindeki çikolataların ağırlık ortalamasının Ankara Çikolataları'nın standart ağırlık ortalaması olan 40 gramdan anlamlı derecede farklı olduğu söylenebilir. Elde edilen düşük p-değeri, bu farkın istatistiksel olarak güçlü bir anlamlılık taşıdığını ve gözlemlenen sonucun rastgele oluşma ihtimalinin oldukça düşük olduğunu ifade etmektedir.

Sonuç olarak, çikolata paletindeki çikolataların Ankara Çikolataları'na ait olmadığı yönünde güçlü bir bulgu elde edilmiştir. Bu çikolataların başka bir üreticiye ait olabileceği değerlendirilmektedir.

5.2.2 İki Örneklem Testleri (two-sample Tests)

5.2.2.1 Birleştirilmemiş İki Örneklem t-testi (Unpooled Two-sample t-test)

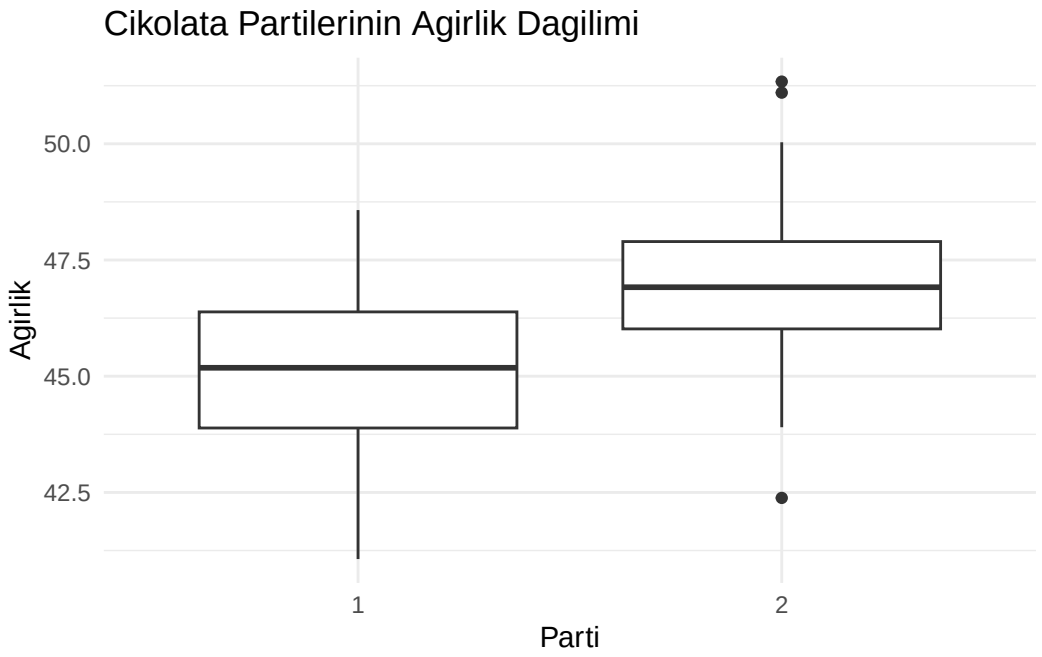
İki farklı çikolata partisini karşılaştırarak, bunların aynı fabrikadan gelip gelmediklerini test etmek istiyoruz. İlk parti 40, ikinci parti 45 paket çikolatadan oluşmaktadır. Popülasyon ortalamaları (μ_1 ve μ_2) ve standart sapmaları bilinmediğinden, bu partiler arasındaki farkı değerlendirmek için **birleştirilmemiş iki örneklem t-testi** uygulanır.

Hipotezler:

- Null hipotez (H_0): İki partinin ortalamaları arasında fark yoktur ($\mu_1 = \mu_2 = 40$).
- Alternatif hipotez (H_1): İki partinin ortalamaları arasında fark vardır ($\mu_1 \neq \mu_2 \neq 40$).

Örneklem verilerini, ortalamaları sırasıyla 45 ve 47 olan normal dağılımlardan üreteceğiz. Ancak, bu bilgilerin bilinmediğini varsayacağız. Ayrıca, örneklem aldığımız dağılımların popülasyon standart sapmalarının 2 olduğu bilinse de, test sırasında bu bilginin de mevcut olmadığını kabul edeceğiz. Gerçek popülasyon ortalamalarını μ_1 ve μ_2 ile ifade edeceğiz.

```r
# Gerekli kütüphanenin yüklenmesi
library(tidyverse)

# Tekrarlanabilirlik için rastgelelik sabitleniyor
set.seed(123)

# Verilerin simülasyonu
parti1 <- rnorm(40, mean = 45, sd = 2) # Birinci partinin ağırlıkları
parti2 <- rnorm(45, mean = 47, sd = 2) # İkinci partinin ağırlıkları

# Veri çerçevesi oluşturma
Parti <- c(rep("1", 40), rep("2", 45)) # Partileri temsil eden faktör değişken
Cikolata <- c(parti1, parti2)          # Birleştirilmiş çikolata ağırlıkları
tablo <- tibble(Parti, Cikolata)

# Boxplot ile görselleştirme
ggplot(tablo, aes(x = Parti, y = Cikolata)) +
  geom_boxplot() +
  labs(
    title = "Cikolata Partilerinin Agirlik Dagilimi",
    x = "Parti",
    y = "Agirlik"
  ) +
  theme_minimal()
```




```r
# Birleştirilmemiş iki örneklem T-testi
t_test3 <- t.test(
  parti1,              # Birinci örneklem verisi
  parti2               # İkinci örneklem verisi
)

# Test sonuçlarını görüntüleme
t_test3
```

```
    Welch Two Sample t-test

data:  parti1 and parti2
t = -4.8753, df = 82.135, p-value = 5.227e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.699807 -1.135072
sample estimates:
mean of x mean of y
 45.09037  47.00781
```

Elde edilen **p-değeri = 5.227e-06**, genellikle kabul edilen anlamlılık düzeyi olan $\alpha = 0.05$'ten ve hatta $\alpha = 0.01$'den bile çok küçüktür. Bu durum, iki çikolata partisinin ağırlık

ortalamalarının aynı olduğu null hipotezin (H_0) doğru olma olasılığının son derece düşük olduğunu göstermektedir. Bu nedenle, **null hipotez reddedilmektedi**r.

Bu sonuca göre, iki çikolata partisi arasında ağırlık ortalamaları açısından anlamlı bir fark olduğu ve partilerin muhtemelen farklı fabrikalardan geldiği söylenebilir.

> **i** e-" nedir?
>
> **"e-"** ifadesi, bilimsel gösterimde kullanılan bir kısaltmadır ve "10 üzeri" anlamına gelir. Örneğin, **5.227e-06** ifadesi, 5.227×10^{-6} yani **0.000005227** anlamına gelir. Bu gösterim, çok küçük veya çok büyük sayıları kolayca ifade etmek için kullanılır.

Bu sefer de $H_0 : \mu_1 = \mu_2$ yerine $H_1 : \mu_1 \leq \mu_2$ hipotezi test edilsin.

```
# Tek yönlü T-testi
t_test4 <- t.test(
  parti1,                # Birinci örneklem verisi
  parti2,                # İkinci örneklem verisi
  alternative = "less")
# Tek yönlü test: Birinci partinin ortalamasının, ikinci partinin ortalamasına
# eşit veya daha küçük olduğunu test eder

# Test sonuçlarını görüntüleme
t_test4
```

```
    Welch Two Sample t-test

data:  parti1 and parti2
t = -4.8753, df = 82.135, p-value = 2.614e-06
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -1.263149
sample estimates:
mean of x mean of y
 45.09037  47.00781
```

P-değeri = 2.614e-06 olduğu için **null hipotez reddedilir**. Birinci partinin ağırlık ortalamasının, ikinci partininkinden anlamlı derecede daha küçük olduğu sonucuna varılır ($\mu_1 \leq \mu_2$).

Bu testte, tek yönlü bir hipotez testi uygulandığı için null hipotezin reddedildiği sonucuna varılmıştır. Tek yönlü testler, pratikte daha yaygındır çünkü veri setleri arasındaki ilişkinin daha odaklı bir açıklamasını sağlar. Örneğin, devlet yardımı alan firmaların ihracat performanslarını değerlendirdiğimizi

düşünelim. Burada, yardımların ihracatı artırıp artırmadığıyla ilgileniyoruz, yani tek yönlü bir alternatif hipotez (H_1 : $ihracatperformansıartar$) kullanırız. Yardımların sadece ihracatı değiştirdiğini, ancak artıp artmadığını bilmediğimiz bir durumda ise iki yönlü bir alternatif hipotez (H_1 : $ihracatperformansıfarklıdır\}$ tercih edilir. Ancak genelde bu tür analizlerde asıl odak, ihracatın artması üzerindedir, bu yüzden tek yönlü test daha uygun olur.

5.2.2.2 Birleştirilmiş İki Örneklem t-testi (Pooled Two-sample t-test)

Eğer örneklemlerin aynı standart sapmaya sahip dağılımlardan geldiğini biliyorsanız, birleştirilmiş iki örneklem t-testi (pooled t-test) uygulanabilir. Bu test, iki grup arasında ortalamalar açısından fark olup olmadığını değerlendirmek için kullanılır ve grup varyanslarının eşit olduğu varsayılır. R'de bu test, t.test fonksiyonunda var.equal = TRUE argümanı ile belirtilir.

```
# Birleştirilmiş iki örneklem T-testi
t_test_pooled <- t.test(
  parti1,           # Birinci parti verisi
  parti2,           # İkinci parti verisi
  var.equal = TRUE  # Varyansların eşit olduğu varsayımı
)

# Test sonuçlarını görüntüleme
t_test_pooled
```

```
    Two Sample t-test

data:  parti1 and parti2
t = -4.8705, df = 83, p-value = 5.254e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.700462 -1.134417
sample estimates:
mean of x mean of y
 45.09037  47.00781
```

1. **t-istatistiği** ($t = -4.8705$):
   - T-testi sonucunda hesaplanan t-değeri, iki grup arasındaki ortalama farkın standart hata cinsinden ne kadar uzakta olduğunu gösterir. Burada elde edilen negatif değer, birinci grubun ortalamasının ikinci grubunkinden daha düşük olduğunu işaret eder.
2. **Serbestlik Derecesi** ($df = 83$):

- Testte kullanılan serbestlik derecesi, veri setlerinin boyutlarına ve test türüne bağlıdır. Burada, iki grubun toplam gözlem sayısından (40 + 45) varyans hesaplamaları için kullanılan parametreler çıkarılarak $df = 83$ elde edilmiştir.

3. **P-değeri** :

   - Elde edilen **p-değeri = 5.254e-06**, kabul edilen anlamlılık düzeyi ($\alpha = 0.05$) ve hatta daha düşük bir düzey olan ($\alpha = 0.05$) ile karşılaştırıldığında oldukça küçüktür. Bu, null hipotezin (H_0: iki grup ortalaması arasında fark yoktur) red-dedilmesine neden olur.
   - Sonuç: İki grup arasında ortalamalar açısından **istatistiksel olarak anlamlı bir fark** vardır.

4. **Güven Aralığı**:

   - %95 güven düzeyi ile, iki grup arasındaki ortalama farkın popülasyonda −2.7-2.7−2.7 ile −1.13-1.13−1.13 arasında olduğu tahmin edilmektedir.
   - Negatif aralık, birinci grubun ortalamasının ikinci grubunkinden daha düşük olduğunu teyit eder.

5. **Örneklem Ortalamaları**

   - Birinci grup (x): Ortalama ağırlık **45.09**.
   - İkinci grup (y): Ortalama ağırlık **47.01**.
   - Bu fark, test sonuçlarında da anlamlı bulunmuştur.

5.2.3 Sonuç:

Bu test, iki çikolata partisi arasındaki ağırlık ortalamalarının anlamlı derecede farklı olduğunu göstermektedir. Elde edilen negatif değerler, birinci partinin ortalamasının ikinci partininkinden daha düşük olduğunu desteklemektedir. İstatistiksel olarak anlamlı bu fark, partilerin muhtemelen farklı fabrikalardan geldiğini düşündürmektedir.

Referans

https://rpubs.com/syedafzalali/R3

https://uw-statistics.github.io/Stat311Tutorial/confidence-intervals.html

https://uw-statistics.github.io/Stat311Tutorial/hypothesis-tests.html

https://www.modernstatisticswithr.com/basicstatistics.html

https://openintro-ims.netlify.app/foundations-of-inference

https://bookdown.org/pbaumgartner/swr-harris/

Part II

Veri Dönüşümleri

Veri dönüşümleri, ham verinin analiz ve modelleme için uygun hale getirilmesi sürecinde temel bir adımdır. Bu başlık altında, veri manipülasyonu ve dönüşüm tekniklerine ilişkin kapsamlı bir içerik sunulmaktadır. Kitabın bu kısmında, sütun seçimi, satır filtreleme, sıralama ve sütun adlarının yeniden adlandırılması gibi temel veri manipülasyonu işlemlerinden başlayarak, yeni değişken oluşturma, istatistiksel özetler hazırlama ve veri birleştirme gibi daha karmaşık süreçlere kadar geniş bir yelpazede yöntemler ele alınmıştır. Bu yöntemlerin, R programlama dili kullanılarak pratik uygulamalarla nasıl gerçekleştirilebileceği adım adım açıklanmıştır.

Veri manipülasyonu işlemlerinin ardından, veri şekillendirme ve metin manipülasyonu gibi daha spesifik konular ele alınmaktadır. Veri çerçevelerinin birleştirilmesi, veri ekleme işlemleri ve metinsel bilgilerin düzenlenmesi, modern veri analizi süreçlerinde sıkça karşılaşılan problemler için çözüm önerileri sunmaktadır. Özellikle, düzenli ifadeler (regex) ve janitor paketi gibi araçlar kullanılarak veri temizleme ve düzenleme süreçleri sade bir şekilde anlatılmıştır. Bu teknikler, büyük ve karmaşık veri setlerinin daha anlaşılır ve işlenebilir bir formata dönüştürülmesini sağlamaktadır.

Çalışmanın bu kısmı ayrıca eksik verilerle çalışmaya yönelik detaylı yöntemler içermektedir. Eksik verilerin tespiti, görselleştirilmesi ve doldurulması gibi süreçler, analizlerin doğruluğunu artırmak ve tutarlı sonuçlar elde etmek için kritik bir öneme sahiptir. Bu bağlamda, DLOOKR ve MICE gibi R paketlerinin kullanımı uygulamalı örneklerle gösterilmiştir. Veri dönüşümüne yönelik bu kapsamlı yaklaşım, veri analizi süreçlerinde verilerin etkin bir şekilde işlenmesine ve analizlerden maksimum verim alınmasına olanak tanımaktadır.

6 Veri Manipülasyonu

6.1 Temel Veri Manipülasyonu İşlemleri

6.1.1 Sütun Seçimi: `select()`

Veri analizi sırasında, genellikle yalnızca belirli sütunlarla çalışmak ya da analiz için gereksiz sütunları veri setinden çıkarmak gerekebilir. R dilinde, sütun seçimi ve yönetimi için en yaygın kullanılan araçlardan biri, `dplyr` paketinin sunduğu **`select()`** fonksiyonudur. Bu fonksiyon, kolay ve esnek bir şekilde sütunları seçmeyi, sıralamayı veya hariç tutmayı mümkün kılar.

`select()` fonksiyonunu kullanırken yalnızca sütun isimlerini belirterek seçim yapabilirsiniz. Ayrıca, sütun seçim işlemini kolaylaştırmak için çeşitli yardımcı fonksiyonlar da kullanılabilir. Bu yardımcı fonksiyonlar, sütun adlarını desenlere, pozisyonlara veya belirli kurallara göre seçmeyi sağlar.

- **Örnek Veri: `starwars` Veri Seti**

Bu eğitimde, sütun seçimi işlemlerini öğrenirken, `dplyr` paketinde yer alan `starwars` veri setinin ilk 5 satırı ve ilk 6 sütunundan oluşan bir alt kümesini kullanacağız. Bu veri seti, sütun seçim işlemlerini göstermek için çeşitli veri türlerini ve isimlendirme desenlerini içeren harika bir örnek sağlar.

Aşağıdaki adımlarda, hem belirli sütunları seçme hem de sütunları hariç tutma işlemlerini nasıl gerçekleştireceğimizi öğrenirken yukarıda bahsedilen yardımcı fonksiyonları pratikte göreceğiz.

```
# Gerekli paketin yüklenmesi
# install.packages("dplyr")
library(dplyr)

# Örnek veri seti oluşturma
df <- starwars[1:5, 1:6]
# starwars veri setinin ilk 5 satırını ve ilk 6 sütununu seçerek bir alt küme
# oluşturur ve bunu df adlı bir nesneye atar.

# Veri setini görüntüleme
df # Bundan sonra df veri seti kullanılacaktır.
```

```
# A tibble: 5 x 6
  name           height  mass hair_color skin_color  eye_color
  <chr>           <int> <dbl> <chr>      <chr>       <chr>
1 Luke Skywalker    172    77 blond      fair        blue
2 C-3PO             167    75 <NA>       gold        yellow
3 R2-D2              96    32 <NA>       white, blue red
4 Darth Vader       202   136 none       white       yellow
5 Leia Organa       150    49 brown      light       brown
```

Starwars Veri Seti

`starwars` veri seti, Star Wars evrenindeki karakterlerin fiziksel özelliklerini ve kimlik bilgilerini içeren bir veri setidir. Bu veri seti, çeşitli değişkenlerle karakterlerin boy, kilo, saç rengi gibi fiziksel özelliklerini ve göz rengi gibi detaylarını sunar. Seçilen veri seti (**ilk 5 satır ve ilk 6 sütun**), aşağıdaki değişkenleri içerir:

- **name**: Karakterin adı.
- **height**: Boy ölçümlerini içerir (santimetre).
- **mass**: Kilo ölçümlerini içerir (kilogram).
- **hair_color**: Saç rengini içerir.
- **skin_color**: Ten rengini içerir.
- **eye_color**: Göz rengini içerir.

`select()` fonksiyonu, bir veri çerçevesinden belirli sütunları seçmek veya hariç tutmak için kullanılır. Sütun seçimi, hem sütun isimleri hem de sütunların konum/indeks bilgileri kullanılarak yapılabilir.

R'da Köşeli Parantez

R'de `[` işareti, bir nesne içinden belirli elemanları seçmek için kullanılan bir alt kümeleme operatörüdür. Veri çerçeveleri, matrisler, vektörler ve listeler üzerinde alt kümeleme yapmak için kullanılır.

- **Vektörlerde Alt Kümeleme:**

```r
# Bir vektör oluşturma
v <- c(10, 20, 30, 40)

# Tek bir eleman seçme
v[2]  # Sonuç: 20 (2. eleman)
```

```
[1] 20
```

```r
# Birden fazla eleman seçme
v[c(1, 3)]  # Sonuç: 10, 30 (1. ve 3. eleman)
```

```
[1] 10 30
```

- **Veri Çerçevelerinde Alt Kümeleme:**

```r
# Bir örnek data frame oluşturma
df_ornek <- data.frame(
  a = 1:3, # Birinci sütun: 1, 2, 3
  b = 4:6  # İkinci sütun: 4, 5, 6
)

# 1. satıra erişim
df_ornek[1, ]  # 1. satır
```

```
  a b
1 1 4
```

```r
# "a" sütununa erişim
df_ornek[, "a"]  # "a" sütunu
```

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

```r
# İlk 2 satır ve "a", "b" sütunlarına erişim
df_ornek[1:2, c("a", "b")]  # İlk 2 satır, "a" ve "b" sütunları
```

```
  a b
1 1 4
2 2 5
```

6.1.1.1 İsimle Seçim (By Name)

Bir veri seti üzerinde çalışırken, **`select`** fonksiyonu içinde seçmek istediğiniz sütunları belirtebilirsiniz. Sütun isimlerini tırnak işaretleriyle (") veya tırnak işareti olmadan yazabilirsiniz. Tek bir sütun ya da birden fazla sütunu seçmeniz mümkündür.

- **Belirli Sütunların Seçilmesi**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Star Wars veri setinden alt küme oluşturma
```

```r
df <- starwars[1:5, 1:6]

# Belirli sütunların seçimi
df_2 <- df %>%
  select(name, height)
# Yalnızca "name" ve "height" sütunlarını seçer.

# Yeni veri setini görüntüleme
df_2
```

```
# A tibble: 5 x 2
  name           height
  <chr>           <int>
1 Luke Skywalker    172
2 C-3PO             167
3 R2-D2              96
4 Darth Vader       202
5 Leia Organa       150
```

Pipe Operatörü Nedir ve Nasıl Çalışır?

Pipe operatörü (%>%), R programlama dilinde bir işlemin sonucunu otomatik olarak bir sonraki işleme girdi olarak aktarır. Bu operatör, özellikle veri manipülasyonu işlemlerini daha okunabilir, düzenli ve anlaşılır hale getirmek için kullanılır. Pipe, bir işlemden gelen veriyi diğerine "aktararak" adım adım bir işlem zinciri oluşturmayı sağlar.
Neden Kullanılır?

1. **Kod Okunabilirliğini Artırır:** Pipe operatörüyle yazılmış kodda, işlemler sırasıyla ve kolayca takip edilebilir. Bu, özellikle uzun ve karmaşık veri işleme süreçlerinde büyük bir avantaj sağlar.
2. **Ara Değişkenleri Ortadan Kaldırır:** Pipe kullanımı, her işlem sonucu için ayrı bir değişken tanımlama ihtiyacını ortadan kaldırır, böylece kod daha sade hale gelir.
3. **Adım Adım İşlem Zinciri Kurar:** Birden fazla işlemi arka arkaya uygulamak gerektiğinde pipe operatörü ile bu işlemler kolayca zincirlenir.

Nasıl Çalışır?
Pipe operatörü, bir nesneyi (örneğin bir veri çerçevesini) bir fonksiyonun girdisi olarak aktarır. Bu sayede kod, "bu işlemden gelen sonucu şu işleme aktar" şeklinde yazılır. Her işlem bir öncekinin sonucunu alır ve yeni bir işlem yapar.
Özetle, pipe operatörü veriyi işlemler arasında taşımak için kullanılır ve kod yazımını hem daha kısa hem de daha anlaşılır hale getirir.

- **Belirli Sütun Aralığının Seçilmesi**

Sütunlar arasında bir dizi seçmek için `:` operatörünü kullanabilirsiniz. Örneğin, `height` sütunundan başlayarak `skin_color` sütununa kadar (her iki sütun da dahil) olan sütunları seçmek isterseniz, şu şekilde yazabilirsiniz:

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# 'height' sütunundan 'skin_color' sütununa kadar olan sütunları seçme
df_2 <- df %>%
  select(height:skin_color)
# Bu ifade, df veri setinden "height" sütunundan başlayıp "skin_color" sütununa kadar
# (her iki sütun dahil) olan sütunları seçer.

# Seçilen sütunları görüntüleme
df_2
```

```
# A tibble: 5 x 4
  height  mass hair_color skin_color
   <int> <dbl> <chr>      <chr>
1    172    77 blond      fair
2    167    75 <NA>       gold
3     96    32 <NA>       white, blue
4    202   136 none       white
5    150    49 brown      light
```

6.1.1.2 İndeks ile Seçim (By Index)

- **Belirli Sütunların İndeks Numarası ile Seçilmesi**

Sütunlar indeks (konum) numaralarına göre de seçilebilir. Bunun için **`select`** fonksiyonu içinde istenilen sütun numaralarını belirtmeniz yeterlidir. Aşağıdaki örnek, birinci, beşinci ve altıncı sütunları seçmektedir:

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
```

```r
df <- starwars[1:5, 1:6]

# Belirli sütunları indeks numarasına göre seçme
df_2 <- df %>%
  select(1, 5, 6)
# Bu ifade, df veri setinden birinci, beşinci ve altıncı sütunları
# indeks numaralarına göre seçer.

# Seçilen sütunları görüntüleme
df_2
```

```
# A tibble: 5 x 3
  name           skin_color  eye_color
  <chr>          <chr>       <chr>
1 Luke Skywalker fair        blue
2 C-3PO          gold        yellow
3 R2-D2          white, blue red
4 Darth Vader    white       yellow
5 Leia Organa    light       brown
```

6.1.1.3 Sütunları Hariç Tutma (Drop Columns)

- **Belirli Bir Sütunu Hariç Tutma**

`select` fonksiyonu, belirli sütunları hariç tutmak (çıkarmak) için de kullanılabilir. Bunun için, sütun isimlerinin önüne `-` sembolünü eklemek yeterlidir. Aşağıdaki örnekte, `mass` sütunu hariç tüm sütunlar seçilmektedir:

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Belirli bir sütunu hariç tutma
df_2 <- df %>%
  select(-mass)
# Bu ifade, df veri setinden "mass" sütununu hariç tutar
# ve geri kalan tüm sütunları seçer.
```

```r
# Seçilen sütunları görüntüleme
df_2
```

```
# A tibble: 5 x 5
  name           height hair_color skin_color  eye_color
  <chr>           <int> <chr>      <chr>       <chr>
1 Luke Skywalker    172 blond      fair        blue
2 C-3PO             167 <NA>       gold        yellow
3 R2-D2              96 <NA>       white, blue red
4 Darth Vader       202 none       white       yellow
5 Leia Organa       150 brown      light       brown
```

- **Birden Fazla Sütunu Hariç Tutma**

Birden fazla sütunu çıkarmak istediğiniz durumlarda, her sütun adının önüne - ekleyebilir ya da sütun adlarını içeren bir vektörün önüne - koyabilirsiniz.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Birden fazla sütunu hariç tutma
df_2 <- df %>%
  select(-height, -mass, -hair_color)
# Alternatif olarak: select(-c(height, mass, hair_color))

# Seçilen sütunları görüntüleme
df_2
```

```
# A tibble: 5 x 3
  name           skin_color  eye_color
  <chr>          <chr>       <chr>
1 Luke Skywalker fair        blue
2 C-3PO          gold        yellow
3 R2-D2          white, blue red
4 Darth Vader    white       yellow
5 Leia Organa    light       brown
```

6.1.1.4 Sütunları Seçmek veya Çıkarmak İçin Yardımcı Fonksiyonlar

- **Belirli Bir Kelimeyi İçeren Sütunları Seçme**

Belirli desenlere veya koşullara dayalı olarak sütunları seçmek için çeşitli yardımcı fonksiyonlar bulunmaktadır. Bu fonksiyonlar şunları içerir:

- `contains`: Belirli bir kelimeyi içeren sütunları seçer.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# "color" kelimesini içeren sütunları seçme
df_contains_color <- df %>%
  select(contains("color")) # "color" kelimesini içeren sütunları seçer.

# Sonucu görüntüleme
df_contains_color
```

```
# A tibble: 5 x 3
  hair_color skin_color  eye_color
  <chr>      <chr>       <chr>
1 blond      fair        blue
2 <NA>       gold        yellow
3 <NA>       white, blue red
4 none       white       yellow
5 brown      light       brown
```

- **Belirli Bir Metin ile Başlayan Sütunları Seçme**
- `starts_with`: Belirli bir metinle başlayan sütunları seçer.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# "h" harfiyle başlayan sütunları seçme
df_starts_with_h <- df %>%
```

```
  select(starts_with("h")) # "h" harfiyle başlayan sütunları seçer.

# Sonucu görüntüleme
df_starts_with_h
```

```
# A tibble: 5 x 2
  height hair_color
   <int> <chr>
1    172 blond
2    167 <NA>
3     96 <NA>
4    202 none
5    150 brown
```

- **Belirli Bir Metin ile Biten Sütunları Seçme**
- `ends_with`: Belirli bir metinle biten sütunları seçer.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# "t" harfiyle biten sütunları seçme
df_ends_with_t <- df %>%
  select(ends_with("t")) # "t" harfiyle biten sütunları seçer.

# Sonucu görüntüleme
df_ends_with_t
```

```
# A tibble: 5 x 1
  height
   <int>
1    172
2    167
3     96
4    202
5    150
```

- **Son Sütunu Seçme**
- `last_col`: Son sütunu seçer.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Son sütunu seçme
df_last_col <- df %>%
  select(last_col()) # Veri setindeki son sütunu seçer.

# Sonucu görüntüleme
df_last_col
```

```
# A tibble: 5 x 1
  eye_color
  <chr>
1 blue
2 yellow
3 red
4 yellow
5 brown
```

- **Belirli Kelimeleri İçeren Sütunları Regex ile Seçme**
- `matches`: Regex desenine uyan sütunları seçer.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# "name" veya "mass" kelimelerini içeren sütunları seçme
df_matches_name_mass <- df %>%
  select(matches("name|mass"))
# "name" veya "mass" kelimelerini içeren sütunları regex (düzenli ifade) ile seçer.

# Sonucu görüntüleme
df_matches_name_mass
```

```
# A tibble: 5 x 2
  name           mass
```

```
  <chr>          <dbl>
1 Luke Skywalker    77
2 C-3PO             75
3 R2-D2             32
4 Darth Vader      136
5 Leia Organa       49
```

- **Numara Aralığı ile Sütun Seçme**
- `num_range`: Numara aralığına göre sütun seçimi yapar.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Numara aralığına göre sütun seçme
df_num_range <- df %>%
  select(num_range("col", 1:2))
# "col1", "col2" gibi sütun isimlerine uyan numara aralığını seçer.
# Ancak bu veri setinde "col1" veya "col2" isimli sütunlar olmadığı için
# sonuç boş olacaktır.

# Sonucu görüntüleme
df_num_range
```

```
# A tibble: 5 x 0
```

- **Tüm Sütunları Seçme**
- `everything`: Veri setindeki tüm sütunları seçer.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Tüm sütunları seçme
df_everything <- df %>%
  select(everything()) # Veri setindeki tüm sütunları seçer.
```

```
# Sonucu görüntüleme
df_everything
```

```
# A tibble: 5 x 6
  name           height  mass hair_color skin_color  eye_color
  <chr>           <int> <dbl> <chr>      <chr>       <chr>
1 Luke Skywalker    172    77 blond      fair        blue
2 C-3PO             167    75 <NA>       gold        yellow
3 R2-D2              96    32 <NA>       white, blue red
4 Darth Vader       202   136 none       white       yellow
5 Leia Organa       150    49 brown      light       brown
```

- **Yalnızca Sayısal Sütunları Seçme**
- **`where`**: Belirli bir koşulu sağlayan sütunları seçer.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Yalnızca sayısal sütunları seçme
df_numeric_cols <- df %>%
  select(where(is.numeric)) # Sadece sayısal veri tipine sahip sütunları seçer.

# Sonucu görüntüleme
df_numeric_cols
```

```
# A tibble: 5 x 2
  height  mass
   <int> <dbl>
1    172    77
2    167    75
3     96    32
4    202   136
5    150    49
```

- **Sütunları Belirli Bir Sıralamayla Düzenleme**

`select()` fonksiyonu yalnızca sütunları seçmek için değil, aynı zamanda sütunların sıralamasını değiştirmek için de kullanılabilir. Sütunları belirli bir düzene göre sıralamak istediğinizde, sütun isimlerini veya yardımcı fonksiyonları sırayla belirterek veri setinizin sütun yapısını yeniden düzenleyebilirsiniz.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri seti
df <- starwars[1:5, 1:6]

# Sütunları sıralama
df_sorted <- df %>%
  select(
    name,                      # İlk olarak 'name' sütunu
    ends_with("_color"),       # Sonra sonu "_color" ile biten sütunlar
    everything()               # En son diğer tüm sütunlar
  )

# Sonucu görüntüleme
df_sorted
```

```
# A tibble: 5 x 6
  name           hair_color skin_color  eye_color height  mass
  <chr>          <chr>      <chr>       <chr>      <int> <dbl>
1 Luke Skywalker blond      fair        blue         172    77
2 C-3PO          <NA>       gold        yellow       167    75
3 R2-D2          <NA>       white, blue red           96    32
4 Darth Vader    none       white       yellow       202   136
5 Leia Organa    brown      light       brown        150    49
```

6.1.2 Satır Filtreleme: `filter`

`filter` fonksiyonu, bir veri çerçevesindeki satırları belirli bir veya birden fazla koşula göre alt kümeye ayırmak için kullanılır. Bu fonksiyon, hem karşılaştırma hem de mantıksal operatörlerle esnek bir şekilde çalışarak, veri setinden yalnızca belirli kriterlere uyan satırları seçmeyi sağlar.

- *Örnek Veri: `women` Veri Seti**

Bu eğitimde, filtreleme işlemlerini öğrenirken R içinde yer alan **`women`** veri setini kullanacağız. Bu veri seti, kadınlara ait boy ve kilo ölçümlerini içeren iki sayısal sütundan oluşur. Boy ve kilo arasındaki ilişkileri incelemek ve koşullara göre filtreleme işlemlerini göstermek için ideal bir örnek teşkil eder.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri çerçevesi
df <- as_tibble(women)
# women veri setini tibble formatına çevirir ve df nesnesine atar.

# Veri çerçevesini görüntüleme
head(df, 10)
```

```
# A tibble: 10 x 2
   height weight
    <dbl>  <dbl>
 1     58    115
 2     59    117
 3     60    120
 4     61    123
 5     62    126
 6     63    129
 7     64    132
 8     65    135
 9     66    139
10     67    142
```

Women Veri Seti

`women` veri seti, kadınlara ait boy ve kilo ölçümlerini içeren bir veri setidir. Bu veri seti, iki sayısal değişken ile kadınların fiziksel özelliklerini sunar.

- **height**: Kadınların boy ölçümlerini içerir (inç cinsinden).
- **weight**: Kadınların kilo ölçümlerini içerir (pound cinsinden).

6.1.2.1 Tek Bir Koşula Dayalı Satır Filtreleme

`filter` fonksiyonu, bir veri çerçevesindeki satırları belirli bir koşula göre alt kümeye ayırmak için kullanılır. Bu fonksiyon sayesinde, değerlerin belirli bir değere eşit olup olmadığını, daha büyük veya küçük olduğunu, ya da belirli bir aralıkta olup olmadığını kontrol ederek veri filtreleme işlemi yapılabilir.

- **R'deki Karşılaştırma Operatörleri**

Aşağıdaki tablo, R'deki karşılaştırma operatörlerini ve açıklamalarını içerir:

Karşılaştırma Operatörü	Açıklama
>	Daha büyük
<	Daha küçük
>=	Daha büyük veya eşit
<=	Daha küçük veya eşit
==	Eşit
!=	Eşit değil

- *Belirli Bir Koşula Göre Satırları Filtreleme**

Aşağıdaki örnekte, `women` veri setinden **`height`** sütununda değeri 68'den büyük olan satırları filtreliyoruz. Bu işlem, yalnızca boyu belirtilen değerden daha büyük olan kadınlara ait bilgileri seçmek için kullanılır.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# height sütununda 68'den büyük olan değerleri filtreleme
df_2 <- df %>%
  filter(height > 68)
# height sütununda  68'den büyük olan değerlerin bulunduğu satırları seçer.

# Filtrelenmiş veri çerçevesini görüntüleme
df_2
```

```
# A tibble: 4 x 2
  height weight
   <dbl>  <dbl>
1     69    150
2     70    154
3     71    159
4     72    164
```

- **Ortalama Değere Göre Satırları Filtreleme**

`filter` fonksiyonu, yalnızca sabit bir değere değil, aynı zamanda bir fonksiyonun çıktısına dayalı olarak da satırları filtreleyebilir. Örneğin, bir sütunun değerlerini, o sütunun ortalamasıyla karşılaştırarak filtreleme yapılabilir.

Aşağıdaki örnekte, **height** sütununda değeri sütunun ortalamasına eşit veya daha düşük olan satırlar filtrelenmektedir.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# height sütununda ortalamaya eşit veya daha düşük olan değerleri filtreleme
df_2 <- df %>%
  filter(height <= mean(height))
# height sütununda, değeri sütunun ortalamasına eşit veya daha düşük olan
# satırları seçer.

# Filtrelenmiş veri çerçevesini görüntüleme
df_2
```

```
# A tibble: 8 x 2
  height weight
   <dbl>  <dbl>
1     58    115
2     59    117
3     60    120
4     61    123
5     62    126
6     63    129
7     64    132
8     65    135
```

6.1.2.2 Mantıksal Operatörler ve Fonksiyonlarla Satır Filtreleme

`filter` fonksiyonu, mantıksal operatörler veya `TRUE` ya da `FALSE` döndüren fonksiyonlarla birlikte kullanılarak daha karmaşık filtreleme işlemleri yapabilir. Aşağıdaki tabloda, R'de sık kullanılan mantıksal operatörler ve fonksiyonlar açıklanmaktadır:

Operatör/Fonksiyon	Açıklama
`!`	Mantıksal değil ('NOT')
`%in%`	Belirtilen kümenin içinde
`!(x %in% y)`	Belirtilen kümenin içinde olmayanlar
`is.na()`	Değer NA olanlar

Operatör/Fonksiyon	Açıklama
`!is.na()`	Değer NA olmayanlar
`grepl()`	Belirtilen bir deseni içerenler
`!grepl()`	Belirtilen bir deseni içermeyenler

- **Belirli Değerlere Göre Satırları Filtreleme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
df <- as_tibble(women)

# 'height' sütununda değeri 65, 70 veya 72 olan satırları seçme
df_2 <- df %>%
  filter(height %in% c(65, 70, 72))
# height sütununda 65, 70 veya 72 olan satırları filtreler.

# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
  height weight
   <dbl>  <dbl>
1     65    135
2     70    154
3     72    164
```

- **Belirli Değerlere Sahip Olmayan Satırları Filtreleme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
df <- as_tibble(women)

# 'height' değeri 65, 70 veya 72 olmayan satırları seçme
df_2 <- df %>%
  filter(!(height %in% c(65, 70, 72)))
# Bu ifade, height sütununda değeri 65, 70 veya 72 olmayan satırları seçer.
```

```
# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 12 x 2
   height weight
    <dbl>  <dbl>
 1     58    115
 2     59    117
 3     60    120
 4     61    123
 5     62    126
 6     63    129
 7     64    132
 8     66    139
 9     67    142
10     68    146
11     69    150
12     71    159
```

- **Belirli Bir Rakamı veya Deseni İçeren Satırları Filtreleme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# height sütununda "5" rakamını içeren satırları filtreleme
df_2 <- df %>%
  filter(grepl("5", height))
# Bu ifade, height sütununda "5" rakamını içeren satırları seçer.

# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
  height weight
   <dbl>  <dbl>
1     58    115
2     59    117
3     65    135
```

6.1.2.3 Birden Fazla Koşula Dayalı Satır Filtreleme

Bir veri çerçevesinde satırları filtrelerken birden fazla koşul kullanılabilir. Örneğin, belirli bir aralıkta bulunan değerleri seçmek veya tarih aralığında veri filtrelemek gibi işlemler yapılabilir. Bunun için mantıksal operatörler kullanılır.

6.1.2.4 R'deki Mantıksal Operatörler ve Açıklamaları

Mantıksal Operatör	Açıklama
`&`	Eleman bazında mantıksal **"VE" (AND)**
`|`	Eleman bazında mantıksal **"VEYA" (OR)**
`xor()`	Eleman bazında kapsamlı mantıksal `!(x | y)`

- **İki Koşulu Aynı Anda Sağlayan Satırları Filtreleme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# 'height' değeri 65'ten büyük VE 68'den küçük olan satırları filtreleme
df_2 <- df %>%
  filter(height > 65 & height < 68)
# Bu ifade, height sütununda değeri 65'ten büyük ve 68'den küçük olan
# satırları seçer.

# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 2 x 2
  height weight
   <dbl>  <dbl>
1     66    139
2     67    142
```

- **Çoklu Koşul ile Satırları Filtreleme**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# 'height' sütunu 65'ten büyük ve 'weight' sütunu 150'den küçük veya
# eşit olan satırları filtreleme
df_2 <- df %>%
  filter(height > 65 & weight <= 150)
# Bu ifade, height sütunu 65'ten büyük ve weight sütunu 150'den küçük veya
# eşit olan satırları seçer.

# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 4 x 2
  height weight
   <dbl>  <dbl>
1     66    139
2     67    142
3     68    146
4     69    150
```

- **Mantıksal "VEYA" Koşulu ile Satırları Filtreleme**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# 'height' sütunu 65'ten büyük VEYA 'weight' sütunu 150'den büyük veya
# eşit olan satırları filtreleme
df_2 <- df %>%
  filter(height > 65 | weight >= 150)
# Bu ifade, height sütunu 65'ten büyük VEYA weight sütunu 150'den büyük veya
# eşit olan satırları seçer.

# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 7 x 2
  height weight
   <dbl>  <dbl>
1     66    139
2     67    142
3     68    146
4     69    150
5     70    154
6     71    159
7     72    164
```

6.1.2.5 Satır Numarasına Göre Filtreleme: `slice`

`filter` fonksiyonuna benzer bir işlev de **`slice`** fonksiyonudur. Bu fonksiyon, satırları **indekslerine/pozisyonlarına** göre filtrelemeye olanak tanır. Girdi olarak bir sıra veya indekslerin bulunduğu bir vektör (**tam sayı değerleri**) alır. Kullanımı aşağıda gösterilmiştir.

- **Belirli Satırları Seçme**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# İlk 3 satırı seçme
df_2 <- df %>%
  slice(1:3) # Bu ifade, df veri setinin ilk 3 satırını seçer.

# Filtrelenmiş veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
  height weight
   <dbl>  <dbl>
1     58    115
2     59    117
3     60    120
```

- **İlk N Satırı Seçme**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# İlk 3 satırı seçme
df_slice_head <- df %>%
  slice_head(n = 3) # İlk 3 satır seçilir.

# Sonucu görüntüleme
df_slice_head
```

```
# A tibble: 3 x 2
  height weight
   <dbl>  <dbl>
1     58    115
2     59    117
3     60    120
```

- **Son N Satırı Seçme**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# Son 3 satırı seçme
df_slice_tail <- df %>%
  slice_tail(n = 3) # Son 3 satır seçilir.

# Sonucu görüntüleme
df_slice_tail
```

```
# A tibble: 3 x 2
  height weight
   <dbl>  <dbl>
1     70    154
2     71    159
3     72    164
```

- **Rastgele Satırlar Seçme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# Rastgele 3 satırı seçme
df_slice_sample <- df %>%
  slice_sample(n = 3) # Rastgele 3 satır seçilir.

# Sonucu görüntüleme
df_slice_sample
```

```
# A tibble: 3 x 2
  height weight
   <dbl>  <dbl>
1     65    135
2     64    132
3     58    115
```

- **Belirli Bir Sütuna Göre En Küçük Satırları Seçme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# height sütununa göre en küçük 2 satırı seçme
df_slice_min <- df %>%
  slice_min(height, n = 2) # height sütunundaki en küçük 2 satır seçilir.

# Sonucu görüntüleme
df_slice_min
```

```
# A tibble: 2 x 2
  height weight
   <dbl>  <dbl>
1     58    115
2     59    117
```

- **Belirli Bir Sütuna Göre En Büyük Satırları Seçme**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# women veri setini tibble formatına çevirme
df <- as_tibble(women)

# weight sütununa göre en büyük 2 satırı seçme
df_slice_max <- df %>%
  slice_max(weight, n = 2) # weight sütunundaki en büyük 2 satır seçilir.

# Sonucu görüntüleme
df_slice_max
```

```
# A tibble: 2 x 2
  height weight
   <dbl>  <dbl>
1     72    164
2     71    159
```

6.1.3 Satırların Sıralanması: `arrange()`

arrange, dplyr paketinde kullanılan ve bir veri çerçevesindeki satırları bir veya daha fazla sütunun değerlerine göre yeniden sıralamayı sağlayan bir fonksiyondur. Varsayılan olarak, satırları **artan düzende** sıralar. **Azalan sıralama** yapmak için **desc** fonksiyonu kullanılır.

starwars Veri Seti

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
df <- starwars[1:10, c(1, 2, 3, 11)]
# starwars veri setinin ilk 10 satırını ve 1, 2, 3 ve 11. sütunlarını seçerek bir alt küme ol

# Veriyi görüntüleme
df
```

```
# A tibble: 10 x 4
   name               height  mass species
```

```
   <chr>               <int> <dbl> <chr>
 1 Luke Skywalker        172    77 Human
 2 C-3PO                 167    75 Droid
 3 R2-D2                  96    32 Droid
 4 Darth Vader           202   136 Human
 5 Leia Organa           150    49 Human
 6 Owen Lars             178   120 Human
 7 Beru Whitesun Lars    165    75 Human
 8 R5-D4                  97    32 Droid
 9 Biggs Darklighter     183    84 Human
10 Obi-Wan Kenobi        182    77 Human
```

```
# Bundan sonra df veri seti kullanılacaktır.
```

Starwars Veri Seti

starwars veri seti, Star Wars evrenindeki karakterlerin fiziksel özelliklerini ve kimlik bilgilerini içeren bir veri setidir. Bu veri seti, çeşitli değişkenlerle karakterlerin boy, kilo, cinsiyet gibi fiziksel özelliklerini ve isim gibi kimlik detaylarını sunar. Seçilen veri seti (ilk 10 satır ve 1, 2, 3, 11. sütunlar), aşağıdaki değişkenleri içerir:

- **name**: Karakterin adı.
- **height**: Boy ölçümlerini içerir (santimetre).
- **mass**: Kilo ölçümlerini içerir (kilogram).
- **gender**: Karakterin cinsiyet bilgisi.

6.1.3.1 Tek Bir Sütuna Göre Sıralama

- **Bir Sütuna Göre Artan Sıralama**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# starwars veri setinden alt küme oluşturma
df <- starwars[1:10, c(1, 2, 3, 11)]

# 'height' sütununa göre artan (ASCENDING) sıralama
df_2 <- df %>%
  arrange(height) # height sütununa göre satırları artan sırayla yeniden düzenler.

# Sıralanmış veriyi görüntüleme
df_2# Sıralama sonucunda elde edilen yeni veri çerçevesi.
```

```
# A tibble: 10 x 4
   name               height  mass species
   <chr>               <int> <dbl> <chr>
 1 R2-D2                  96    32 Droid
 2 R5-D4                  97    32 Droid
 3 Leia Organa           150    49 Human
 4 Beru Whitesun Lars    165    75 Human
 5 C-3PO                 167    75 Droid
 6 Luke Skywalker        172    77 Human
 7 Owen Lars             178   120 Human
 8 Obi-Wan Kenobi        182    77 Human
 9 Biggs Darklighter     183    84 Human
10 Darth Vader           202   136 Human
```

desc(): Bir sütunu azalan sırada sıralamak için kullanılır.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# starwars veri setinden alt küme oluşturma
df <- starwars[1:10, c(1, 2, 3, 11)]

# 'height' sütununa göre AZALAN sıralama
df_2 <- df %>%
  arrange(desc(height))
# desc(): height sütununu azalan sıraya göre sıralamak için kullanılır.

# Sıralanmış veri seti
df_2
```

```
# A tibble: 10 x 4
   name               height  mass species
   <chr>               <int> <dbl> <chr>
 1 Darth Vader           202   136 Human
 2 Biggs Darklighter     183    84 Human
 3 Obi-Wan Kenobi        182    77 Human
 4 Owen Lars             178   120 Human
 5 Luke Skywalker        172    77 Human
 6 C-3PO                 167    75 Droid
 7 Beru Whitesun Lars    165    75 Human
 8 Leia Organa           150    49 Human
 9 R5-D4                  97    32 Droid
```

```
10 R2-D2                  96    32 Droid
```

- **Belirli Bir Sütunun Belirli Bir Karakterine Göre Sıralama**

`substr(x, start, stop)`: Bir metinden belirli bir başlangıç ve bitiş pozisyonu arasındaki karakterleri döndürür.

`x`: İşlem yapılacak metin veya karakter vektörü (örneğin, bir sütun adı).

- `start`: Metnin hangi pozisyondan başlayacağını belirtir (dahil).
- `stop`: Metnin hangi pozisyonda duracağını belirtir (dahil).

Eğer `x` "Star Wars" ise:

- `substr(x, 1, 4)` → `"Star"` (İlk 4 karakter).
- `substr(x, 6, 9)` → `"Wars"` (6. ve 9. karakterler arası).

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# starwars veri setinden alt küme oluşturma
df <- starwars[1:10, c(1, 2, 3, 11)]

# 'name' sütununu adın ilk harfine göre sıralama
df_2 <- df %>%
  arrange(substr(name, 1, 2))
# substr(): 'name' sütununun ilk iki harfine göre sıralama yapar.

# Veriyi görüntüleme
df_2
```

```
# A tibble: 10 x 4
   name               height  mass species
   <chr>               <int> <dbl> <chr>
 1 Beru Whitesun Lars    165    75 Human
 2 Biggs Darklighter     183    84 Human
 3 C-3PO                 167    75 Droid
 4 Darth Vader           202   136 Human
 5 Leia Organa           150    49 Human
 6 Luke Skywalker        172    77 Human
 7 Obi-Wan Kenobi        182    77 Human
 8 Owen Lars             178   120 Human
 9 R2-D2                  96    32 Droid
10 R5-D4                  97    32 Droid
```

6.1.3.2 Birden Fazla Sütuna Göre Satırları Sıralama

Satırlar birden fazla sütuna göre de sıralanabilir. Bu durumda sıralama sırasıyla gerçekleşir: önce birinci sütun, ardından ikinci sütun ve devam eder. Aşağıdaki örnek, satırların `height` ve `mass` değişkenlerine göre sıralanmasını göstermektedir.

- **Birden Fazla Sütuna Göre Sıralama**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# starwars veri setinden alt küme oluşturma
df <- starwars[1:10, c(1, 2, 3, 11)]

# 'height' ve ardından 'mass' sütununa göre sıralama
df_2 <- df %>%
  arrange(height, mass)
# Önce height sütununa, ardından mass sütununa göre artan sıralama yapar.

# Veriyi görüntüleme
df_2
```

```
# A tibble: 10 x 4
   name               height  mass species
   <chr>               <int> <dbl> <chr>
 1 R2-D2                  96    32 Droid
 2 R5-D4                  97    32 Droid
 3 Leia Organa           150    49 Human
 4 Beru Whitesun Lars    165    75 Human
 5 C-3PO                 167    75 Droid
 6 Luke Skywalker        172    77 Human
 7 Owen Lars             178   120 Human
 8 Obi-Wan Kenobi        182    77 Human
 9 Biggs Darklighter     183    84 Human
10 Darth Vader           202   136 Human
```

- **Birden Fazla Sütuna Göre Artan Sıralama**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# starwars veri setinden alt küme oluşturma
```

```r
df <- starwars[1:10, c(1, 2, 3, 11)]

# 'mass' ve ardından 'height' sütununa göre sıralama
df_2 <- df %>%
  arrange(mass, height)
# Önce mass sütununa, ardından height sütununa göre artan sıralama yapar.

# Veriyi görüntüleme
df_2
```

```
# A tibble: 10 x 4
   name               height  mass species
   <chr>               <int> <dbl> <chr>
 1 R2-D2                  96    32 Droid
 2 R5-D4                  97    32 Droid
 3 Leia Organa           150    49 Human
 4 Beru Whitesun Lars    165    75 Human
 5 C-3PO                 167    75 Droid
 6 Luke Skywalker        172    77 Human
 7 Obi-Wan Kenobi        182    77 Human
 8 Biggs Darklighter     183    84 Human
 9 Owen Lars             178   120 Human
10 Darth Vader           202   136 Human
```

6.1.4 Sütun Adlarını Yeniden Adlandırma: `rename()`

R'de **`dplyr`** paketinin **`rename()`** fonksiyonu, bir veri çerçevesindeki sütun isimlerini değiştirmek için kullanılır. Bu fonksiyon, belirli sütunlara yeni isimler atamanıza olanak tanır.

Ayrıca, **`rename_with()`** fonksiyonu, sütunları bir fonksiyon kullanarak toplu halde yeniden adlandırmanıza olanak sağlar.

`dplyr` paketindeki **`band_instruments`** veri setini kullanacağız. Bu veri seti, **`name`** ve **`plays`** adlı iki sütunu içermektedir.

- İlk sütunu "First Name" olarak yeniden adlandırmak istediğinizi düşünüyorsanız, aşağıdaki komutu çalıştırabilirsiniz:

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# 'name' sütununu 'First Name' olarak yeniden adlandırma
```

```r
df_2 <- band_instruments %>%
  rename("First Name" = name)

# Veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
  `First Name` plays
  <chr>        <chr>
1 John         guitar
2 Paul         bass
3 Keith        guitar
```

- **Sütun Adını İndeks ile Yeniden Adlandırma**

Sütunları indeks numarasına göre de yeniden adlandırabilirsiniz. Aşağıdaki örnek, veri setinin ikinci sütununun nasıl yeniden adlandırılacağını göstermektedir.

```r
library(tidyverse)

# İkinci sütunu 'Second column' olarak yeniden adlandırma
df_2 <- band_instruments %>%
  rename("Second column" = 2)

# Veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
  name  `Second column`
  <chr> <chr>
1 John  guitar
2 Paul  bass
3 Keith guitar
```

- **Birden Fazla Sütun Adını Yeniden Adlandırma**
- Birden fazla sütunu aynı anda yeniden adlandırmak mümkündür. Bunun için, fonksiyona **`new_name = old_name`** ifadeleri eklenir ve bu ifadeler virgülle ayrılır. Aşağıdaki örnek, **`name`** sütununu **`Member`**, **`plays`** sütununu ise **`Instrument`** olarak yeniden adlandırmaktadır.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# 'name' sütununu 'Member', 'plays' sütununu 'Instrument' olarak adlandırma
df_2 <- band_instruments %>%
  rename("Member" = name,
         "Instrument" = plays)

# Veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
  Member Instrument
  <chr>  <chr>
1 John   guitar
2 Paul   bass
3 Keith  guitar
```

6.2 Veri Dönüştürme

6.2.1 Yeni Değişkenler Oluşturma ve Düzenleme: `mutate()`

`mutate()`, R'de **`dplyr`** paketinde kullanılan bir fonksiyondur ve bir veri çerçevesinde yeni sütunlar oluşturmak veya mevcut sütunları değiştirmek için kullanılır. Veri çerçevesinin orijinal yapısını korur ve sonuçları yeni sütunlar olarak saklar.

6.2.1.1 `mutate()` Fonksiyonu Sözdizimi

- `.data` *Veri çerçevesi*
- `...` *Yeni sütunlar (örneğin, yeni_sütun = işlem)*
- `.by = NULL`, *Gruplama değişkenleri (isteğe bağlı)*
- `.keep = c("all", "used", "unused", "none")`, *Hangi sütunların tutulacağı*
- `.before = NULL`, *Yeni sütunları belirli bir sütundan önce yerleştirme*
- `.after = NULL` *Yeni sütunları belirli bir sütundan sonra yerleştirme*

6.2.1.2 Yeni Sütun Oluşturma

Bir veri çerçevesine yeni bir sütun eklemek için, yeni sütunun adını (örneğin, `Var3`) ve yeni sütunun değerlerini hesaplamak için bir ifadeyi belirtmeniz yeterlidir. Aşağıdaki örnekte, yeni sütunun değeri, diğer iki sütunun toplamı (`Var1 + Var2`) olarak hesaplanmıştır.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütun: 'Var3', 'Var1' ve 'Var2'nin toplamı
df_2 <- df %>%
  mutate(Var3 = Var1 + Var2)
# Yeni sütun ekler: Var3, Var1 ve Var2 sütunlarının toplamı olarak hesaplanır.

# Yeni sütun eklenmiş veri seti
df_2
```

```
  Var1 Var2 Var3
1   32   39   71
2   34    1   35
3   15   29   44
4   12    3   15
5   42   35   77
```

- **Yeni Sütun Olarak Karekök Hesaplama**

Bir veri çerçevesine yeni bir sütun eklemek için mevcut bir sütuna bir fonksiyon uygulayabilirsiniz. Aşağıdaki örnek, bir sütunun karekökünü hesaplayarak yeni bir sütun (`Sqrt_Var1`) oluşturmayı göstermektedir.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütun: 'Sqrt_Var1', 'Var1' sütununun karekökü
```

```r
df_2 <- df %>%
  mutate(Sqrt_Var1 = sqrt(Var1))
# Yeni sütun ekler: Sqrt_Var1, Var1 sütununun karekökü olarak hesaplanır.

# Veriyi görüntüleme
df_2 # Yeni sütun eklenmiş veri seti.
```

```
  Var1 Var2 Sqrt_Var1
1   32   39  5.656854
2   34    1  5.830952
3   15   29  3.872983
4   12    3  3.464102
5   42   35  6.480741
```

- **Birden Fazla Yeni Sütun Eklemek ve Koşullu Değer Atamak**

mutate fonksiyonuna birden fazla ifade ekleyerek aynı anda birden fazla sütun oluşturabilirsiniz. Bunun için ifadeleri virgülle ayırmanız yeterlidir.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütunlar: 'Var3', 'Var4' ve 'Var5'
df_2 <- df %>%
  mutate(
    Var3 = Var1 + Var2,             # Var1 ve Var2'nin toplamı
    Var4 = cumsum(Var1),            # Var1'in kümülatif toplamı
    Var5 = if_else(Var1 > Var2, TRUE, FALSE)
    # Var1, Var2'den büyükse TRUE, değilse FALSE
  )

# Veriyi görüntüleme
df_2 # Birden fazla sütun eklenmiş veri seti.
```

```
  Var1 Var2 Var3 Var4  Var5
1   32   39   71   32 FALSE
2   34    1   35   66  TRUE
3   15   29   44   81 FALSE
```

```
4   12    3   15   93  TRUE
5   42   35   77  135  TRUE
```

i if_else fonksyonu

`if_else()` fonksiyonu R programlama dilinde koşullu ifadeler oluşturmak için kullanılan bir fonksiyondur. Temelde, bir koşulun doğru olup olmamasına göre farklı değerler döndürür. Bu, daha geleneksel `if` ve `else` yapılarının vektörleştirilmiş bir karşılığıdır ve özellikle veri manipülasyonu ve vektörler üzerinde işlem yaparken çok daha verimli olabilir.

```
if_else(condition, true_value, false_value)
```

- `condition`: Mantıksal bir vektör veya ifade. Her bir eleman için `TRUE` veya `FALSE` değerini döndürmelidir.
- `true_value`: `condition` vektöründeki ilgili eleman `TRUE` ise döndürülecek değer. Bu bir vektör olabilir.
- `false_value`: `condition` vektöründeki ilgili eleman `FALSE` ise döndürülecek değer. Bu da bir vektör olabilir.

Nasıl Çalışır?
`if_else()` fonksiyonu, `condition` vektöründeki her bir elemanı teker teker kontrol eder. Eğer ilgili eleman `TRUE` ise, `true_value` vektöründeki aynı konumdaki değeri döndürür. Eğer ilgili eleman `FALSE` ise, `false_value` vektöründeki aynı konumdaki değeri döndürür.

- **Belirli Sütunlarda İşlem Yapmak ve Yeni Sütunlar Oluşturmak: `across()` Kullanımı**

`across()` fonksiyonu, **`mutate`** ile birlikte kullanılarak belirli sütunlara fonksiyonlar uygulamayı sağlar. Aynı zamanda yardımcı fonksiyonlar (**`contains`**, **`starts_with`**, vb.) ile sütun seçiminde esneklik sunar.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
df <- tibble(Var1 = c(1, 4, 9), Var2 = c(16, 25, 36), Var3 = c(49, 64, 81))

# "1" içeren sütunlara karekök uygulama ve yeni sütunlar oluşturma
df_2 <- df %>%
   mutate(
      across(
         .cols = contains("Var"), # İsimlerinde "Var" geçen sütunları seçer
         .fns = sqrt,           # Karekök fonksiyonunu uygular
```

```
          .names = "{.col}_sqrt")) # Yeni sütun isimleri: Eski isim + "_sqrt"

# Veriyi görüntüleme
df_2 # Yeni sütunlar eklenmiş veri seti.
```

```
# A tibble: 3 x 6
   Var1  Var2  Var3 Var1_sqrt Var2_sqrt Var3_sqrt
  <dbl> <dbl> <dbl>     <dbl>     <dbl>     <dbl>
1     1    16    49         1         4         7
2     4    25    64         2         5         8
3     9    36    81         3         6         9
```

6.2.1.3 Mevcut Sütunları Güncelleme

- **Mevcut Bir Sütunun Değerlerini Güncelleme**

`mutate()` fonksiyonu, mevcut sütunları güncellemek veya üzerinde işlem yapmak için de kullanılabilir. Bunu gerçekleştirmek için, `eski_sütun_adı = ifade` sözdizimini kullanmanız yeterlidir.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
df <- tibble(Var1 = c(10, 20, 30), Var2 = c(5, 10, 15))

# 'Var1' sütununun değerlerini ikiye katlama
df_2 <- df %>%
  mutate(Var1 = Var1 * 2) # Var1 sütununun yeni değerleri Var1 * 2

# Veriyi görüntüleme
df_2
```

```
# A tibble: 3 x 2
   Var1  Var2
  <dbl> <dbl>
1    20     5
2    40    10
3    60    15
```

- **Birden Fazla Yeni Sütun Eklemek ve Koşullu Değer Atamak**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(123)
df <- tibble(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütunlar: 'Var3', 'Var4' ve 'Var5'
df_2 <- df %>%
  mutate(
    Var1 = Var1 + Var2,             # Var1 ve Var2'nin toplamı
    Var2 = cumsum(Var1),            # Var1'in kümülatif toplamı
    Var3 = if_else(Var1 >= Var2, "Yes", "No")
    # Var1, Var2'den büyükse "Yes", değilse "No"
  )

# Veriyi görüntüleme
df_2 # Birden fazla sütun eklenmiş veri seti.
```

```
# A tibble: 5 x 3
   Var1  Var2 Var3
  <int> <int> <chr>
1    81    81 Yes
2    58   139 No
3    51   190 No
4    17   207 No
5    67   274 No
```

- **Belirli Sütunlarda İşlem Yapmak**

`across()` fonksiyonunu kullanarak belirli sütunları seçebilir ve bunlara özel bir fonksiyon uygulayabilirsiniz. Yeni sütun oluşturmadan, mevcut sütunların değerlerini doğrudan değiştirebilirsiniz.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- tibble(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# 'Var1' hariç tüm sütunlara logaritma işlemi uygulama
df_2 <- df %>%
```

```
  mutate(across(!Var1, log)) # Var1 dışındaki tüm sütunlara log uygulanır

# Veriyi görüntüleme
df_2 # Güncellenmiş veri seti.
```

```
# A tibble: 5 x 2
   Var1  Var2
  <int> <dbl>
1    32  3.66
2    34  0
3    15  3.37
4    12  1.10
5    42  3.56
```

6.2.1.4 Yeni Sütunların Konumu

Varsayılan olarak, **mutate** fonksiyonu yeni sütunları veri çerçevesinin sonuna ekler. Ancak, **.before** veya **.after** argümanlarını kullanarak yeni sütunun başka bir sütuna göre konumunu belirleyebilirsiniz.

- **Yeni Sütunlar Eklemek ve Belirli Pozisyonlara Yerleştirmek**

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütunlar ekleniyor
df_2 <- df %>%
  mutate(Var3 = Var1 / Var2, .before = Var2) %>%
  # Yeni sütun: Var3, Var2'den önce ekleniyor
  mutate(Var4 = Var1 * Var2, .before = Var1) %>%
  # Yeni sütun: Var4, ilk sütundan önce ekleniyor
  mutate(Var5 = (Var1 * Var2) / 2, .after = Var4)
  # Yeni sütun: Var5, Var4'ten sonra ekleniyor

# Veriyi görüntüleme
df_2
```

```
  Var4  Var5 Var1       Var3 Var2
1 1248 624.0   32  0.8205128   39
2   34  17.0   34 34.0000000    1
3  435 217.5   15  0.5172414   29
4   36  18.0   12  4.0000000    3
5 1470 735.0   42  1.2000000   35
```

6.2.1.5 Sütunları saklama veya çıkarma

Yeni sütunlar bir veri çerçevesine eklendiğinde, varsayılan olarak diğer tüm sütunlar korunur. Ancak `.keep` argümanı sayesinde bu davranış değiştirilebilir. `.keep` varsayılan olarak `"all"` değerine sahiptir, ancak aşağıdaki şekilde ayarlanabilir:

- **`"all"`**: Tüm sütunları korur (varsayılan).
- **`"used"`**: Sadece **`mutate`** içinde kullanılan sütunları korur.
- **`"unused"`**: **`mutate`** içinde kullanılmayan sütunları korur.
- **`"none"`**: Eski tüm sütunları siler, sadece yeni sütunlar kalır.
- **Kullanılan Sütunları Saklamak `.keep = "used"`**

Aşağıdaki örnek, sadece kullanılan sütunları saklamak için **`.keep = "used"`** ayarının nasıl kullanılacağını göstermektedir:

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütun ekleme ve sadece kullanılan sütunu ('Var1') ve
# yeni sütunu ('Var3') koruma
df_2 <- df %>%
  mutate(Var3 = Var1 * 2,
         .keep = "used")

df_2
```

```
  Var1 Var3
1   32   64
2   34   68
```

```
3   15   30
4   12   24
5   42   84
```

- **Yalnızca Kullanılmayan Sütunları Saklamak `.keep = "unused"`**

Tam tersi, yalnızca yeni sütunu ve kullanılmayan sütunları korumaktır. Bunun için **`.keep = "unused"`** ayarı kullanılabilir. Bu durumda, **`mutate`** içinde kullanılan sütunlar hariç tutulur.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütun ekleme ve sadece yeni sütunu ('Var3') ve kullanılmayan
# sütunu ('Var2') koruma
df_2 <- df %>%
  mutate(Var3 = Var1 * 2, .keep = "unused")

df_2
```

```
  Var2 Var3
1   39   64
2    1   68
3   29   30
4    3   24
5   35   84
```

- **Yalnızca Yeni Sütunu Saklamak `.keep = "none"`**

Son olarak, orijinal veri çerçevesindeki tüm sütunları kaldırmak için **`.keep = "none"`** kullanabilirsiniz.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(8)
df <- data.frame(Var1 = sample(1:50, 5), Var2 = sample(1:50, 5))

# Yeni sütun ekleme ve yalnızca yeni sütunu ('Var3') saklama
```

```r
df_2 <- df %>%
  mutate(Var3 = Var1 * 2, .keep = "none")

df_2
```

```
  Var3
1   64
2   68
3   30
4   24
5   84
```

6.2.2 İstatistiksel Özetler Oluşturma: `summarise()`

`summarise` (veya **`summarize`**) fonksiyonu, verileri toplulaştırmak ve özetlemek için kullanılır. Bu fonksiyon, özellikle verileri gruplara ayırarak her grup için istatistiksel özetler veya hesaplamalar yapmak açısından oldukça faydalıdır. Belirtilen özet istatistiklerle birlikte yeni bir veri çerçevesi oluşturur ve her grup için tek bir satır döndürür.

6.2.2.1 Sözdizimi

```r
# summarise(data, new_column = function(column))
```

Argümanlar:

- **`data`**: Özetleme yapmak istediğiniz veri çerçevesi veya gruplandırılmış veri.
- **`new_column`**: Yeni oluşturulacak sütunun adı.
- **`function(column)`**: Belirli bir sütuna uygulanacak fonksiyon (örneğin: **`sum`**, **`mean`**, **`max`**).

6.2.2.2 Verinin İstatistiksel Özetleri

Bir veri setinden belirli değişkenlerin istatistiksel özetlerini içeren yeni bir veri çerçevesi oluşturabilirsiniz. **`summarise`** fonksiyonu ile kullanılabilecek en faydalı fonksiyonlar aşağıda açıklanmıştır:

6.2.3 Verinin İstatistiksel Özetleri

Bir veri setinden belirli değişkenlerin istatistiksel özetlerini içeren yeni bir veri çerçevesi oluşturabilirsiniz. `summarise` fonksiyonu ile kullanılabilecek en faydalı fonksiyonlar aşağıda açıklanmıştır:

Kullanışlı Fonksiyonlar

Fonksiyon	Açıklama
`mean()`	Değerlerin ortalaması
`median()`	Değerlerin medyanı
`sd(), var()`	Değerlerin standart sapması ve varyansı
`quantile()`	Değerlerin çeyrek dilimleri
`IQR()`	Değerlerin interçeyrek aralığı
`min(), max()`	Minimum ve maksimum değer
`first()`	İlk değer
`last()`	Son değer
`nth()`	N. sıradaki değer
`n()`	Her gruptaki eleman sayısı
`n_distinct()`	Benzersiz değerlerin sayısı

- **Veri Seti Üzerinde Özetleme İşlemi**

Aşağıdaki örnekte, orijinal veri setindeki sayısal sütunların **ortalamalarını** hesaplayarak yeni bir veri çerçevesi oluşturmayı gösteriyoruz.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(9)
df <- data.frame(group = sample(c("G1", "G2"), 5, replace = TRUE),
                 x = sample(1:50, 5), y = sample(1:50, 5))

# Yeni sütunlar: 'mean_x' ve 'mean_y'
df_2 <- df %>%
   summarise(mean_x = mean(x), # x sütununun ortalaması
             mean_y = mean(y))  # y sütununun ortalaması

# Veriyi görüntüleme
df_2 # x ve y sütunlarının ortalaması alınmış veri seti.
```

```
  mean_x mean_y
1   21.6   38.2
```

Sonuçta elde edilen çıktı, giriş fonksiyonu tarafından döndürülen değerler kadar satır içerecektir.

6.2.3.1 Gruplara Göre Veriyi Özetleme (`group_by`)

`summarise` fonksiyonu, **`group_by`** ile birlikte kullanıldığında, her grup için istatistiksel özetler oluşturmak açısından oldukça etkili bir yöntem sunar. Bu kombinasyon, veri setini belirli bir değişkene göre gruplandırarak, her grup için özelleştirilmiş özet istatistiklerin elde edilmesini sağlar.

Aşağıdaki örnek, **group** değişkenine göre gruplandırılmış veri seti üzerinde, her sütunun ortalama değerini hesaplayarak özet bir veri çerçevesi oluşturmaktadır.

```
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(9)
df <- data.frame(group = sample(c("G1", "G2"), 5, replace = TRUE),
                 x = sample(1:50, 5),
                 y = sample(1:50, 5))

# Gruplara göre 'x' ve 'y' sütunlarının toplamları
df_2 <- df %>%
   group_by(group) %>%
   summarise(mean_x = sum(x), # x sütununun toplamı
             mean_y = sum(y))  # y sütununun toplamı

# Veriyi görüntüleme
df_2 # Gruplara göre özetlenmiş veri seti.
```

```
# A tibble: 2 x 3
  group mean_x mean_y
  <chr>  <int>  <int>
1 G1        52     97
2 G2        56     94
```

- **Çoklu Gruplara Göre Veri Setini Özetleme**

Ayrıca, birden fazla kategorik değişkene göre gruplama yapabilirsiniz. Bu durumda, fonksiyon her grup ve alt grup için istatistiksel özetler hesaplar. Varsayılan olarak, çıktı, birinci kategorik değişkene göre gruplandırılır ve bu durum bir mesaj ile belirtilir.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(9)
df <- data.frame(group = sample(c("G1", "G2"), 5, replace = TRUE),
                 group_2 = sample(c("G3", "G4"), 5, replace = TRUE),
                 x = sample(1:50, 5),
                 y = sample(1:50, 5))

# Gruplara göre 'x' ve 'y' sütunlarının toplamları
df_2 <- df %>%
   group_by(group, group_2) %>% # 'group' ve 'group_2' sütunlarına göre gruplama
   summarise(sum_x = sum(x),    # x sütununun toplamını hesaplar
             sum_y = sum(y))    # y sütununun toplamını hesaplar

# Veriyi görüntüleme
df_2 # Gruplara göre özetlenmiş veri seti.
```

```
# A tibble: 4 x 4
# Groups:   group [2]
  group group_2 sum_x sum_y
  <chr> <chr>   <int> <int>
1 G1    G3         74    86
2 G1    G4         18    30
3 G2    G3         48    22
4 G2    G4         37    35
```

- **`.groups` argümanı**

`.groups` argümanı isteğe bağlıdır ve gruplama işlemi sonrası grupların nasıl ele alınacağını kontrol etmek için kullanılır. Aşağıdaki değerlerden birini alabilir:

1. `"drop_last"`: Eğer birden fazla gruplama seviyesi varsa, son gruplama seviyesi kaldırılır, ancak diğerleri korunur. Örneğin, iki kategorik değişkenle gruplama yaptıysanız, birinci değişkene göre gruplama devam eder.
2. `"drop"`: Tüm gruplama seviyelerini kaldırır. Sonuç, gruplandırılmamış, düz bir veri çerçevesi olur.
3. `"keep"`: Orijinal gruplama yapısını korur. Özetleme işlemi tamamlandıktan sonra veri, hala gruplandırılmış halde kalır.
4. `"rowwise"`: Her satırı kendi başına bir grup olarak ele alır. Bu, satır bazında işlem yapmak için kullanılır ve genellikle özelleştirilmiş hesaplamalarda faydalıdır.

- **Gruplama Sonrası Gruplama Seviyelerini Kaldırma `.groups`**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(9)
df <- data.frame(group = sample(c("G1", "G2"), 5, replace = TRUE),
                 group_2 = sample(c("G3", "G4"), 5, replace = TRUE),
                 x = sample(1:50, 5),
                 y = sample(1:50, 5))

# Gruplara göre 'x' ve 'y' sütunlarının toplamları, tüm gruplama seviyeleri kaldırılır
df_2 <- df %>%
   group_by(group, group_2) %>% # 'group' ve 'group_2' sütunlarına göre gruplama
   summarise(sum_x = sum(x),    # x sütununun toplamı
             sum_y = sum(y),    # y sütununun toplamı
             .groups = "drop")  # Tüm gruplama seviyelerini kaldırır

# Veriyi görüntüleme
df_2 # Gruplama bilgisi olmadan özetlenmiş veri seti.
```

```
# A tibble: 4 x 4
  group group_2 sum_x sum_y
  <chr> <chr>   <int> <int>
1 G1    G3         74    86
2 G1    G4         18    30
3 G2    G3         48    22
4 G2    G4         37    35
```

- **Aynı işlem `ungroup()` fonksyonu ile de yapılabilir:**

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(9)
df <- data.frame(group = sample(c("G1", "G2"), 5, replace = TRUE),
                 group_2 = sample(c("G3", "G4"), 5, replace = TRUE),
                 x = sample(1:50, 5),
                 y = sample(1:50, 5))
```

```r
# Gruplara göre 'x' ve 'y' sütunlarının toplamları, ardından gruplama kaldırılır
df_2 <- df %>%
   group_by(group, group_2) %>% # 'group' ve 'group_2' sütunlarına göre gruplama
   summarise(sum_x = sum(x),     # x sütununun toplamını hesaplar
             sum_y = sum(y)) %>% # y sütununun toplamını hesaplar
   ungroup()                     # Gruplama kaldırılır

# Veriyi görüntüleme
df_2 # Gruplama bilgisi kaldırılmış özetlenmiş veri seti.
```

```
# A tibble: 4 x 4
  group group_2 sum_x sum_y
  <chr> <chr>   <int> <int>
1 G1    G3         74    86
2 G1    G4         18    30
3 G2    G3         48    22
4 G2    G4         37    35
```

6.2.3.2 Birden Fazla Sütunu Özetleme

Birden fazla sütunu manuel olarak belirtmek yerine, **`summarise`** ve **`across`** kombinasyonunu kullanarak bir koşula göre sütunları seçerek özetler oluşturabilirsiniz. Sütunları seçmek için kullanılan yardımcı fonksiyonların listesine göz atabilirsiniz.

Aşağıdaki örnekte, **group** sütunu hariç tüm sütunların varyansı hesaplanmakta ve sonuç sütunları, orijinal sütun adlarına "var" eklenerek yeniden adlandırılmaktadır.

```r
# Gerekli paketin yüklenmesi
library(tidyverse)

# Örnek veri
set.seed(9)
df <- data.frame(group = sample(c("G1", "G2"), 5, replace = TRUE),
                 x = sample(1:50, 5),
                 y = sample(1:50, 5))

# Varyans hesaplama: 'group' hariç tüm sütunların varyansı
df_2 <- df %>%
  summarise(
    across(!group,  # 'group' sütunu hariç diğer sütunlar
           var,     # Varyans hesaplamak için 'var' fonksiyonu
```

```
            .names = "{.col}_var")) # Sütun adlarına "_var" ekler

# Veriyi görüntüleme
df_2 # 'group' dışındaki sütunların varyansını içeren veri seti.
```

```
  x_var y_var
1 254.3 149.2
```

6.3 Veri Şekillendirme

6.3.1 Veri Çerçevelerini Birleştirme: `merge()`

R'deki **`merge`** fonksiyonu, iki veri çerçevesini ortak sütunlar veya satır isimleri aracılığıyla birleştirmeye olanak tanır. Bu fonksiyon, sol birleşim (`left join`), iç birleşim (`inner join`), sağ birleşim (`right join`) veya tam birleşim (`full join`) gibi farklı veri tabanı (**SQL**) birleşim türlerini gerçekleştirebilir. Bu eğitimde, R'nin temel fonksiyonlarını kullanarak veri setlerini birleştirmenin çeşitli yöntemlerini örneklerle öğrenebilirsiniz.

6.3.1.1 R'deki `merge()` Fonksiyonu

```
library(tidyverse)

# merge(x, y, ...)
```

Argümanlar

- **`x, y`**: Birleştirilecek veri çerçeveleri veya dönüştürülebilecek diğer objeler.
- **`by`**: Birleştirme işlemi için kullanılacak ortak sütunların adları. Varsayılan olarak, her iki veri çerçevesinde bulunan sütun adlarının kesişimi kullanılır.
- **`by.x, by.y`**: Sırasıyla **`x`** ve **`y`** veri çerçevelerinden birleştirme için kullanılacak sütunların adları. Özel sütunlar belirtmek için kullanılır.
- **`all`**: **`TRUE`** olarak ayarlandığında hem **`all.x`** hem de **`all.y`** **`TRUE`** olur ve tam birleşim (full join) gerçekleştirilir.
- **`all.x, all.y`**: **`all.x = TRUE`**: **`x`**'deki tüm satırları korur, **`y`**'de eşleşmeyenler için eksik değerler (NA) eklenir (sol birleşim - left join). **`all.y = TRUE`**: **`y`**'deki tüm satırları korur, **`x`**'de eşleşmeyenler için eksik değerler (NA) eklenir (sağ birleşim - right join).
- **`sort`**: **`TRUE`** ise çıktı, birleştirme için kullanılan sütunlara göre sıralanır. Varsayılan değer **`TRUE`**'dur.

- `suffixes`: Ortak sütun isimlerini ayırt etmek için kullanılan ekler. Varsayılan olarak `c(".x", ".y")` olarak ayarlanmıştır.
- `no.dups`: `TRUE` ise, tekrarlanan sütun adlarını önlemek için daha fazla ekler ekler.
- `incomparables`: Eşleştirilemeyen değerlerle nasıl başa çıkılacağını belirtir. Varsayılan olarak `NULL`'dır.

Örnek Veri Çerçevesi Oluşturma - 1

```
# Örnek veri çerçevesi oluşturma
data1 <- data.frame(ID = 1:2,            # ID sütunu, 1 ve 2 değerleri
                    X1 = c("a1", "a2"),  # X1 sütunu, "a1" ve "a2" değerleri
                    stringsAsFactors = FALSE) # 'stringsAsFactors' parametresi,
# karakter sütunlarının faktör yerine karakter olarak saklanmasını sağlar.

# Veri çerçevesini görüntüleme
data1
```

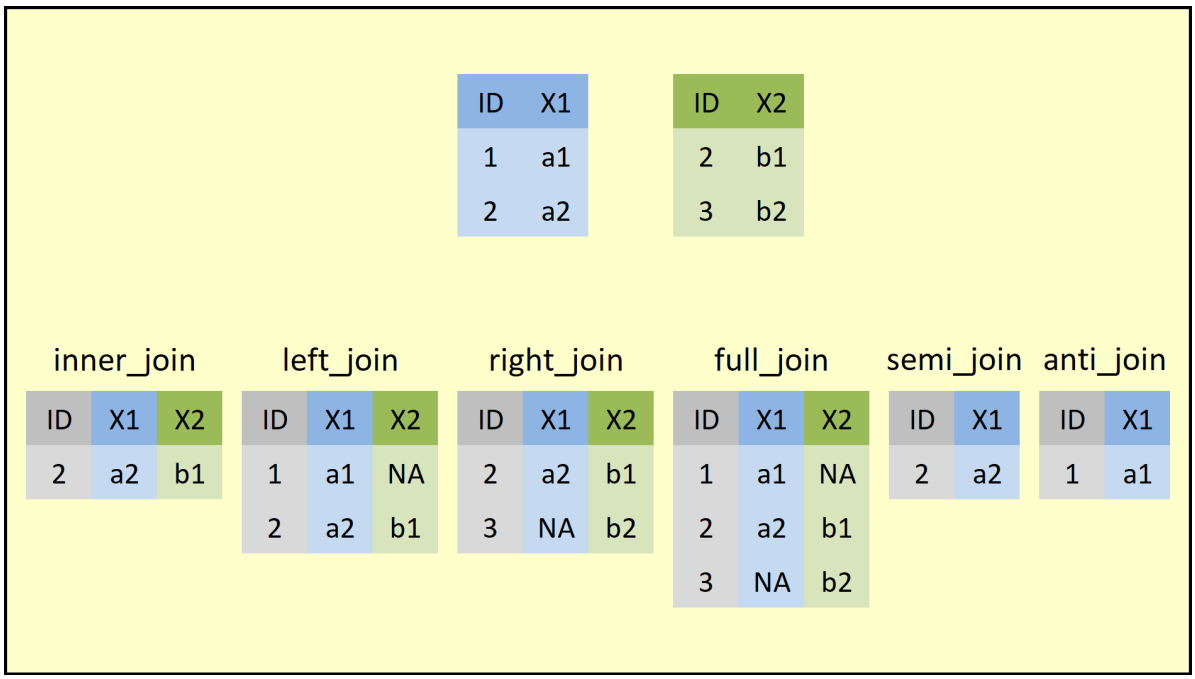
```
  ID X1
1  1 a1
2  2 a2
```

Örnek Veri Çerçevesi Oluşturma - 2

```
# İkinci veri çerçevesi oluşturma
data2 <- data.frame(ID = 2:3,            # ID sütunu, 2 ve 3 değerleri
                    X2 = c("b1", "b2"),  # X2 sütunu, "b1" ve "b2" değerleri
                    stringsAsFactors = FALSE) # 'stringsAsFactors' parametresi,
# karakter sütunlarının faktör yerine karakter olarak saklanmasını sağlar.

# Veri çerçevesini görüntüleme
data2
```

```
  ID X2
1  2 b1
2  3 b2
```

...



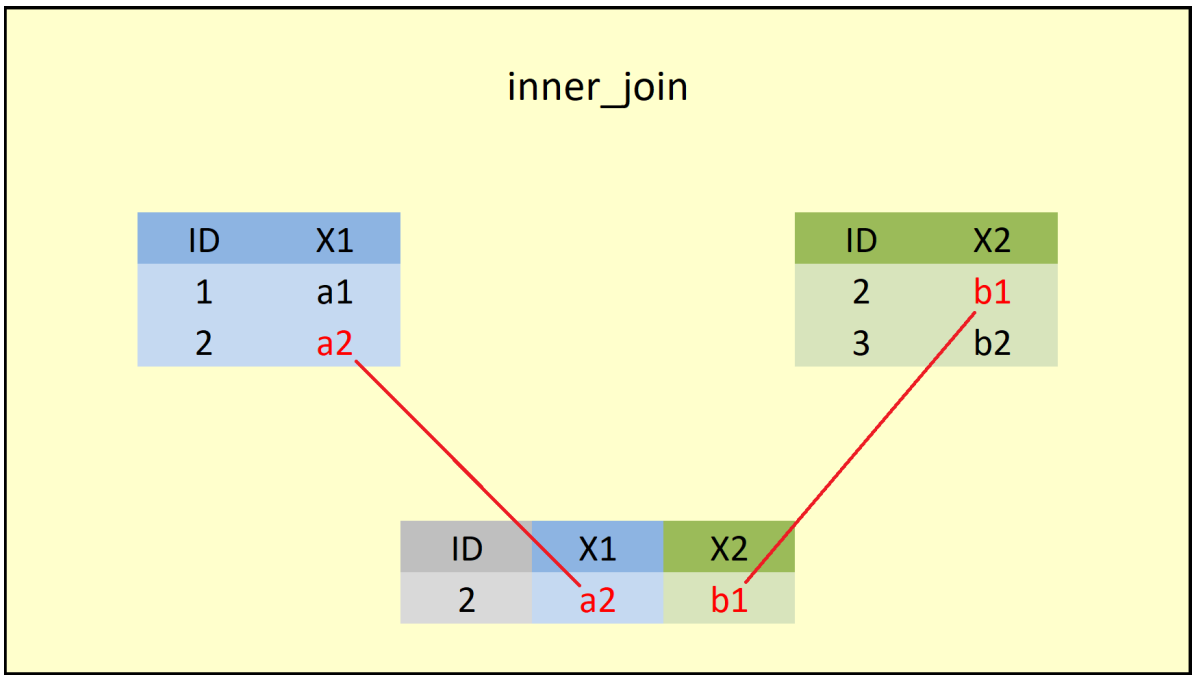
6.3.1.2 Inner Join - İç Birleştirme İşlemi

Inner join işlemi ile veri çerçevelerini birleştirmek için, birleştirilecek veri çerçevelerinin adlarını (**data1** ve **data2**) ve birleştirme işleminin gerçekleştirileceği ortak sütunun adını (**ID sütunu**) belirtmek yeterlidir. Bu işlem, iki veri çerçevesinin kesişim kümesini oluşturarak her iki çerçevede de bulunan ortak öğeleri içeren bir yapı sağlar.

```
# Gerekli paketin yüklenmesi
library(dplyr)

# İç birleştirme işlemi: 'inner_join' kullanımı
inner_join(data1, data2, by = "ID") # 'ID' sütununa göre birleştirme
```

```
  ID X1 X2
1  2 a2 b1
```

- `inner_join`, `dplyr` paketinin bir fonksiyonudur ve iki veri çerçevesini iç birleştirme (inner join) mantığıyla birleştirir.
- **İç birleştirme (inner join)**, yalnızca her iki veri çerçevesinde de ortak olan satırları birleştirir. Ortak bir sütun üzerinden eşleşmeyen satırlar sonuçta yer almaz.
- `by = "ID"`: Birleştirme işleminin **ID** sütunu üzerinden yapılacağını belirtir.
- Sonuç olarak, sadece **ID** sütununda ortak olan satırlar birleştirilir ve diğerleri dışlanır.



Inner join, her iki veri çerçevesinde ortak olan satırları birleştirir. Bu işlem sonucunda, her iki veri çerçevesinde de bulunan `ID` **2** tutulur. Ortaya çıkan sonuç veri çerçevesi, `ID` **2**'ye ait ortak sütunları içerir; bunlar, ilk veri çerçevesinden `X1` sütunundaki **a2** değeri ve ikinci veri çerçevesinden `X2` sütunundaki **b1** değeridir. Sonuç olarak, inner join işlemi yalnızca iki veri çerçevesinin ortak öğelerini birleştirir ve diğer öğeleri dışarıda bırakır. Bu yöntem, veri çerçevelerini karşılaştırarak kesişim kümesini oluşturmaya olanak tanır.

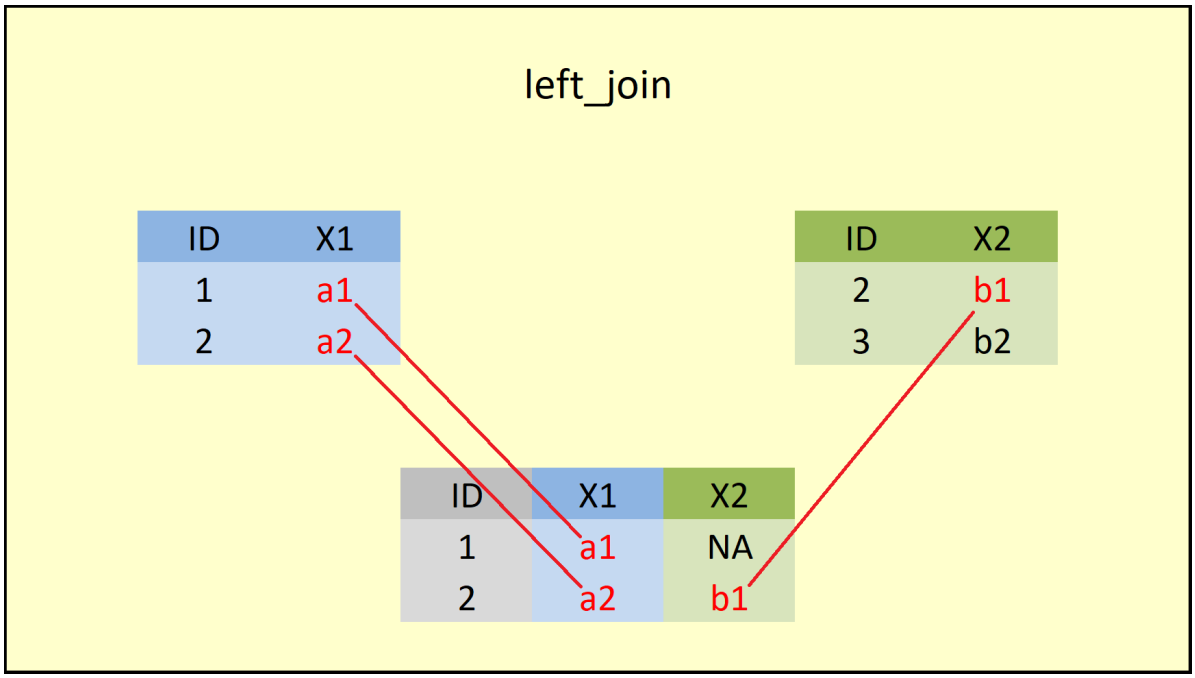
6.3.1.3 Left Join - Sol Birleştirme İşlemi

```
# Gerekli paketin yüklenmesi
library(dplyr)

# Sol birleştirme işlemi: 'left_join' kullanımı
left_join(data1, data2, by = "ID") # 'ID' sütununa göre birleştirme
```

```
  ID X1   X2
1  1 a1 <NA>
2  2 a2   b1
```

- `left_join` fonksiyonu, `dplyr` paketinin bir fonksiyonudur ve iki veri çerçevesini sol birleştirme mantığıyla birleştirir.

- **Sol birleştirme (left join)**, birinci veri çerçevesindeki tüm satırları tutar ve ikinci veri çerçevesinden sadece eşleşen değerleri ekler. Eğer ikinci veri çerçevesinde bir eşleşme yoksa, eksik değerler **NA** olarak atanır.
- `by = "ID"`: Birleştirme işleminin **ID** sütunu üzerinden yapılacağını belirtir.



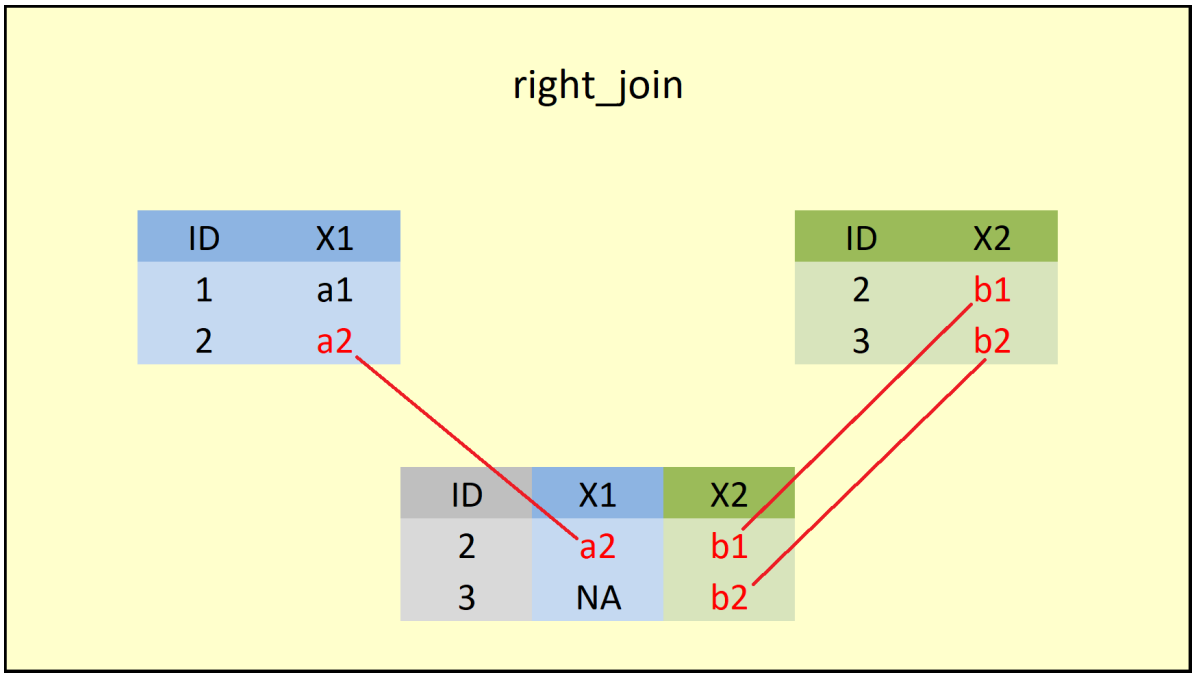
Left join, sol veri çerçevesindeki tüm satırları tutar ve sağ veri çerçevesinden yalnızca eşleşen değerleri ekler. Eşleşme yoksa eksik değerler (NA) atanır. **Inner join** ise, yalnızca her iki veri çerçevesinde **ortak olan** satırları tutar ve diğer satırları dışlar.

6.3.1.4 Right Join - Sağ Birleştirme İşlemi

```
# Gerekli paketin yüklenmesi
library(dplyr)

# Sağ birleştirme işlemi: 'right_join' kullanımı
right_join(data1, data2, by = "ID") # 'ID' sütununa göre sağ birleştirme
```

```
  ID   X1 X2
1  2   a2 b1
2  3 <NA> b2
```

- `right_join`, `dplyr` paketinin bir fonksiyonudur ve sağ birleştirme (right join) işlemini gerçekleştirir.
- **Sağ Birleştirme (Right Join)**, sağ veri çerçevesindeki tüm satırları tutar ve sol veri çerçevesinden yalnızca eşleşen değerleri ekler. Eğer sol veri çerçevesinde eşleşen bir satır yoksa, bu satır için eksik değerler (**NA**) atanır.
- `by = "ID"`: Birleştirmenin **ID** sütunu üzerinden yapılacağını belirtir.



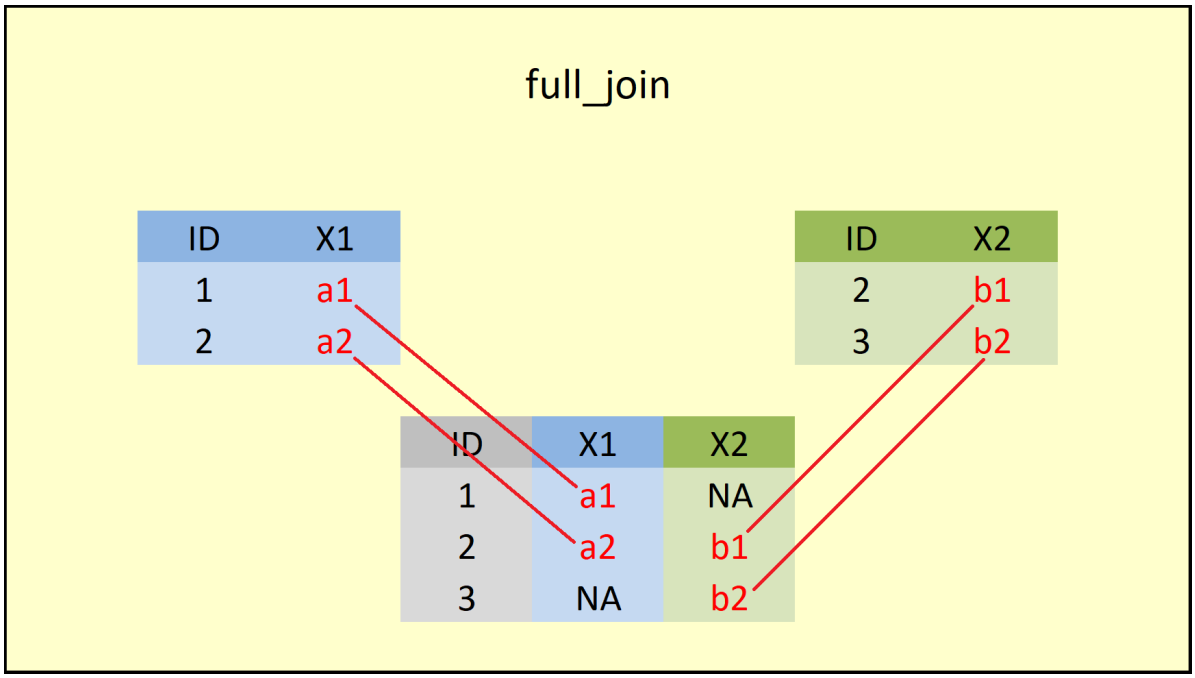
Right join, sağ veri çerçevesindeki tüm satırları korurken sol veri çerçevesinden yalnızca eşleşen değerleri ekler ve eşleşme olmayan satırlar için eksik değerler (**NA**) atar. **Left join** ile farkı, **Right join**'in sağ veri çerçevesini referans alması, **Left join**'in ise sol veri çerçevesini referans almasıdır. **Inner join** ile farkı ise, **Right join** sağ veri çerçevesindeki tüm satırları tutarken, **Inner join** yalnızca her iki veri çerçevesinde ortak olan satırları döndürmesidir; bu nedenle **Inner join** eksik değer içermezken, **Right join** eksik değerler barındırabilir.

6.3.1.5 Full Join - Tam Birleştirme İşlemi

```
# Gerekli paketin yüklenmesi
library(dplyr)

# Tam birleştirme işlemi: 'full_join' kullanımı
full_join(data1, data2, by = "ID") # 'ID' sütununa göre tam birleştirme
```

```
  ID   X1   X2
1  1   a1 <NA>
2  2   a2   b1
3  3 <NA>   b2
```

- `full_join`, `dplyr` paketinin bir fonksiyonudur ve tam birleştirme (full join) işlemini gerçekleştirir.
- **Tüm Birleştirme (full_join)**, iki veri çerçevesindeki tüm satırları tutar. Her iki veri çerçevesinde de eşleşen satırlar birleştirilir; eşleşmeyen satırlar ise eksik değerler (**NA**) ile tamamlanır.
- `by = "ID"`: Birleştirme işleminin **ID** sütunu üzerinden yapılacağını belirtir.



Full join, her iki veri çerçevesindeki tüm satırları sonuç veri çerçevesine dahil eder. Ortak olan satırlar birleştirilir, eşleşmeyen satırlar ise eksik değerlerle (**NA**) tamamlanır. Bu yöntem, iki veri çerçevesinin birleşim kümesini oluşturur. **Inner join** ise yalnızca her iki veri çerçevesinde ortak olan satırları içerir ve eşleşmeyen satırları dışlar. Bu yöntem, iki veri çerçevesinin kesişim kümesini oluşturur. **Full join** eksik değerler barındırırken, **Inner join** sadece ortak değerleri içerdiği için eksik değer içermez.

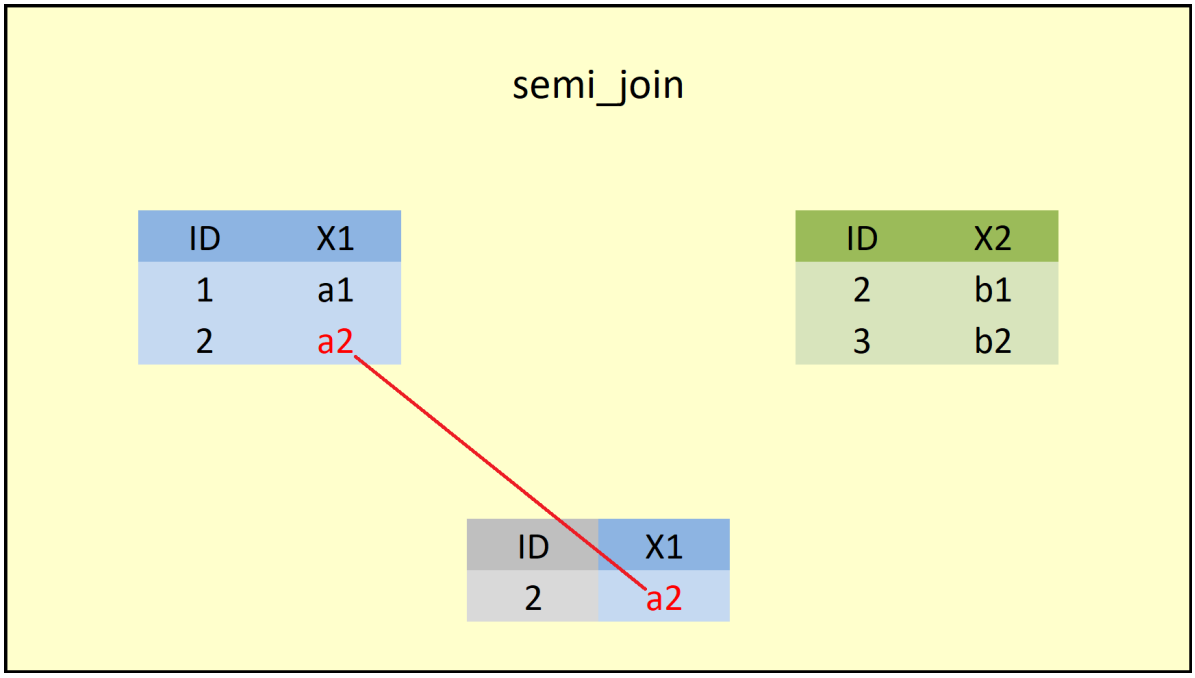
6.3.1.6 Semi Join - Yarı Birleştirme İşlemi

```r
# Gerekli paketin yüklenmesi
library(dplyr)
```

```r
# Yarı birleştirme işlemi: 'semi_join' kullanımı
semi_join(data1, data2, by = "ID") # 'ID' sütununa göre yarı birleştirme
```

```
  ID X1
1  2 a2
```

- **`semi_join`**, **`dplyr`** paketinin bir fonksiyonudur ve yarı birleştirme (semi join) işlemini gerçekleştirir.
- **Semi Join**, yalnızca birinci veri çerçevesindeki (data1) satırları döndürür, ancak bu satırlar ikinci veri çerçevesiyle (data2) eşleşen satırlardır.
- **`by = "ID"`**: Eşleşmenin **ID** sütunu üzerinden yapılacağını belirtir.
- Eşleşen satırlara yalnızca birinci veri çerçevesindeki sütunlar eklenir, ikinci veri çerçevesinin sütunları eklenmez.



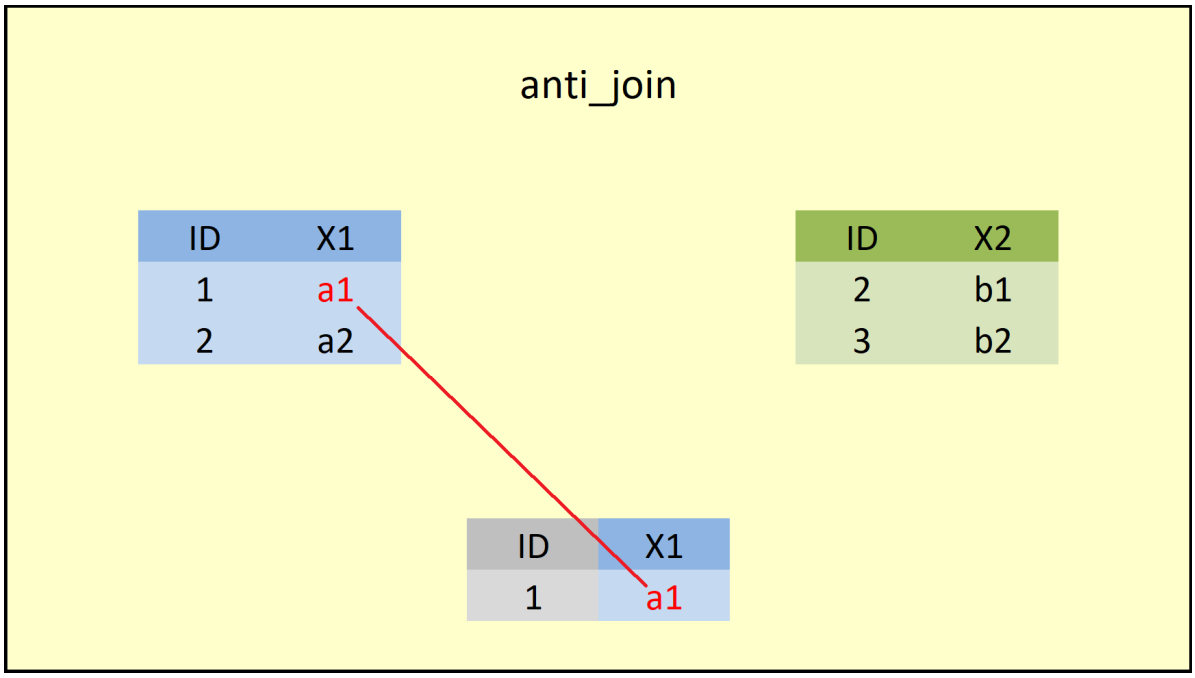
Semi join ve **Inner join** arasındaki temel fark, sonuç veri çerçevesinin içeriğidir. **Inner join**, her iki veri çerçevesinde ortak olan satırları birleştirir ve sonuç veri çerçevesine her iki veri çerçevesinin sütunlarını ekler. **Semi join** ise sadece birinci veri çerçevesindeki ortak satırları döndürür ve ikinci veri çerçevesinin sütunlarını sonuçta dahil etmez. **Semi join**, eşleşen satırları birinci veri çerçevesi perspektifinden filtrelemek için kullanılırken, **Inner join**, iki veri çerçevesinin kesişimini oluşturur.

6.3.1.7 Anti Join - Anti Birleştirme İşlemi

```
# Gerekli paketin yüklenmesi
library(dplyr)

# Anti birleştirme işlemi: 'anti_join' kullanımı
anti_join(data1, data2, by = "ID") # 'ID' sütununa göre anti join işlemi
```

```
  ID X1
1  1 a1
```

- **anti_join**, **dplyr** paketinin bir fonksiyonudur ve ters birleştirme (anti join) işlemini gerçekleştirir.
- **Anti Join**, birinci veri çerçevesindeki (data1) ve ikinci veri çerçevesindeki (data2) satırları karşılaştırır ve yalnızca ikinci veri çerçevesiyle eşleşmeyen birinci veri çerçevesi satırlarını döndürür.
- **by = "ID"**: Eşleşmenin **ID** sütunu üzerinden yapılacağını belirtir.



6.3.2 Veri Eklemeler: `bind_rows()`, `bind_cols()`

6.3.2.1 Satır Ekleme: `bind_rows()`

`bind_rows()`, birden fazla veri çerçevesini satır bazında birleştirmek için kullanılan bir fonksiyondur. Bu fonksiyon, sütun isimlerini eşleştirerek çalışır ve eksik sütunlar varsa bu sütunlara eksik değerler (**NA**) atar. **`rbind()`** ile farkı, **`bind_rows()`**'un sütun sıralarına bağlı kalmadan sütun isimlerini dikkate alarak birleştirme yapabilmesidir. Ayrıca, **`bind_rows()`**, sütunları eksik olan veri çerçevelerinde hata vermek yerine eksik değerler ekleyerek birleştirmeyi tamamlar. Buna karşın, **`rbind()`**, veri çerçevelerindeki sütun isimlerinin ve sıralarının birebir aynı olmasını gerektirir; aksi halde hata verir. Bu nedenle, **`bind_rows()`**, esneklik ve kullanım kolaylığı açısından genellikle daha avantajlıdır.

```
# Örnek veri çerçevesi 1: data1 oluşturma
data1 <- data.frame(
  x1 = 1:5,          # Sayılar 1'den 5'e kadar
  x2 = letters[1:5]  # Harfler 'a' ile 'e' arasında
)
data1
```

```
  x1 x2
1  1  a
2  2  b
3  3  c
4  4  d
5  5  e
```

```
# Örnek veri çerçevesi 2: data2 oluşturma
data2 <- data.frame(
  x1 = 0,            # Sabit bir değer
  x3 = 5:9           # Sayılar 5'ten 9'a kadar
)
data2
```

```
  x1 x3
1  0  5
2  0  6
3  0  7
4  0  8
5  0  9
```

```r
# Örnek veri çerçevesi 3: data3 oluşturma
data3 <- data.frame(
  x3 = 5:9,          # Sayılar 5'ten 9'a kadar
  x4 = letters[5:9]  # Harfler 'e' ile 'i' arasında
)
data3
```

```
  x3 x4
1  5  e
2  6  f
3  7  g
4  8  h
5  9  i
```

Bu kodda, üç farklı veri çerçevesi oluşturuluyor:

1. **data1**: **x1** ve **x2** sütunlarından oluşur. **x1** sayılar, **x2** ise harfler içerir.
2. **data2**: **x1** ve **x3** sütunlarından oluşur. **x1** sabit bir değer, **x3** ise bir sayı aralığıdır.
3. **data3**: **x3** ve **x4** sütunlarından oluşur. **x3** sayılar, **x4** ise harfler içerir.

- **Satır Bazında Birleştirme İşlemi**

```r
# Gerekli paketin yüklenmesi
library(dplyr)

# Satır bazında birleştirme
result <- bind_rows(data1, data2)

# Sonucu görüntüleme
result
```

```
   x1   x2 x3
1   1    a NA
2   2    b NA
3   3    c NA
4   4    d NA
5   5    e NA
6   0 <NA>  5
7   0 <NA>  6
8   0 <NA>  7
9   0 <NA>  8
10  0 <NA>  9
```

- `bind_rows()`, `dplyr` paketinden bir fonksiyon olup birden fazla veri çerçevesini satır bazında birleştirir.
- Sütunlar eşleştiği sürece, eksik sütunlara **NA** atanarak işlemi tamamlar.
- Bu örnekte, `data1` ve `data2` satır bazında birleştirilir ve sonuçta her iki veri çerçevesinin birleşimi elde edilir.
- Eğer `data1` ve `data2`'de eksik sütunlar varsa, **bind_rows()** eksik sütunları **NA** ile doldurur.

6.3.2.2 Sütun ekleme: `bind_cols()`

`bind_cols()`, birden fazla veri çerçevesini sütun bazında birleştirmek için kullanılan bir fonksiyondur. Bu fonksiyon, satır sayılarını dikkate alarak veri çerçevelerini yan yana birleştirir. Eğer veri çerçevelerinde farklı sayıda satır varsa, eksik satırlar için **NA** eklenir. `cbind()` ile farkı, `bind_cols()`'un daha esnek olmasıdır; sütunları birleştirirken veri çerçevelerinin isimlendirilmiş sütun yapılarını korur ve modern veri işleme ihtiyaçlarına daha uygun şekilde çalışır. Buna karşılık, `cbind()`, sütun isimlerine bakmaz ve uyumsuz satır sayılarına sahip veri çerçevelerinde hata verir. Bu nedenle, `bind_cols()`, sütun bazında birleştirme işlemlerinde kullanım kolaylığı ve hata toleransı açısından daha avantajlıdır.

- **Sütun Bazında Birleştirme İşlemi**

```
# Gerekli paketin yüklenmesi
library(dplyr)

# Sütun bazında birleştirme
result <- bind_cols(data1, data3)

# Sonucu görüntüleme
result
```

```
  x1 x2 x3 x4
1  1  a  5  e
2  2  b  6  f
3  3  c  7  g
4  4  d  8  h
5  5  e  9  i
```

- `bind_cols()`, `dplyr` paketinden bir fonksiyon olup birden fazla veri çerçevesini sütun bazında birleştirir.
- Satır sayılarını dikkate alır; eğer veri çerçevelerindeki satır sayıları eşleşmezse, eksik satırlar için **NA** ekler.
- Bu örnekte, `data1` ve `data3` sütun bazında birleştirilir ve iki veri çerçevesi yan yana eklenerek yeni bir veri çerçevesi oluşturulur.

6.4 Metin (String) Manipülasyonu

6.4.1 Temel Metin Manipülasyonu

6.4.1.1 Metin Birleştirme

`str_c()`, stringr paketinde bulunan ve metinleri birleştirmek için kullanılan güçlü bir fonksiyondur. Bu fonksiyon, birden fazla metin girdisini yan yana getirerek tek bir metin oluşturur. `str_c()` fonksiyonu, temel birleştirme işlemlerinin yanı sıra, `sep` argümanı ile birleştirme sırasında kullanılacak ayırıcı karakteri belirleme, vektörleri birleştirme ve farklı veri tiplerini metin olarak birleştirme gibi birçok esnek özellik sunar.

- T**emel Metin Birleştirme İşlemi (`str_c()`)**

```
# Gerekli paketlerin yüklenmesi
# install.packages("stringr")
library(stringr)

# Örnek metinler
text1 <- "Merhaba"
text2 <- "Dunya"
text3 <- "!"

# Temel metin birleştirme
result_basic <- str_c(text1, text2, text3)

# Sonucu görüntüleme
result_basic
```

```
[1] "MerhabaDunya!"
```

1. **`str_c()` Fonksiyonu:**
   - **Açıklama**: `str_c()` fonksiyonu, **`stringr`** paketine ait bir fonksiyondur ve bir veya daha fazla metni birleştirmek için kullanılır. Bu fonksiyon, verilen metinleri ardışık olarak birleştirir.
   - **Argümanlar**: **`text1`, `text2`, `text3`**: Birleştirilecek metinlerdir.
   - **Sonuç**: Verilen metinler, aralarındaki boşluk veya herhangi bir ek karakter olmadan doğrudan birleştirilir.
2. **Birleştirme İşlemi:**

- **Açıklama**: `str_c(text1, text2, text3)` kodu, `text1`, `text2`, ve `text3` metinlerini birleştirir. Sonuç olarak `"MerhabaDunya!"` metni elde edilir.

- **`sep` Argümanı ile Metin Birleştirme**

```
# Gerekli paketlerin yüklenmesi
library(stringr)

# Örnek metinler
text1 <- "Merhaba"
text2 <- "Dunya"
text3 <- "!"

# sep argümanı ile metin birleştirme
result_sep <- str_c(text1, text2, text3, sep = " ")
# 'sep = " "', metinler arasına boşluk ekler.

# Sonucu görüntüleme
result_sep
```

```
[1] "Merhaba Dunya !"
```

- **`str_c()` Fonksiyonu ile Veri Çerçevesinde Metin Birleştirme**

```
# Örnek veri çerçevesi oluşturma
data <- data.frame(
  ad = c("Ali", "Ayse", "Veli"),
  soyad = c("Yilmaz", "Demir", "Kara"),
  yas = c(25, 30, 28))

# Gerekli kütüphanenin yüklenmesi
library(stringr)

# 'ad_soyad_yas' adında yeni bir sütun oluşturuluyor ve bu sütun,
# 'ad', 'soyad' ve 'yas' sütunlarının birleşiminden oluşuyor
data$ad_soyad_yas <- str_c(   # str_c() fonksiyonu bu birleştirme işlemini yapıyor.
  data$ad,           # 'ad' sütunundaki değerler
  data$soyad,        # 'soyad' sütunundaki değerler
  " (",              # Metin içerisinde sabit bir karakter dizisi, parantez açma
  data$yas,          # 'yas' sütunundaki değerler (örneğin: 25)
  " yas)",
  # Metin içerisinde sabit bir karakter dizisi, parantez kapama ve 'yas' kelimesi
```

```
  sep = " "          # Birleştirilen metin parçaları arasına boşluk ekleniyor.
)

# Sonuç veri çerçevesini görüntüleme
data
```

```
    ad  soyad yas           ad_soyad_yas
1  Ali Yilmaz  25 Ali Yilmaz  ( 25  yas)
2 Ayse  Demir  30 Ayse Demir  ( 30  yas)
3 Veli   Kara  28  Veli Kara  ( 28  yas)
```

i str_c() vs. paste()

`str_c()` (stringr paketi) ve `paste()` (base R) Karşılaştırması

```
# stringr::str_c() örneği
library(stringr)
metin1 <- c("a", "b", "c")
metin2 <- c(1, 2, 3)
sonuc_str_c <- str_c(metin1, metin2, sep = "-")
sonuc_str_c  # "a-1" "b-2" "c-3"
```

```
[1] "a-1" "b-2" "c-3"
```

```
# base::paste() örneği
metin1 <- c("a", "b", "c")
metin2 <- c(1, 2, 3)
sonuc_paste <- paste(metin1, metin2, sep = "-")
sonuc_paste  # "a-1" "b-2" "c-3"
```

```
[1] "a-1" "b-2" "c-3"
```

```
# NA değerler ile paste ve str_c kullanımı
metin3 <- c("a", "b", NA)
metin4 <- c(1, 2, 3)

str_c(metin3, metin4, sep = "-")  # "a-1" "b-2" NA
```

```
[1] "a-1" "b-2" NA
```

```
paste(metin3, metin4, sep = "-") # "a-1" "b-2" "NA-3"
```

```
[1] "a-1"  "b-2"  "NA-3"
```

```
# collapse argümanı ile paste kullanımı
paste(metin1, collapse = ", ") # "a, b, c"
```

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

```
# str_c()'de collapse argümanı yoktur.
```

6.4.1.2 Metin Uzunluğu

`str_length()` fonksiyonu, stringr paketine ait bir fonksiyondur ve verilen metnin karakter sayısını döndürür. Bu fonksiyon, boşluklar da dahil olmak üzere tüm karakterleri sayar. Örnek metinler oluşturularak farklı uzunluklardaki ve özelliklerdeki (boşluklu, boşluksuz) metinlerin uzunlukları `str_length()` fonksiyonu ile hesaplanır. Ardından `cat()` fonksiyonu kullanılarak sonuçlar ekrana yazdırılır. `str_length()` fonksiyonunun bir diğer kullanım şekli de, bir veri çerçevesindeki metinlerin uzunluklarını hesaplayıp, yeni bir sütuna kaydetmektir. Bu işlem sırasında `stringAsFactors = FALSE` argümanı kullanılarak metinlerin faktör değil, string olarak kalması sağlanır.

- **`str_length()` Fonksiyonu ile Metin Uzunluğu Hesaplama**

```
# Gerekli kütüphanenin yüklenmesi
library(stringr)

# str_length() fonksiyonu, stringr paketine ait bir fonksiyondur ve verilen metnin
# karakter sayısını döndürür. Bu fonksiyon, boşluklar da dahil olmak üzere tüm
# karakterleri sayar.

# Örnek metinler
text1 <- "Hello World"
text2 <- "  Space  "
text3 <- "MerhabaDunya!"

# Metinlerin uzunlukları str_length() fonksiyonu ile hesaplanır.
length1 <- str_length(text1)
length2 <- str_length(text2)
length3 <- str_length(text3)

# cat() fonksiyonu ile sonuçlar ekrana yazdırılır.
cat("Text 1:'",text1, "\n", "'- Length:", length1, "\n")
```

```
Text 1:' Hello World
 '- Length: 11
```

```
cat("Text 2: '", text2, "' - Length: ", length2)
```

```
Text 2: '   Space   ' - Length:  9
```

```
cat("Text 3: '", text3, "' - Length: ", length3, "\n")
```

```
Text 3: ' MerhabaDunya! ' - Length:  13
```

cat() fonksyonu ve işlevleri

`cat()` Fonksiyonu

- **Temel İşlevi:** `cat()` fonksiyonu, R'da metinleri ve diğer değerleri (sayılar, mantıksal değerler vb.) ekrana yazdırmak için kullanılan bir temel fonksiyondur.
- **Çıktı Biçimi:** `cat()`, print() fonksiyonuna göre daha basit bir çıktı üretir. Genellikle, çıktıya ek bilgiler (satır numaraları, değişken isimleri vb.) eklemez, sadece verilen metni ve değerleri yan yana yazdırır.
- **Birleştirme:** `cat()` fonksiyonu, birden fazla metin veya değeri birleştirmek için kullanılabilir. Bu, farklı türdeki bilgileri aynı satırda görüntülemek için oldukça kullanışlıdır.
- **Ayırıcı Karakter:** `cat()` fonksiyonu varsayılan olarak metinleri bir boşlukla ayırır, ancak bu davranış sep argümanı ile değiştirilemez. sep argümanı `print()` fonksiyonunda vardır.
- **Yeni Satır Karakteri:** `cat()` fonksiyonunda yeni bir satıra geçmek için `\n` kaçış dizisi (escape sequence) kullanılır. Bu sayede çıktı daha düzenli hale getirilebilir.
- **Genel Kullanım:** `cat()`, özellikle döngüler içinde veya koşullu çıktıların oluşturulmasında kullanışlıdır. Gelişmiş çıktılar veya değişkenlere ait bilgilerin görüntülenmesi için print() veya message() fonksiyonları daha uygun olabilir.

Özet:
`cat()` fonksiyonu, R'da metin ve değerleri basit bir şekilde ekrana yazdırmak için kullanılan bir araçtır. Özellikle birden fazla metni birleştirmek ve formatlı çıktılar oluşturmak için kullanışlıdır. `print()` fonksiyonuna göre daha yalın bir çıktı verir ve sep argümanı kullanılamaz, ancak `\n` kaçış dizisi ile yeni satır eklenebilir.

- **`str_length()` Fonksiyonu ile Veri Çerçevesinde Yeni Sütun Eklemek**

```r
# Gerekli paketlerin yüklenmesi
library(stringr)

# str_length() fonksiyonunun bir diğer kullanım şekli de, bir veri çerçevesindeki
# metinlerin uzunluklarını hesaplayıp, yeni bir sütuna kaydetmektir.

# Örnek bir veri çerçevesi oluşturulur
data_fruits <- data.frame(
  text = c("apple", "banana", "cherry", "date"),
  stringsAsFactors = FALSE # Metinlerin faktör değil string olarak kalmasını sağlar
)

# str_length() fonksiyonu kullanılarak metinlerin uzunlukları hesaplanır ve
# yeni bir sütuna eklenir
data_fruits$text_length <- str_length(data_fruits$text)

# Sonucu görüntüleme
data_fruits
```

```
    text text_length
1  apple           5
2 banana           6
3 cherry           6
4   date           4
```

$ İşareti ile Yeni Sütun/Değişken Eklemek

R dilinde, veri çerçevelerine yeni sütun eklemek için **$** işareti sıkça kullanılır. Bu, veri çerçevesine doğrudan yeni bir sütun eklemeye olanak tanır.

Veri Çerçevesinde Yeni Değişken Ekleme:

Veri çerçevesine yeni bir sütun (veya değişken) eklemek için şu adımları izlersiniz:

- **$ işareti** kullanarak yeni sütunun adını belirtirsiniz.
- Sağdaki kısımda, eklemek istediğiniz değişkenin hesaplamasını veya değerini belirtirsiniz.

```r
# data$new_column <- 5  # Tüm satırlar için 5 değeri eklenir.
```

Bu işlem, `data` veri çerçevesine `new_column` adında yeni bir sütun ekler ve tüm satırlara 5 değeri atar.

Veri Çerçevesine Hesaplanan Değer Ekleme:

Bir sütun, başka sütunlardaki değerlerin hesaplanması ile de eklenebilir. Örneğin, metin uzunlukları, sayılar, vb. hesaplanarak yeni bir sütun oluşturulabilir.

```
# data$new_column <- data$var1 + data$var2  # var1 ve var2 sütunlarının toplamı
```

Bu örnekte, **`data$new_column`** sütunu, **`data$var1`** ve **`data$var2`** sütunlarının toplamını içerir.

Özet:

- **`$` işareti**, veri çerçevesine yeni bir sütun eklerken kullanılır. Yeni sütunun adı belirlenir ve sağdaki kısımda bu sütuna atanacak değerler veya hesaplamalar belirtilir.
- **Yeni sütun eklemek için**: `data$new_column <- value` şeklinde kullanılır. Bu, veri çerçevesine **`new_column`** adında bir sütun ekler ve **`value`**'yu bu sütuna atar.
- Bu yöntem, veri çerçevesinde hızlı bir şekilde yeni değişkenler oluşturmanızı sağlar ve veri manipülasyonu için oldukça kullanışlıdır.

6.4.1.3 Metin Dönüştürme

- **Büyük/Küçük Harf Dönüşümleri:** **`str_to_lower()`** **ve** **`str_to_upper()`**

Bu fonksiyonlar, karakter dizilerindeki harfleri tamamen küçük veya büyük harfe dönüştürmek için kullanılır. Bu işlem, metin normalizasyonu ve karşılaştırma gibi işlemlerde oldukça faydalıdır.

`str_to_lower()`: Metni tamamen küçük harfe dönüştürür.

`str_to_upper()`: Metni tamamen büyük harfe dönüştürür.

```
# Gerekli paketlerin yüklenmesi
library(stringr)

# Küçük harfe dönüştürme
text <- "Merhaba Dunya!"
lower_text <- str_to_lower(text)
print(lower_text) # Çıktı: "merhaba dunya!"
```

```
[1] "merhaba dunya!"
```

```
# Büyük harfe dönüştürme
upper_text <- str_to_upper(text)
print(upper_text) # Çıktı: "MERHABA DÜNYA!"
```

```
[1] "MERHABA DUNYA!"
```

- **Alt Dizeleri Çıkarma:** `str_sub()`

Bu fonksiyon, bir karakter dizisinden belirli bir başlangıç ve bitiş pozisyonuna göre alt dizeler çıkarmak veya mevcut bir alt diziyi değiştirmek için kullanılır.

```
# Gerekli kütüphanenin yüklenmesi
library(stringr)

# Örnek bir telefon numarası tanımlanır
phone_number <- "+90 123 456 7890"

# str_sub() fonksiyonu kullanılarak telefon numarasının alan kodu alınır
area_code <- str_sub(phone_number, 1, 3) # 1.'den 3. karaktere kadar olan kısmı alır.

# Sonucu ekrana yazdırma
print(area_code) # Çıktı: "+90"
```

```
[1] "+90"
```

- **Baş ve Sondaki Boşlukları Temizleme:** `str_trim()`

Bu fonksiyon, bir metnin başındaki ve sonundaki boşlukları temizlemek için kullanılır. `side` argümanıyla, sadece baştan, sondan veya her iki taraftan boşlukları temizleme seçeneği sunar.

`str_trim(string)`: Baş ve sondaki boşlukları temizler.

`str_trim(string, side = "left")`: Sadece baştaki boşlukları temizler.

`str_trim(string, side = "right")`: Sadece sondaki boşlukları temizler.

```
# Gerekli kütüphanenin yüklenmesi
library(stringr)

# Kullanıcı girdisi (boşluklu)
user_input <- "   Kullanici Adi   "

# str_trim() fonksiyonu kullanılarak baştaki ve sondaki boşluklar temizlenir
cleaned_input <- str_trim(user_input)  # Boşlukları temizler

# Sonucu ekrana yazdırma
print(cleaned_input) # Çıktı: "Kullanici Adi"
```

```
[1] "Kullanici Adi"
```

6.4.2 Desen Eşleştirme ve Değiştirme

`str_detect()` fonksiyonu, bir karakter vektöründe belirli bir desenin varlığını kontrol etmek için kullanılır. Bu fonksiyon, bir **mantıksal vektör** (TRUE/FALSE) döndürür.

```
library(stringr)

#str_detect(string, #Aranacak metin ya da metin vektörü.
           #pattern) #Aranacak desen (düz metin ya da regex).
```

`str_detect()` Fonksiyonu ile Desen Arama

```
# Gerekli kütüphanenin yüklenmesi
library(stringr)

# Örnek metinler
text_vector <- c("apple", "banana", "cherry", "date", "apricot")

# Desen arama örnekleri

# "a" içeren kelimeler
has_a <- str_detect(text_vector, "a")
has_a
```

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

```
# "ap" ile başlayan kelimeler
has_ap <- str_detect(text_vector, "^ap")
has_ap
```

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

```
# "e" ile biten kelimeler
has_e <- str_detect(text_vector, "e$")
has_e
```

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

```r
# Rakam içeren kelimeler (bu örnekte yok)
has_num <- str_detect(text_vector, "\\d")
has_num
```

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

Yukarıdaki kodda, text_vector adında bir metin vektörü tanımlanır. Ardından, `str_detect()` fonksiyonu kullanılarak bu vektördeki metinlerde belirli desenler aranır. Örneğin, "a" harfini içeren kelimeler, "ap" ile başlayan kelimeler, "e" ile biten kelimeler ve rakam içeren kelimeler kontrol edilir. Sonuçlar, `TRUE` veya `FALSE` değerlerinden oluşan mantıksal bir vektör olarak döndürülür.

6.4.2.1 Desen Değiştirme: `str_replace()` ve `str_replace_all()`

`str_replace()` ve `str_replace_all()` fonksiyonları, metin içindeki belirli desenleri değiştirmek için kullanılır. `str_replace()` fonksiyonu, desenin ilk eşleşmesini değiştirirken, `str_replace_all()` fonksiyonu desenin tüm eşleşmelerini değiştirir.

```r
# Gerekli kütüphanenin yüklenmesi
library(stringr)

# Örnek metin
text2 <- "red apple, green apple, red banana"

# İlk eşleşmeyi değiştirme
replaced_text1 <- str_replace(text2, "apple", "orange")
replaced_text1
```

```
[1] "red orange, green apple, red banana"
```

```r
# Tüm eşleşmeleri değiştirme
replaced_text2 <- str_replace_all(text2, "apple", "orange")
replaced_text2
```

```
[1] "red orange, green orange, red banana"
```

6.4.3 Regex Temelleri (Düzenli İfadeler - Regular Expressions)

Temel Regex Karakterleri

Karakter	Anlamı	Örnek
`.`	Herhangi bir karakter	`a.b` → “acb”, “a2b”
`*`	Önceki karakter 0 veya daha fazla kez	`ab*` → “a”, “ab”, “abb”
`+`	Önceki karakter 1 veya daha fazla kez	`ab+` → “ab”, “abb”
`?`	Önceki karakter 0 veya 1 kez	`ab?` → “a”, “ab”
`[]`	Karakter kümesi	`[aeiou]` → “a”, “e”, “o”
`^`	Başlangıç	`^abc` → “abcdef”
`$`	Bitiş	`xyz$` → “123xyz”
`\d`	Rakam (digit)	`\d+` → “123”, “45”
`\w`	Kelime karakteri	`\w+` → “abc”, “123”, “word1”
`\s`	Boşluk karakteri	`\s+` → ” “,” ”

Regex ile `str_detect()`, `str_replace()`, ve `str_replace_all()` Kullanımı

```
# Örnek metinler
metin <- c("abc123", "def456", "ghi")
# - "abc123": Hem harf hem rakam içerir.
# - "def456": Hem harf hem rakam içerir.
# - "ghi": Harf içerir.

# Desen arama işlemi
sonuc <- str_detect(metin, "\\d+")
# - "\\d+": Bir veya daha fazla rakamı arayan düzenli ifade (regex).
# - Dönen sonuç: Her bir öğe için TRUE/FALSE (mantıksal vektör).

print(sonuc)
```

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

```
# Açıklama: Her bir öğe en az bir rakam içerdiği için tüm sonuçlar TRUE döner.
```

Örnek Kullanım:

```r
# Gerekli paketlerin yüklenmesi
# install.packages("kableExtra")
library(kableExtra)
library(tidyverse)
library(stringr)

# Örnek tibble oluşturma
data_regex <- tibble(
  ID = 1:5,
  Customer_Info = c(
    " John Doe, john.doe@example.com ",
    " Jane Smith, JANE.SMITH@example.com ",
    " Bob Brown, BOB.BROWN@example.com ",
    " Alice Johnson, alice.johnson@example.com ",
    " Carol White, carol.white@example.com "
  )
)

# Veri manipülasyonu
processed_data_regex <- data_regex %>%
  # 1. str_trim(): Baş ve sondaki boşlukları temizleme
  mutate(Customer_Info = str_trim(Customer_Info)) %>%

  # 2. str_detect(): E-posta adreslerinin doğru formatta olup olmadığını kontrol etme
  mutate(Valid_Email = str_detect(Customer_Info, "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z

  # 3. str_sub(): İsim ve e-posta adreslerini ayıklama
  mutate(
    Name = str_sub(Customer_Info, 1, str_locate(Customer_Info, ",")[, 1] - 1),
    # Virgülden önceki kısmı isim olarak al
    Email = str_sub(Customer_Info, str_locate(Customer_Info, ",")[, 1] + 2)
    # Virgülden sonraki kısmı e-posta olarak al
  ) %>%

    # 4. str_to_lower() ve str_to_upper(): İsimleri büyük harf, e-posta adreslerini
    # küçük harf yapma
  mutate(
    Name = str_to_upper(Name),
    Email = str_to_lower(Email)
  ) %>%

  # 5. str_length(): İsmin uzunluğunu hesaplama
```

```
  mutate(Name_Length = str_length(Name)) %>%

  # 6. str_replace(): E-postalardaki "example.com" kısmını "domain.com" ile değiştirme
  mutate(Email = str_replace(Email, "example.com", "domain.com")) %>%

  # 7. str_c(): İsim ve yeni e-posta adresini birleştirerek tam müşteri bilgisi oluşturma
  mutate(Updated_Info = str_c(Name, " <", Email, ">"))

# Sonuçları göster
kableExtra::kable(processed_data_regex)
```

ID	Customer_Info	Valid_Email	Name	Email	Name_Length	Updated_Info
1	John Doe, john.doe@example.com	TRUE	JOHN DOE	john.doe@domain.com	8	JOHN DOE john.doe@domain.com
2	Jane Smith, JANE.SMITH@example.com	TRUE	JANE SMITH	jane.smith@domain.com	10	JANE SMITH jane.smith@domain.com
3	Bob Brown, BOB.BROWN@example.com	TRUE	BOB BROWN	bob.brown@domain.com	9	BOB BROWN bob.brown@domain.com
4	Alice Johnson, alice.johnson@example.com	TRUE	ALICE JOHNSON	alice.johnson@domain.com	13	ALICE JOHNSON alice.johnson@domain.com
5	Carol White, carol.white@example.com	TRUE	CAROL WHITE	carol.white@domain.com	11	CAROL WHITE carol.white@domain.com

6.4.4 Janitor Paketi ile Veri Temizleme

```
# Gerekli paketlerin yüklenmesi
library(janitor)

# Örnek veri çerçevesi
data_janitor <- data.frame(
  "First Name" = c("Ali", "Ayse", "Mehmet"),
  "LastNAME" = c("Kara", "Demir", "Yilmaz"),
  "DateOFBirth" = c("1990-01-01", "1985-05-12", "2000-07-22"),
  stringsAsFactors = FALSE
)

# Orijinal veri çerçevesi
print(data_janitor)
```

```
  First.Name LastNAME DateOFBirth
1        Ali     Kara  1990-01-01
2       Ayse    Demir  1985-05-12
3     Mehmet   Yilmaz  2000-07-22
```

```r
# clean_names() fonksiyonu ile sütun isimlerini temizleme
cleaned_data_janitor <- clean_names(data_janitor)

# Temizlenmiş veri çerçevesi
print(cleaned_data_janitor)
```

```
  first_name last_name date_of_birth
1        Ali      Kara    1990-01-01
2       Ayse     Demir    1985-05-12
3     Mehmet    Yilmaz    2000-07-22
```

6.5 Fonksiyonlarla Veri Manipülasyonu

6.5.1 Fonksiyon Uygulamaları:

- `apply()`, `sapply()`, `lapply()` gibi fonksiyonlar ile veri üzerinde işlemler gerçekleştirme.

6.5.2 Gruplama İşlemleri:

- `tapply()` fonksiyonu ile gruplar üzerinde hesaplamalar yapma.

6.5.3 Kesim ve Sınıflandırma:

- `cut()` fonksiyonu ile sayısal veriyi sınıflara ayırma.

Referanslar

https://www.tidyverse.org/

https://dplyr.tidyverse.org/

https://tibble.tidyverse.org/

https://stringr.tidyverse.org/

https://r4ds.hadley.nz/

https://mine-cetinkaya-rundel.github.io/r4ds-solutions/

https://www.geeksforgeeks.org/merge-function-in-r/

https://statisticsglobe.com/r-dplyr-join-inner-left-right-full-semi-anti

https://statisticsglobe.com/r-bind_rows-bind_cols-functions-dplyr-package

https://r-coder.com/r-data-manipulation/

https://r-primers.andrewheiss.com/

https://app.datacamp.com/learn/courses/data-manipulation-with-dplyr

7 Veri Temizleme

7.1 Eksik Verilerle Çalışma

Büyük veri, hacim, hız, çeşitlilik, doğruluk ve değer gibi özellikleriyle öne çıkar. Bu karmaşık veri dünyasında anlamlı bilgiler çıkarma ve analiz süreçlerini yönetme görevi veri bilimine düşmektedir. Ancak, büyük veri analizi sürecinde en sık karşılaşılan zorluklardan biri eksik verilerdir. Eksik veriler, genellikle yanıt eksiklikleri veya veri kaybı gibi nedenlerle ortaya çıkar ve bu durum, analiz sonuçlarının doğruluğunu ve güvenilirliğini tehlikeye atar. Eksik veri problemi, istatistiksel gücün azalması, parametre tahminlerinde yanlılık, örneklemlerin temsil gücünün zayıflaması ve analiz süreçlerinin karmaşıklaşması gibi sorunlara yol açabilir. Bu nedenle, eksik verilerle doğru bir şekilde başa çıkmak, sağlam ve güvenilir analiz sonuçları elde etmek için kritik öneme sahiptir.

https://www.rpubs.com/justjooz/miss_data

Eksik veri yönetimi, veri analizi sürecinin temel yapı taşlarından biri olarak değerlendirilmelidir. Bu bağlamda, eksik verilerin tespiti ve görselleştirilmesi için **naniar** paketi kullanılabilir; özellikle `vis_miss()` fonksiyonu, eksik veri desenlerini analiz etmek için etkili bir araçtır. Eksik verileri doldurma yöntemleri arasında, özellikle çok değişkenli veri setleri için uygun olan **mice** paketi dikkat çeker. Bu paket, birden fazla doldurma yöntemi sunarak, veri setinin istatistiksel gücünü ve temsiliyetini artırır. Ayrıca, eksik veri profillerini detaylı bir şekilde analiz etmek ve raporlamak için **dlookr** paketi gibi araçlardan yararlanmak mümkündür. Eksik veri yönetiminde kullanılan bu yaklaşımlar, veri analistlerinin daha doğru öngörüler yapmasını sağlayarak, stratejik karar alma süreçlerine destek olur. Böylece, eksik veri probleminin üstesinden gelmek için yöntem seçimi ve uygulaması, veri analizinde güvenilirlik ve doğruluk açısından vazgeçilmez bir süreçtir.

Veri Seti `airquality`

```
# Gerekli kütüphanelerin yüklenmesi
library(dplyr)
# datasets paketini yükleme (otomatik olarak yüklü olmalı)
library(datasets)

# airquality veri setinin görüntülenmesi
head(airquality, 10)
```

```
   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
```

Airquality Veri Seti

`airquality` veri seti, 1973 yılında New York'ta ölçülen hava kalitesi değerlerini içeren bir veri setidir. Bu veri seti, hava kalitesini etkileyen çeşitli değişkenleri içerir ve çevresel analizler için kullanılır. Veri seti, aşağıdaki değişkenlerden oluşur:

- **Ozone**: Ozon seviyelerini ifade eder (ppb - parts per billion).
- **Solar.R**: Solar radyasyon değerlerini içerir (langley).
- **Wind**: Rüzgar hızını içerir (mph - miles per hour).
- **Temp**: Günlük maksimum sıcaklık ölçümlerini içerir (Fahrenheit).
- **Month**: Ölçümün yapıldığı ayı temsil eder (1-12 arasında).
- **Day**: Ölçümün yapıldığı gün bilgisini içerir (1-31 arasında).

7.1.1 Ad-hoc Yöntemler - Liste Bazlı Silme (Listwise Deletion)

Eksik verilerle başa çıkmak için veri bilimciler tarafından en sık kullanılan yöntemlerden biri, eksik değerlere sahip durumları tamamen çıkarmak ve yalnızca kalan veri setini analiz etmektir. Bu yönteme **liste bazlı silme** veya **tam durum analizi** (complete-case analysis) denir. R programında `na.omit()` fonksiyonu, veri setinde bir veya daha fazla eksik değeri olan tüm durumları kaldırır.

```
head(airquality)
```

```
  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
```

Veri setinde bazı `NA` değerlerini şimdiden gözlemleyebiliyoruz.

Sonraki adımda, `NA` değerleri içeren durumları veri setinden kaldırıyoruz.

- **`na.omit()` ile Eksik Verileri Çıkarma**

```
# Eksik verileri çıkarma
airquality_omit <- na.omit(airquality)

# İlk birkaç satırı görüntüleme
head(airquality_omit)
```

```
  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
7    23     299  8.6   65     5   7
8    19      99 13.8   59     5   8
```

İlk çıktıda `airquality`, eksik değerler (**NA**) hala veri setinde bulunurken, `airquality_omit` veri setinde eksik değerler içeren satırlar tamamen çıkarılmıştır. Bu, satır sayısının azalmasına yol açar.

i Liste bazlı silme

Liste bazlı silme (Listwise deletion) yöntemi, eksik veriler içeren satırları tamamen kaldırdığı için genellikle birkaç nedenden dolayı tercih edilmez:

1. **Veri Kaybı**: Eksik değerlere sahip satırların tamamen silinmesi, veri setinin boyutunu küçültür ve bu da analiz için kullanılabilir bilgi miktarını azaltır. Bu durum, özellikle eksik verilerin oranı yüksekse, analiz sonuçlarını ciddi şekilde etkileyebilir.
2. **Örnekleme Yanlılığı**: Eksik veriler rastgele (MCAR - Missing Completely at Random) değilse, bu yöntemin kullanımı örneklemde yanlılığa neden olabilir. Sonuç olarak, elde edilen analiz sonuçları tüm veri setini veya popülasyonu doğru bir şekilde temsil etmeyebilir.
3. **İstatistiksel Güç Kaybı**: Veri setinin boyutunun küçülmesi, istatistiksel gücü azaltır. Bu da yapılan analizlerin daha az anlamlı sonuçlar üretmesine yol açabilir.
4. **Karmaşık Eksiklik Yapıları**: Eksik veriler farklı desenler izleyebilir ve listwise deletion, bu desenleri dikkate almadan tüm eksik satırları kaldırır. Bu, özellikle eksik verilerin analizin anahtar değişkenlerinde olduğu durumlarda önemli bilgilerin kaybolmasına neden olabilir.

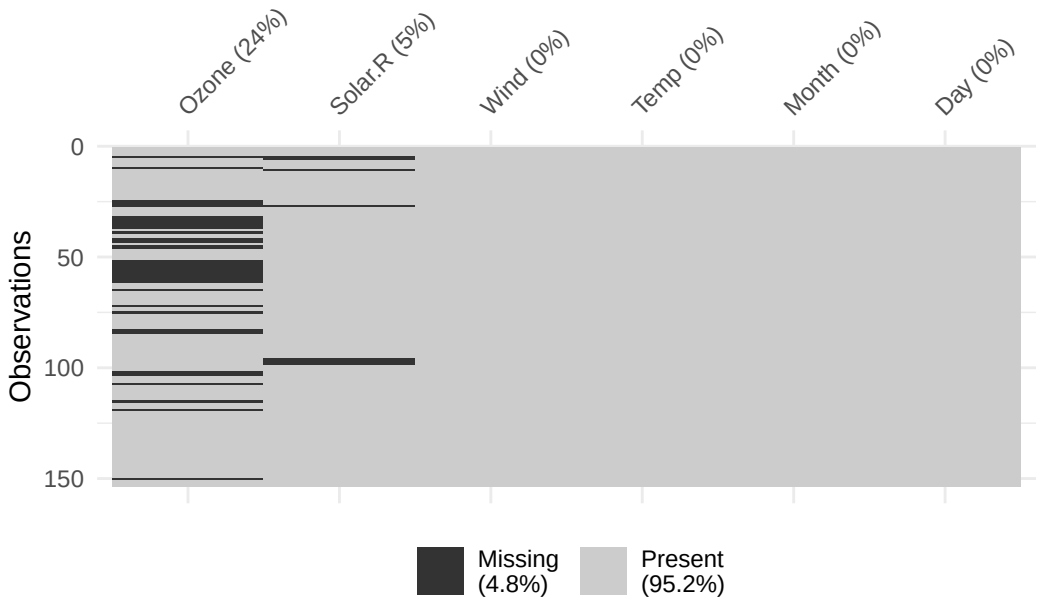
Bu nedenlerle, eksik verilerle başa çıkmak için **çoklu doldurma (multiple imputation)** veya

eksik değerlerin modelleme yöntemleriyle ele alınması gibi daha gelişmiş yöntemler genellikle listwise deletion'a tercih edilir.

7.1.2 Eksik Verilerin Grafik Olarak Tespiti

- **Eksik Verileri Görselleştirme (`naniar` Paketinin Kullanımı)**

```
# naniar kütüphanesini yükleme (eğer yüklü değilse)
# install.packages("naniar")

# naniar kütüphanesini yükleme
library(naniar)

# Eksik verileri görselleştirme
vis_miss(airquality)
```



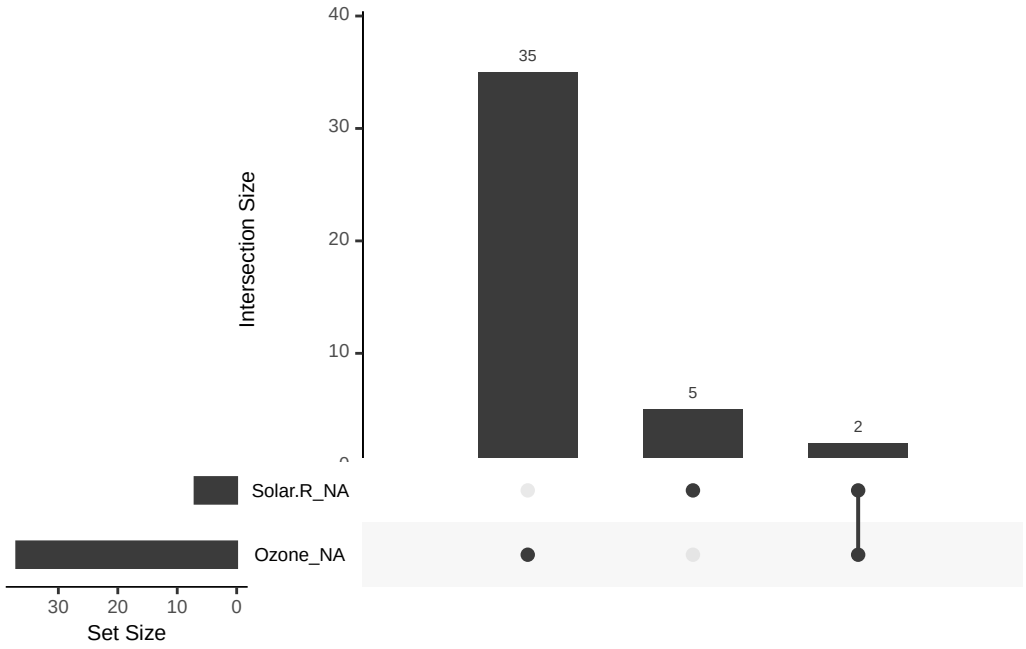
Grafik, `vis_miss()` fonksiyonu ile oluşturulmuş ve **airquality** veri setindeki eksik verilerin genel yapısını göstermektedir.

- **Siyah alanlar** eksik verileri (**Missing**) temsil ederken, **gri alanlar** mevcut verileri (**Present**) temsil eder.

- Ozon değişkeninde %24, Solar.R değişkeninde %5 oranında eksik veri bulunmaktadır. Diğer değişkenler (Wind, Temp, Month, Day) ise eksiksizdir.

7.1.2.1 Eksik Değerlerin Eşzamanlı Görülmesi

```
# naniar kütüphanesini yükleme
library(naniar)

# Eksik verilerin UpSet grafiği ile gösterimi
gg_miss_upset(airquality)
```



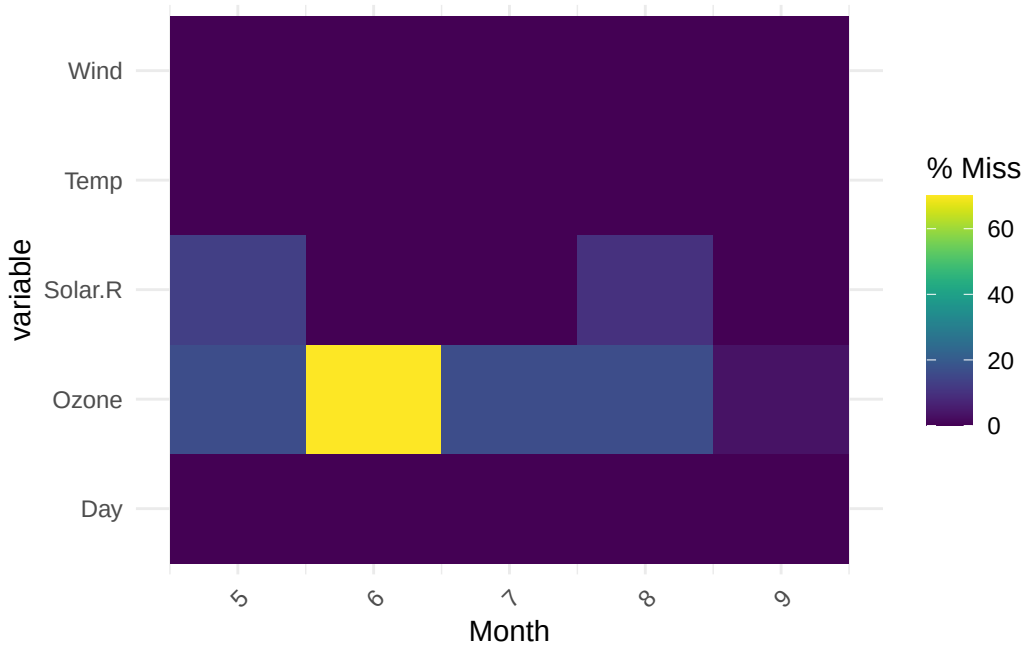
`gg_miss_upset(airquality)` fonksiyonu, `naniar` paketine ait bir fonksiyondur ve eksik değerlerin birlikteliğini (co-occurrence) görselleştirmek için UpSetR paketini kullanarak bir grafik oluşturur. Bu grafik, hangi değişkenlerin birlikte eksik olduğunu ve bu kombinasyonların ne sıklıkta görüldüğünü gösterir.

- **Çubuk grafikler (dikey):** Her bir çubuk, belirli bir eksik değer kombinasyonunu temsil eder. Çubuğun yüksekliği, bu kombinasyonun veri setinde kaç kez tekrarlandığını gösterir.
- **Noktalar ve çizgiler (yatay):** Her bir değişken için bir nokta bulunur. Eğer bir çubukta o değişkenle ilgili nokta doluysa (yani çizgiyle bağlıysa), o kombinasyonda

o değişkende eksik değer olduğu anlamına gelir. Örneğin, sadece "`Ozone`" değişkenine bağlı bir çubuk, sadece "`Ozone`" değerinin eksik olduğu satırları temsil eder. Hem "`Ozone`" hem de "`Solar.R`" değişkenlerine bağlı bir çubuk ise, her iki değişkende de aynı anda eksik değer olan satırları temsil eder.

7.1.2.2 Faktör Düzeyine Göre Veri Eksikliğini Görselleştirme

```
# naniar kütüphanesini yükleme
library(naniar)

# Eksik verileri faktör düzeyine göre görselleştirme
gg_miss_fct(x = airquality, fct = Month)
```



`gg_miss_fct(x = airquality, fct = Month)` fonksiyonu, `naniar` paketine ait bir fonksiyondur ve `airquality` veri setindeki eksik verilerin `Month` değişkenine göre dağılımını görselleştirir. `Month` burada bir faktör (kategorik değişken) olarak kabul edilir ve her bir ay (5, 6, 7, 8, 9) için ayrı bir çubuk gösterilir.

Grafikte şunlar görülebilir:

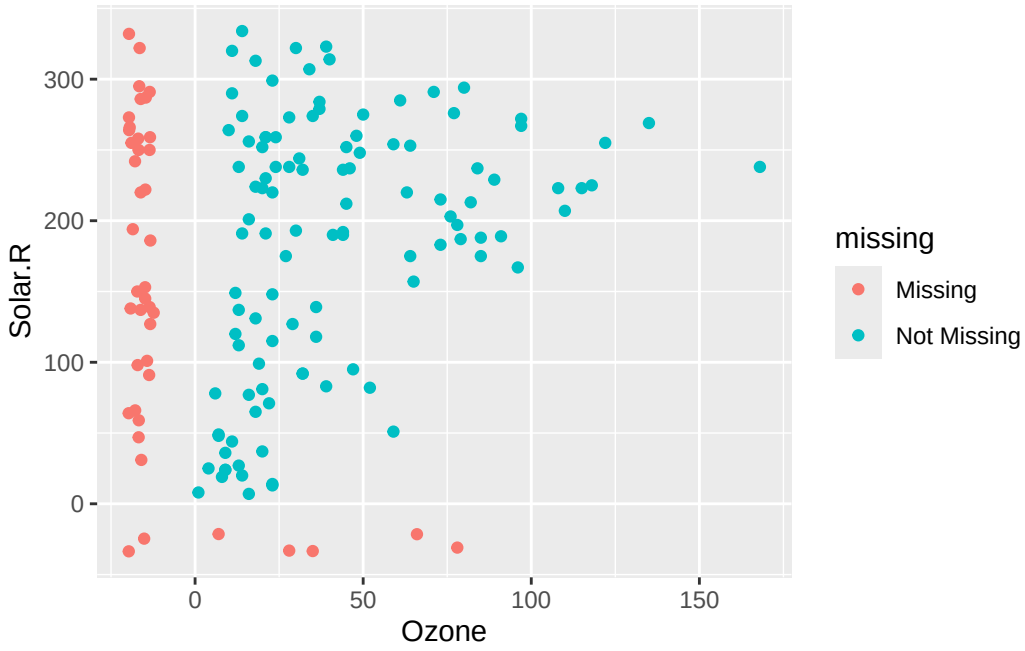
- **X ekseni (Month):** Ayları temsil eder (5 = Mayıs, 6 = Haziran, …, 9 = Eylül).
- **Y ekseni (Missing Percentage):** Eksik veri yüzdesini temsil eder.

- **Çubuklar:** Her bir ay için bir çubuk bulunur. Çubuğun yüksekliği, o ayda ne kadar eksik veri olduğunu (tüm değişkenler için toplamda) gösterir.

Bu grafik, eksik verilerin aylara göre nasıl değiştiğini anlamak için çok faydalıdır. Örneğin, belirli aylarda daha fazla eksik veri olup olmadığını veya eksik verilerin aylara göre bir örüntü izleyip izlemediğini görebilirsiniz. Bu bilgi, veri toplama sürecindeki olası sorunları veya mevsimsel etkileri anlamanıza yardımcı olabilir. Örneğin, eğer belirli bir ayda ölçüm cihazlarında bir arıza olduysa, o ayda daha fazla eksik veri görülebilir.

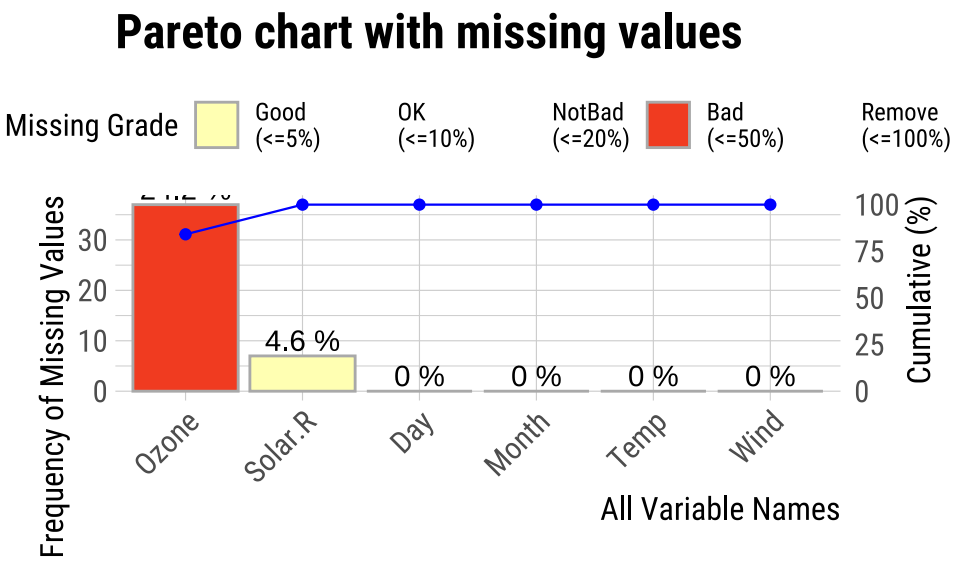
- **Eksik Verileri Faktör Düzeyine Göre Nokta Grafiği ile Görselleştirme**

```
# Gerekli paketlerin yüklenmesi
library(ggplot2)
library(naniar)

# Eksik veri noktalarını görselleştirme
ggplot(airquality, aes(x = Ozone, y = Solar.R)) +
  geom_miss_point()  # Eksik verileri noktalar olarak görselleştirir
```



7.1.2.3 `dlookr` Paketi ile Eksik Veriler

```r
# dlookr kütüphanesini yükleme (gerekli fonksiyon için)
# install.packages("dlookr")
library(dlookr)

# Eksik verilerin Pareto grafiği ile gösterimi
plot_na_pareto(airquality, col = "blue")
```



`plot_na_pareto(airquality)` fonksiyonu, `dlookr` paketine ait bir fonksiyondur ve `airquality` veri setindeki eksik değerleri bir Pareto grafiği ile görselleştirir.

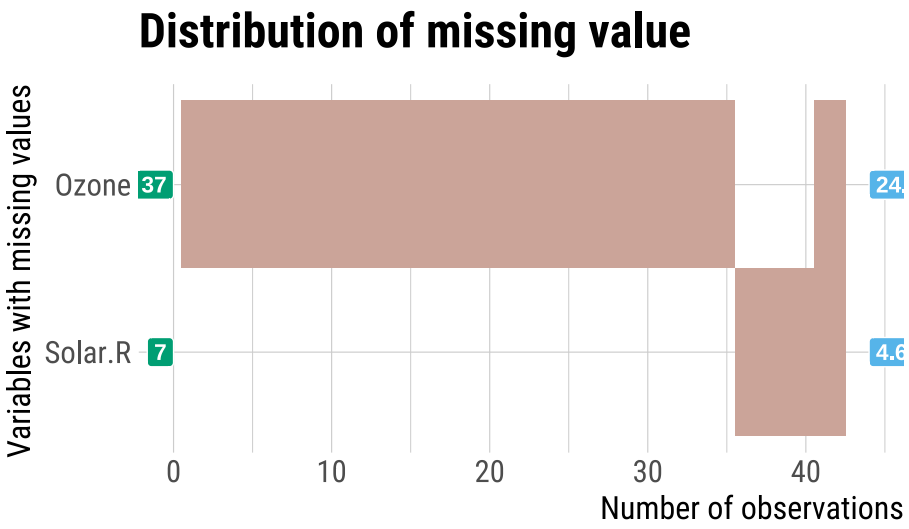
- **X ekseni:** Değişkenleri temsil eder. Değişkenler, eksik değer sayılarına göre en çoktan en aza doğru sıralanmıştır.
- **Sol Y ekseni:** Eksik değer sayısını temsil eder. Her bir değişken için bir çubuk bulunur ve çubuğun yüksekliği o değişkendeki eksik değer sayısını gösterir.
- **Sağ Y ekseni:** Kümülatif eksiklik yüzdesini temsil eder. Çizgi grafiği, değişkenler eklendikçe toplam eksiklik oranının nasıl arttığını gösterir.

Pareto grafiği, hangi değişkenlerde en çok eksik değer olduğunu ve bu değişkenlerin toplam eksikliğe ne kadar katkıda bulunduğunu hızlıca anlamak için kullanışlıdır. Genellikle, birkaç değişkenin toplam eksikliğin büyük bir kısmını oluşturduğu görülür ("80/20 kuralı" olarak da bilinir). Bu grafik, eksik verilerle başa çıkarken önceliklerin belirlenmesine yardımcı olabilir. Örneğin, en çok eksik değere sahip değişkenlere odaklanmak,

genel eksiklik sorununu çözmek için daha etkili bir yaklaşım olabilir. `airquality` örneğinde `Ozone` değişkeninin diğerlerine göre çok daha fazla eksik veriye sahip olduğu kolayca görülebilir.

Eksik Verilerin Hiyerarşik Kümeleme Grafiği ile Gösterimi `dlookr`

```
# dlookr kütüphanesini yükleme
library(dlookr)

# Eksik verilerin hiyerarşik kümeleme grafiği ile gösterimi
plot_na_hclust(airquality, main = "Distribution of missing value")
```

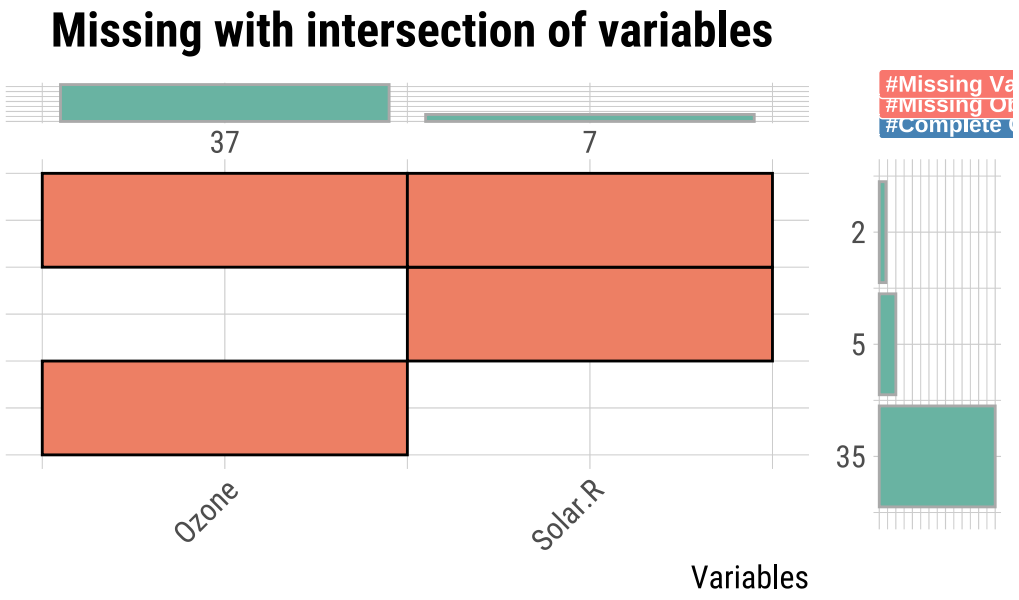


- `plot_na_hclust(airquality, main = "Distribution of missing value")` fonksiyonu, `dlookr` paketine ait bir fonksiyondur ve `airquality` veri setindeki eksik değer örüntülerini hiyerarşik kümeleme (hierarchical clustering) kullanarak görselleştirir.
- `main = "Distribution of missing value"` argümanı, grafiğe bir başlık ekler.

Bu grafik, eksik değerlerin veri setinde rastgele mi dağıldığını yoksa belirli örüntüler izleyip izlemediğini anlamak için çok faydalıdır. Örneğin, belirli satır gruplarının benzer eksik değer örüntülerine sahip olduğunu görmek, veri toplama sürecinde veya verilerin doğasında bir sorun olduğunu gösterebilir. Bu bilgi, eksik verilerle nasıl başa çıkılacağına (örneğin, hangi doldurma yönteminin kullanılacağına) karar verirken önemli bir rol oynayabilir.

Eksik Verilerin Kesişim Grafiği ile Gösterimi `dlookr`

```
# dlookr kütüphanesini yükleme
library(dlookr)

# Eksik verilerin kesişim grafiği ile gösterimi
plot_na_intersect(airquality)
```



`plot_na_intersect(airquality)` fonksiyonu, `dlookr` paketine ait bir fonksiyondur ve `airquality` veri setindeki eksik değerlerin kesişimlerini (yani hangi değişkenlerin aynı satırlarda birlikte eksik olduğunu) görselleştirir.

7.1.3 Eksik Değerlerin Toplam Sayıları ve Oranları

`n_miss` fonksiyonu, verilerdeki tüm `NA` (yani eksik) değerlerinin toplam sayısını döndürür.

7.1.3.1 `NA` olan değerlerin sayısı için:

```r
# naniar kütüphanesini yükleme
library(naniar)

# Eksik değer sayısını hesaplama
n_miss(airquality)
```

```
[1] 44
```

`n_miss(airquality)` fonksiyonu, `naniar` paketine ait bir fonksiyondur ve `airquality` veri setindeki toplam eksik değer (NA) sayısını hesaplar. Çıktı olarak `[1] 44` değeri döner. Bu, `airquality` veri setinde toplam **44** adet eksik değer olduğunu gösterir.

7.1.3.2 `NA` olmayan (complete) değerlerin sayısı için:

```r
# naniar kütüphanesini yükleme
library(naniar)

# Tamamlanmış değer sayısını hesaplama
n_complete(airquality)
```

```
[1] 874
```

`n_complete(airquality)` fonksiyonu, `naniar` paketine ait bir fonksiyondur ve `airquality` veri setindeki *tamamlanmış* (yani eksik olmayan) toplam değer sayısını hesaplar. Çıktı olarak `[1] 874` değeri döner. Bu, `airquality` veri setinde toplam **874** adet tamamlanmış değer olduğunu gösterir.

7.1.3.3 `NA` olan değerlerin oranı için:

```r
# naniar kütüphanesini yükleme
library(naniar)

# Eksik değer oranını hesaplama
prop_miss(airquality)
```

```
[1] 0.04793028
```

`prop_miss(airquality)` fonksiyonunun çıktısı olan `[1] 0.04792626`, `airquality` veri setindeki verilerin yaklaşık **%4.79**'unun eksik olduğunu gösterir. Bu oran, eksik değer sayısının toplam veri noktası sayısına bölünmesiyle bulunur.

7.1.3.4 `NA` olmayan (complete) değerlerin oranı için:

```
# naniar kütüphanesini yükleme
library(naniar)

# Tamamlanmış değer oranını hesaplama
prop_complete(airquality)
```

```
[1] 0.9520697
```

`prop_complete(airquality)` fonksiyonu, `naniar` paketine ait bir fonksiyondur ve `airquality` veri setindeki *tamamlanmış* (yani eksik olmayan) değerlerin oranını hesaplar. Çıktı olarak `[1] 0.9520737` değeri döner. Bu, `airquality` veri setindeki değerlerin yaklaşık **%95.2**'sinin tamamlanmış olduğunu gösterir.

7.1.3.5 Eksik veriler için pareto tablosu `dlookr`

```
# dlookr kütüphanesini yükleme (gerekli fonksiyon için)
# install.packages("dlookr")
library(dlookr)

# Eksik verilerin Pareto grafiği ile gösterimi
plot_na_pareto(airquality,
               only_na = TRUE,
               # sadece eksik değer içeren değişkenlerin gösterilmesini sağlar.
               plot = FALSE)
```

```
# A tibble: 2 x 5
  variable frequencies  ratio grade cumulative
  <fct>          <int>  <dbl> <fct>      <dbl>
1 Ozone             37 0.242  Bad         84.1
2 Solar.R            7 0.0458 Good       100
```

```
               # grafik yerine sadece tablo çıktısının gösterilmesini sağlar.
```

7.1.4 Web Raporu Oluşturma

```r
# dlookr kütüphanesini yükleme
library(dlookr)

# Web raporu oluşturma
# diagnose_web_report(airquality, subtitle = "airquality")
```

`diagnose_web_report(airquality, subtitle = "airquality")` fonksiyonu, `dlookr` paketine ait bir fonksiyondur ve `airquality` veri seti için kapsamlı bir veri teşhis raporu oluşturur. Bu rapor bir HTML dosyası olarak kaydedilir ve bir web tarayıcısında görüntülenebilir.

7.1.5 `DLOOKR` Paketi ile Eksik Değerleri Doldurma

`dlookr` paketi ve `imputate_na()`

`dlookr` paketi, **veri teşhisi** (*data diagnosis*) ve **veri keşfi** (*data exploration*) için tasarlanmış bir R paketidir. Bu paket, veri kalitesini değerlendirmek, veri setini özetlemek, değişkenler arasındaki ilişkileri incelemek ve eksik verilerle başa çıkmak için çeşitli kullanışlı fonksiyonlar içerir. **`imputate_na()`** fonksiyonu da bu paketin eksik veri yönetimi araçlarından biridir.

`imputate_na()` fonksiyonunun temel amacı, bir veri setindeki eksik değerleri (NA) çeşitli yöntemlerle doldurmaktır. Bu fonksiyon, hem sayısal (numeric) hem de kategorik (categorical) değişkenlerdeki eksik değerleri ele alabilir ve farklı doldurma yöntemleri sunar.

- **Sayısal Değişkenler için Doldurma Yöntemleri:**
  - `"mean"`: Eksik değerleri değişkenin ortalamasıyla doldurur.
  - `"median"`: Eksik değerleri değişkenin medyanıyla (ortanca) doldurur.
  - `"mode"`: Eksik değerleri değişkenin moduyla (en sık tekrar eden değer) doldurur.
  - `"knn"`: K-en yakın komşu algoritmasını kullanarak eksik değerleri doldurur. Bu yöntem, eksik değerin bulunduğu satıra en yakın olan K tane gözlemi bulur ve bu gözlemlerin değerlerini kullanarak eksik değeri tahmin eder. Bu yöntem için bir referans değişken belirtmek gereklidir.
  - `"rpart"`: Özyinelemeli Bölümleme ve Regresyon Ağaçları (Recursive Partitioning and Regression Trees) yöntemini kullanarak eksik değerleri doldurur. Bu yöntem, bir karar ağacı modeli oluşturarak eksik değerleri tahmin eder. Bu yöntem için bir referans değişken belirtmek gereklidir.
  - `"mice"`: Zincirleme Denklemlerle Çoklu Atama (Multivariate Imputation by Chained Equations) yöntemini kullanarak eksik değerleri doldurur. Bu yöntem, her eksik değişken için bir model oluşturur ve diğer değişkenleri kullanarak eksik değerleri tahmin eder. Bu yöntem için bir referans değişken belirtmek ve bir rastgele sayı başlangıç değeri (random seed) ayarlamak gereklidir.

- **Kategorik Değişkenler için Doldurma Yöntemleri:**
  - `"mode"`: Eksik değerleri değişkenin moduyla (en sık tekrar eden kategori) doldurur.
  - `"rpart"`: Özyinelemeli Bölümleme ve Regresyon Ağaçları yöntemini kullanarak eksik değerleri doldurur. Bu yöntem için bir referans değişken belirtmek gereklidir.
  - `"mice"`: Zincirleme Denklemlerle Çoklu Atama yöntemini kullanarak eksik değerleri doldurur. Bu yöntem için bir referans değişken belirtmek ve bir rastgele sayı başlangıç değeri (random seed) ayarlamak gereklidir.

`imputate_na()` fonksiyonu, veri ön işleme adımlarında eksik verileri ele almak için kullanışlı bir araçtır. Doldurma yöntemini seçerken, verinizin yapısını ve analizin amacını göz önünde bulundurmanız önemlidir. Örneğin, ortalama ile doldurma, aykırı değerlerden etkilenebilirken, medyan ile doldurma bu etkiyi azaltır. `knn`, `rpart` ve `mice` gibi daha gelişmiş yöntemler ise, değişkenler arasındaki ilişkileri dikkate alarak daha doğru tahminler yapabilir.

7.1.5.1 Eksik değer içeren sütunu görüntüleme

```
data("airquality")

# airquality veri setinin Ozone sütununu görüntüleme
airquality$Ozone
```

```
  [1]  41  36  12  18  NA  28  23  19   8  NA   7  16  11  14  18  14  34   6
 [19]  30  11   1  11   4  32  NA  NA  NA  23  45 115  37  NA  NA  NA  NA  NA
 [37]  NA  29  NA  71  39  NA  NA  23  NA  NA  21  37  20  12  13  NA  NA  NA
 [55]  NA  NA  NA  NA  NA  NA  NA 135  49  32  NA  64  40  77  97  97  85  NA
 [73]  10  27  NA   7  48  35  61  79  63  16  NA  NA  80 108  20  52  82  50
 [91]  64  59  39   9  16  78  35  66 122  89 110  NA  NA  44  28  65  NA  22
[109]  59  23  31  44  21   9  NA  45 168  73  NA  76 118  84  85  96  78  73
[127]  91  47  32  20  23  21  24  44  21  28   9  13  46  18  13  24  16  13
[145]  23  36   7  14  30  NA  14  18  20
```

`airquality$Ozone` kodu, `airquality` adlı veri çerçevesinin `Ozone` adlı sütununu seçer ve bu sütundaki tüm değerleri bir vektör olarak döndürür.

Çıktıda görüldüğü gibi, `Ozone` sütunu sayısal değerler ve `NA` (Not Available - Mevcut Değil) değerleri içermektedir. `NA` değerleri, o gün için ozon ölçümünün yapılamadığını veya kaydedilmediğini gösterir.

> **Vektör Çıktılarında Köşeli Parantez**
>
> Çıktının başında ve sonunda `[1]`, `[28]`, `[55]` gibi ifadeler bulunur. Bunlar, çıktının hangi indeksinden itibaren yeni bir satıra geçildiğini gösterir. Örneğin, `[28]` ifadesi, o satırda 28. elemandan itibaren değerlerin listelendiğini belirtir. Bu, çıktının daha okunabilir olmasını sağlar, özellikle de çok uzun vektörler görüntülendiğinde.

7.1.5.2 Eksik değerleri ortalama (`mean`) ile doldurma

```
# dlookr kütüphanesini yükleme
library(dlookr)

# Ozone değişkenini Temp i referans alarak ortalama ile doldurma
aq_imp_ozone_mean <- imputate_na(airquality, Ozone, Temp, method = "mean")

# SADECE Ozone sütununu görüntüleme
aq_imp_ozone_mean
```

```
  [1]  41.00000  36.00000  12.00000  18.00000  42.12931  28.00000  23.00000
  [8]  19.00000   8.00000  42.12931   7.00000  16.00000  11.00000  14.00000
 [15]  18.00000  14.00000  34.00000   6.00000  30.00000  11.00000   1.00000
 [22]  11.00000   4.00000  32.00000  42.12931  42.12931  42.12931  23.00000
 [29]  45.00000 115.00000  37.00000  42.12931  42.12931  42.12931  42.12931
 [36]  42.12931  42.12931  29.00000  42.12931  71.00000  39.00000  42.12931
 [43]  42.12931  23.00000  42.12931  42.12931  21.00000  37.00000  20.00000
 [50]  12.00000  13.00000  42.12931  42.12931  42.12931  42.12931  42.12931
 [57]  42.12931  42.12931  42.12931  42.12931  42.12931 135.00000  49.00000
 [64]  32.00000  42.12931  64.00000  40.00000  77.00000  97.00000  97.00000
 [71]  85.00000  42.12931  10.00000  27.00000  42.12931   7.00000  48.00000
 [78]  35.00000  61.00000  79.00000  63.00000  16.00000  42.12931  42.12931
 [85]  80.00000 108.00000  20.00000  52.00000  82.00000  50.00000  64.00000
 [92]  59.00000  39.00000   9.00000  16.00000  78.00000  35.00000  66.00000
 [99] 122.00000  89.00000 110.00000  42.12931  42.12931  44.00000  28.00000
[106]  65.00000  42.12931  22.00000  59.00000  23.00000  31.00000  44.00000
[113]  21.00000   9.00000  42.12931  45.00000 168.00000  73.00000  42.12931
[120]  76.00000 118.00000  84.00000  85.00000  96.00000  78.00000  73.00000
[127]  91.00000  47.00000  32.00000  20.00000  23.00000  21.00000  24.00000
[134]  44.00000  21.00000  28.00000   9.00000  13.00000  46.00000  18.00000
[141]  13.00000  24.00000  16.00000  13.00000  23.00000  36.00000   7.00000
[148]  14.00000  30.00000  42.12931  14.00000  18.00000  20.00000
```

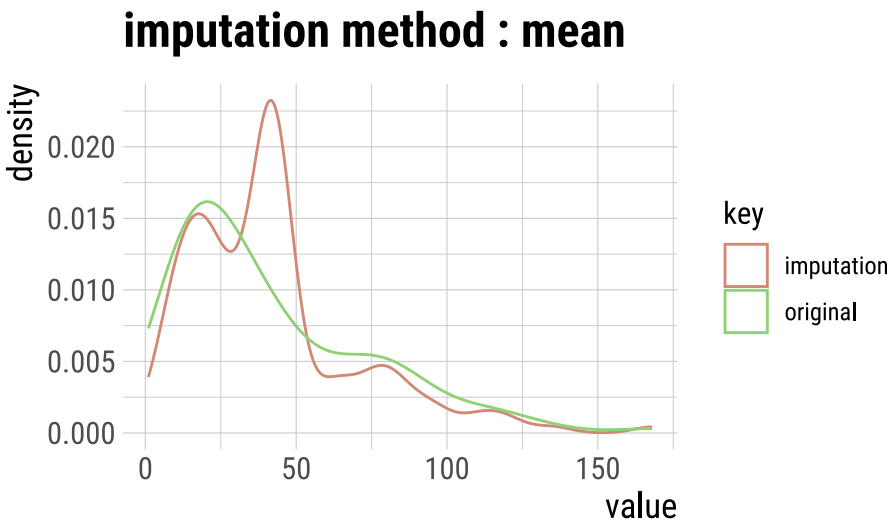
```
attr(,"var_type")
[1] "numerical"
attr(,"method")
[1] "mean"
attr(,"na_pos")
 [1]   5  10  25  26  27  32  33  34  35  36  37  39  42  43  45  46  52  53  54
[20]  55  56  57  58  59  60  61  65  72  75  83  84 102 103 107 115 119 150
attr(,"type")
[1] "missing values"
attr(,"message")
[1] "complete imputation"
attr(,"success")
[1] TRUE
attr(,"class")
[1] "imputation" "numeric"
```

- `data`: İşlem yapılacak veri seti. Burada `airquality` veri seti kullanılmıştır.
- `target`: Eksik değerlerin doldurulacağı sütun. Burada `Ozone` sütunu belirtilmiştir.
- `ref`: Referans alınacak sütun. Burada `Temp` sütunu kullanılmıştır.
- `method`: Doldurma yöntemi. `"mean"` ile ortalama kullanılarak doldurma yapılır.

Çalışma Prensibi: Eksik değerler `Temp` sütununun ortalamasına göre doldurulmuş ve yeni bir veri seti (`aq_imp_ozone`) oluşturulmuştur.

7.1.5.3 Ortalama (`mean`) ile doldurma öncesi ve sonrası yoğunluk dağılımları

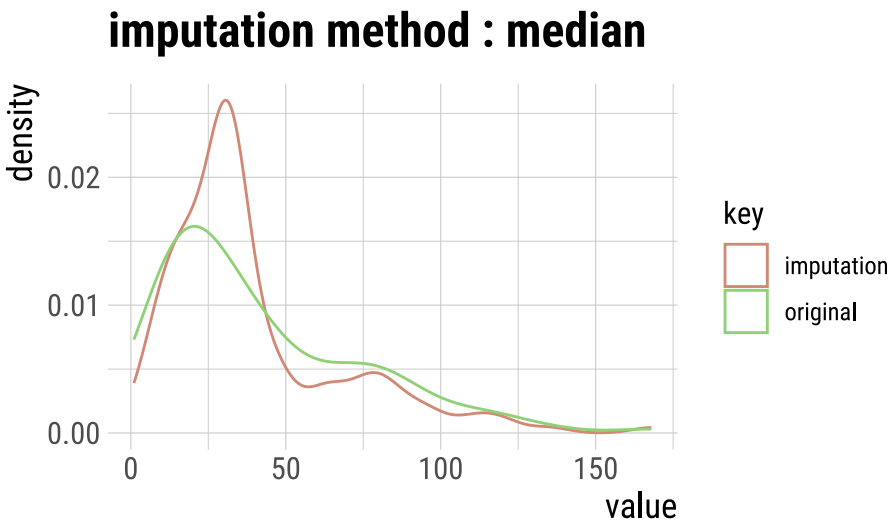
```
plot(aq_imp_ozone_mean)
```



Eksik değerlerin ortalama ile doldurulması, veri setinin genel dağılımında hafif değişikliklere yol açmıştır. Eksik değerlerin doldurulması, yoğunluk eğrisini daha düzgün hale getirmiştir, ancak bu işlem, verilerin doğal dağılımını biraz değiştirebilir. Özellikle veri çok eksikse, ortalama ile doldurma yöntemi dağılımın şeklini etkileyebilir. Eğer veri setinin doğal varyasyonunu korumak çok önemliyse, alternatif doldurma yöntemleri (örneğin, knn veya regresyon tabanlı yöntemler) düşünülebilir.

7.1.5.4 Medyan (median) ile doldurma öncesi ve sonrası yoğunluk dağılımları

```
library(dlookr)

# Ozone ve Temp değişkenlerini medyan ile doldurma
aq_imp_ozone_median <- imputate_na(airquality, Ozone, Temp, method = "median")

# plot() fonksiyonu ile çizim (indekse karşı değer grafiği)
plot(aq_imp_ozone_median)
```



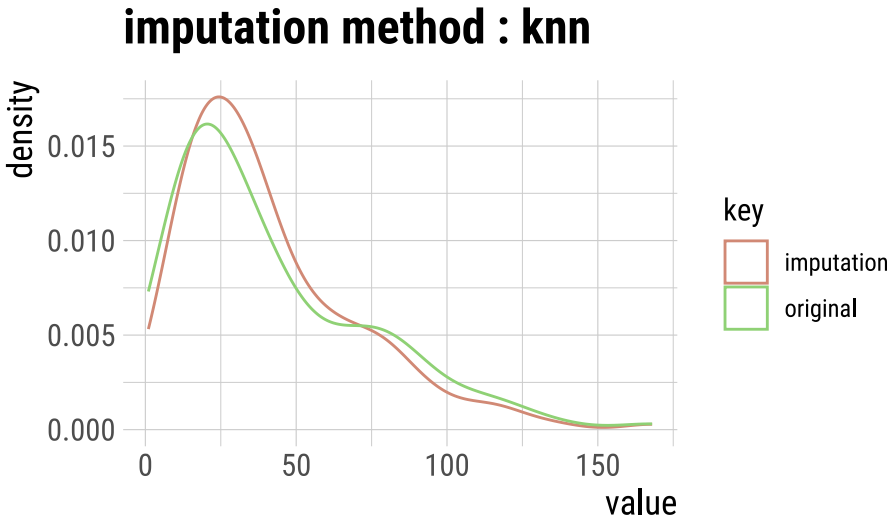
Grafikte, x ekseni vektördeki elemanların sırasını (indeks), y ekseni ise medyan ile doldurulmuş `Ozone` değerlerini gösterir. Ortalama ile doldurmaya benzer şekilde, grafikte noktaların rastgele yukarı aşağı hareket ettiğini görürsünüz. Ancak, medyan ile doldurmada, ortalama ile doldurmaya kıyasla grafikte daha az yatay çizgi veya düz bölge görürsünüz. Bunun nedeni, medyanın ortalamadan farklı değerlere sahip olabilmesi ve aynı değerin daha az tekrar etmesidir.

> Ortalama vs. Medyan ile Eksik Değer Doldurma
>
> Ortalama ile doldurma, dağılımın ortasında bir yığılmaya neden olurken, medyan ile doldurma bu yığılmayı daha az belirgin hale getirir. Çünkü medyan, aykırı değerlerden ortalamaya göre daha az etkilenir. Bu nedenle, verilerinizde aykırı değerler varsa, medyan ile doldurma ortalama ile doldurmaya göre daha iyi bir seçenek olabilir.

7.1.5.5 `knn` ile doldurma öncesi ve sonrası yoğunluk dağılımları

```
library(dlookr)

# Ozone değişkenini knn ile doldurma (Temp'i referans değişken olarak kullanır)
aq_imp_ozone_knn <- imputate_na(airquality, Ozone, Temp, method = "knn")
```

```r
# plot() fonksiyonu ile çizim (indekse karşı değer grafiği)
plot(aq_imp_ozone_knn)
```

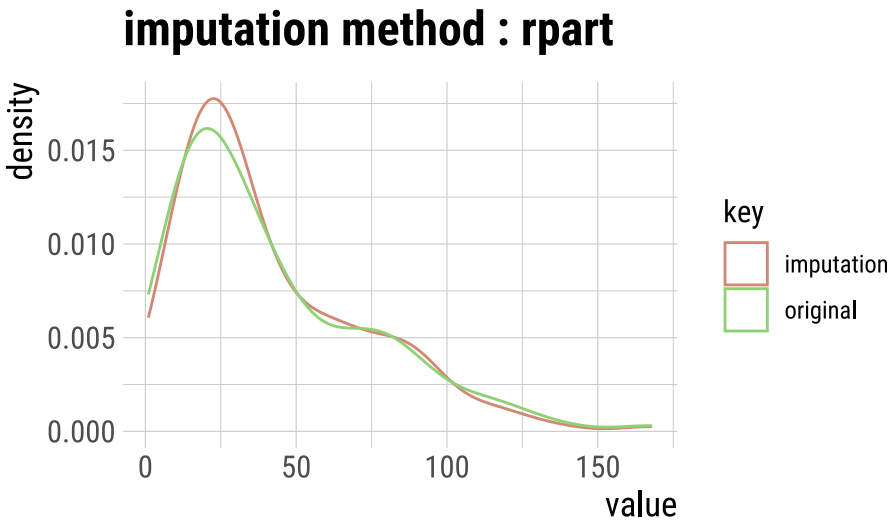


Yukarıdaki kod, `Ozone` değişkenindeki eksik değerleri ***knn*** **(k-Nearest Neighbors - k-En Yakın Komşu)** yöntemiyle dolduruyor ve ardından bu doldurulmuş değerleri `plot()` fonksiyonu ile çiziliyor.

- **Referans Değişkenin Önemi:** knn ile doldurma yaparken, seçilen referans değişkenin (burada `Temp`) eksik değerlere sahip olmaması veya çok az eksik değere sahip olması önemlidir. Aksi takdirde, modelin doğruluğu düşebilir.
- **Benzer Gözlemler:** knn, eksik değere sahip olan gözleme en benzer k tane gözlemi bulur ve bu gözlemlerin değerlerini kullanarak eksik değeri tahmin eder. Bu nedenle, verideki yerel örüntüleri yakalamada etkilidir.
- **k Değeri:** k parametresi (komşu sayısı) önemlidir. Çok küçük bir k değeri, aşırı uyuma (overfitting) neden olabilirken, çok büyük bir k değeri, yerel örüntüleri kaçırmaya neden olabilir. `imputate_na()` fonksiyonunda k değeri varsayılan olarak 5'tir, ancak gerekirse değiştirilebilir.
- **Dağılımın Değişimi:** knn ile doldurma, ortalama veya medyan ile doldurmaya göre dağılımı daha az etkiler. Çünkü bu yöntem, eksik değerleri tek bir sabit değerle doldurmak yerine, benzer gözlemlerin değerlerine göre farklı değerlerle doldurur.

7.1.5.6 `rpart` ile doldurma öncesi ve sonrası yoğunluk dağılımları

```
library(dlookr)

# Ozone değişkenini rpart ile doldurma (Temp'i referans değişken olarak kullanır)
aq_imp_ozone_rpart <- imputate_na(airquality, Ozone, Temp, method = "rpart")

# plot() fonksiyonu ile çizim (indekse karşı değer grafiği)
plot(aq_imp_ozone_rpart)
```

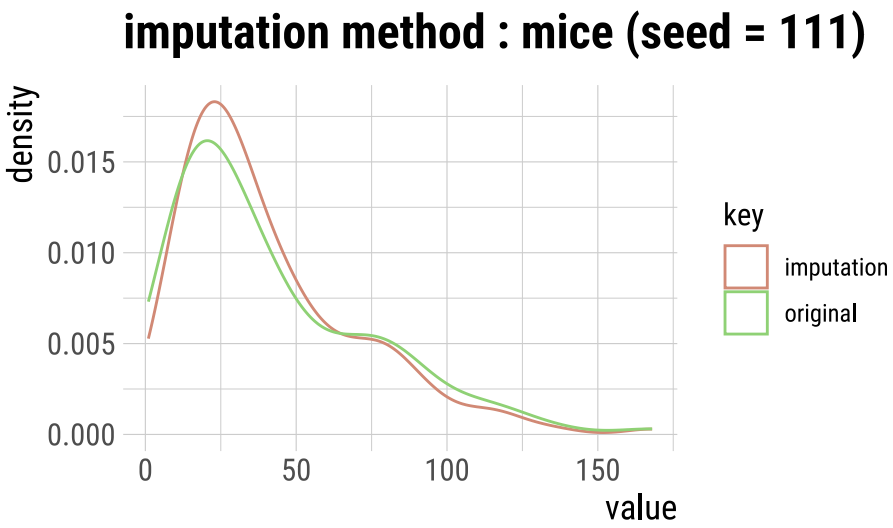


Yukarıdaki kod ile `Ozone` değişkenindeki eksik değerleri ***rpart*** **(Recursive Partitioning and Regression Trees - Özyinelemeli Bölümleme ve Regresyon Ağaçları)** yöntemiyle dolduruyor ve ardından bu doldurulmuş değerleri `plot()` fonksiyonu ile çiziyor.

- **referans Değişkenin Önemi:** rpart ile doldurma yaparken, seçilen referans değişkenin (burada `Temp`) eksik değerlere sahip olmaması veya çok az eksik değere sahip olması önemlidir. Aksi takdirde, modelin doğruluğu düşebilir.
- **Doğrusal Olmayan İlişkiler:** rpart, değişkenler arasındaki doğrusal olmayan ilişkileri de yakalayabildiği için, ortalama veya medyan ile doldurmaya göre daha doğru sonuçlar verebilir. Ancak, aşırı uyum (overfitting) riskini de beraberinde getirebilir.
- **Dağılımın Değişimi:** rpart ile doldurma, ortalama veya medyan ile doldurmaya göre dağılımı daha az etkiler. Çünkü bu yöntem, eksik değerleri tek bir sabit değerle doldurmak yerine, referans değişkene göre farklı değerlerle doldurur.

7.1.5.7 `mice` ile doldurma öncesi ve sonrası yoğunluk dağılımları

```
library(dlookr)
library(mice)

# Ozone değişkenini mice ile doldurma (Temp'i ve diğer değişkenleri kullanır)
aq_imp_ozone_mice <- imputate_na(airquality, Ozone, Temp,
                                 method = "mice",
                                 seed = 111,
                                 print =FALSE)

# plot() fonksiyonu ile çizim (indekse karşı değer grafiği)
plot(aq_imp_ozone_mice)
```



Yukarıdaki kod ile `Ozone` değişkenindeki eksik değerleri *mice* (Multivariate Imputation by Chained Equations - Zincirleme Denklemlerle Çoklu Atama) yöntemiyle dolduruyor ve ardından bu doldurulmuş değerleri `plot()` fonksiyonu ile çiziyor.

- **Çoklu Atama:** `mice`, eksik değerler için birden fazla olası değer ürettiği için, eksik verilerin belirsizliğini daha iyi yansıtır. Bu, daha doğru ve güvenilir sonuçlar elde etmenizi sağlar.

- **Değişkenler Arası İlişkiler:** `mice`, değişkenler arasındaki ilişkileri dikkate aldığı için, diğer yöntemlere göre daha iyi tahminler yapabilir.
- **Dağılımın Korunması:** `mice`, verinin orijinal dağılımını daha iyi korur. Ortalama veya medyan ile doldurma, dağılımda bozulmalara neden olabilirken, `mice` bu etkiyi en aza indirir.

7.1.5.8 Doldurulmuş veriyi orijinal veriye entegre etme - Aşama 1

```
# Gerekli kütüphanelerin yüklenmesi
library(dlookr)
library(tidyverse)
library(mice)

# Orijinal veri setini kopyalama
airquality_imp <- airquality

# Doldurulmuş Ozone verisini orijinal veri setine atama
airquality_imp$Ozone <- aq_imp_ozone_mice

# Doldurulmuş veri setini görüntüleme
head(airquality_imp, 10)
```

```
   Ozone Solar.R Wind Temp Month Day
1   41.0     190  7.4   67     5   1
2   36.0     118  8.0   72     5   2
3   12.0     149 12.6   74     5   3
4   18.0     313 11.5   62     5   4
5   21.0      NA 14.3   56     5   5
6   28.0      NA 14.9   66     5   6
7   23.0     299  8.6   65     5   7
8   19.0      99 13.8   59     5   8
9    8.0      19 20.1   61     5   9
10  40.2     194  8.6   69     5  10
```

7.1.5.9 Doldurulmuş veriyi orijinal veriye entegre etme - Aşama 2

```
# Gerekli kütüphanelerin yüklenmesi
library(dlookr)
```

```r
library(mice)
library(tidyverse)

# Ozone değişkenini ortalama ile doldurma
aq_imp_solar_mice <- imputate_na(airquality_imp, Solar.R, Temp,
                                 method = "mice",
                                 seed = 111,
                                 print = FALSE)
# "print =" argümanı eğer TRUE olarak ayarlanırsa, mice işlemin geçmişini konsolda
# yazdıracaktır. Sessiz bir hesaplama için print=FALSE kullanın.

# Doldurulmuş Ozone verisini orijinal veri setine atama
airquality_imp$Solar.R <- aq_imp_solar_mice

# Doldurulmuş veri setini görüntüleme
head(airquality_imp, 10)
```

```
   Ozone Solar.R Wind Temp Month Day
1   41.0   190.0  7.4   67     5   1
2   36.0   118.0  8.0   72     5   2
3   12.0   149.0 12.6   74     5   3
4   18.0   313.0 11.5   62     5   4
5   21.0   266.2 14.3   56     5   5
6   28.0   218.0 14.9   66     5   6
7   23.0   299.0  8.6   65     5   7
8   19.0    99.0 13.8   59     5   8
9    8.0    19.0 20.1   61     5   9
10  40.2   194.0  8.6   69     5  10
```

Orijinal veri seti ile karşılaştırma

```r
head(airquality, 10)
```

```
   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
```

7.1.6 `MICE` Paketi ile Eksik Değerleri Doldurma

MICE paketi, eksik veri problemini çözmek için kullanılan bir araçtır ve eksik verileri çoklu imputasyon yöntemini kullanarak işleme alır. Süreç, eksik veri içeren bir veri setiyle başlar. Bu veri genellikle bir data frame formatındadır ve eksik verilerin, diğer değişkenlerle olan ilişkilerine dayanarak doldurulması hedeflenir.

MICE paketi, eksik veri problemini çözmek için şu adımları takip eder:

1. **Eksik verileri birden fazla doldurur (`mice()`).**

   İlk adımda, `mice()` fonksiyonu kullanılarak eksik veriler birden fazla iterasyonla doldurulur. Her iterasyonda eksik olan değişkenler, diğer değişkenlerle olan ilişkileri kullanılarak tahmin edilir. Bu işlem sonucunda, doldurulmuş veri setlerini içeren bir "mids" nesnesi oluşturulur.

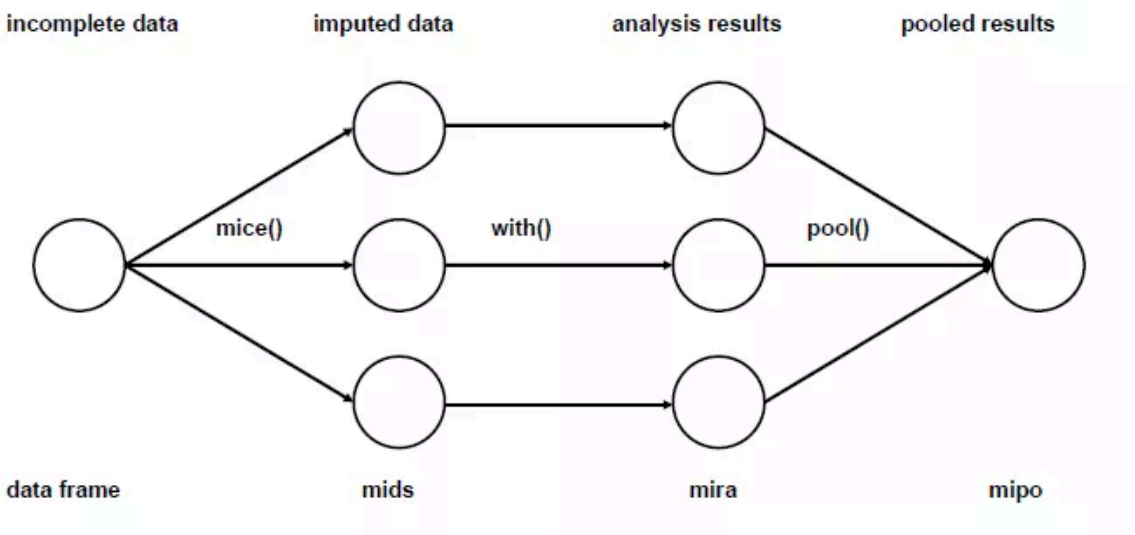
2. **Doldurulan veri setleri üzerinde analizler yapar (`with()`).**

   Daha sonra, `with()` fonksiyonu aracılığıyla doldurulan veri setleri üzerinde istatistiksel analizler gerçekleştirilir. Örneğin, her doldurulmuş veri seti için regresyon analizi gibi istatistiksel işlemler yapılabilir ve bu analizlerin sonuçları "mira" nesnesi olarak saklanır.

3. **Analiz sonuçlarını havuzlar ve birleştirir (`pool()`).**

   Son aşamada, `pool()` fonksiyonu kullanılarak her bir doldurulmuş veri seti üzerinde yapılan analizlerin sonuçları birleştirilir. Bu birleştirme işlemi, eksik veri kaynaklı belirsizliği hesaba katarak daha güvenilir ve tutarlı sonuçlar elde etmeyi sağlar. Bu süreç sonucunda, analizlerin nihai sonuçları "mipo" nesnesi olarak elde edilir.

MICE paketi, eksik veri problemini istatistiksel olarak en iyi şekilde ele alarak analizlerin güvenilirliğini artırmayı hedefler ve eksik veriden kaynaklanan yanlılığı azaltır.



Figür: https://www.jstatsoft.org/article/view/v045i03

Veri Seti `nhanes`

```
# Gerekli kütüphanelerin yüklenmesi
# install.packages("mice")
# install.packages("tidyverse")
# install.packages("NHANES")

library(mice)
library(tidyverse)
library(NHANES)

nhanes3 <- NHANES %>%
   select(Weight, Height, TotChol, PhysActive)

# NHANES veri setinin görüntülenmesi (örnek olarak ilk 10 satır)
head(nhanes3, 10)
```

```
# A tibble: 10 x 4
   Weight Height TotChol PhysActive
    <dbl>  <dbl>   <dbl> <fct>
 1   87.4   165.    3.49 No
 2   87.4   165.    3.49 No
 3   87.4   165.    3.49 No
 4   17     105.   NA    <NA>
```

```
 5   86.7   168.    6.7  No
 6   29.8   133.    4.86 <NA>
 7   35.2   131.    4.09 <NA>
 8   75.7   167.    5.82 Yes
 9   75.7   167.    5.82 Yes
10   75.7   167.    5.82 Yes
```

`nhanes` Veri Seti

NHANES (Ulusal Sağlık ve Beslenme İnceleme Anketi), ABD'de yetişkinlerin ve çocukların sağlık ve beslenme durumunu ölçen bir CDC araştırmasıdır. Anketler ve fiziksel muayeneler içerir. 76 farklı değişkeni bulunmaktadır. Araştırmamızda özellikle aşağıda yer alan

Çalışmamızda aşağıda yer alan değişkenlere odaklanılacaktır:

- **Weight (Kilo):** Obezite ve aşırı kiloyu değerlendirmek için ölçülür.
- **Height (Boy):** VKİ (Vücut Kitle İndeksi) hesaplamak için kullanılır.
- **TotChol (Toplam Kolesterol):** Kalp hastalığı riskini gösterir.
- **PhysActive (Fiziksel Aktivite):** Genel sağlık için önemlidir.

Bu veriler, halk sağlığı sorunlarını anlamak ve sağlık politikalarını değerlendirmek için kullanılır.

`nhanes` Veri Setinde Eksik Değerler Özet Tablosu

```r
library(naniar)

# miss_var_summary() fonksiyonunu uygulama
miss_var_summary(nhanes3)
```

```
# A tibble: 4 x 3
  variable   n_miss pct_miss
  <chr>       <int>    <num>
1 PhysActive   1674    16.7
2 TotChol      1526    15.3
3 Height        353     3.53
4 Weight         78     0.78
```

- **`PhysActive 167 16.7:`** `PhysActive` değişkeninde 167 eksik veri vardır ve bu, toplam verinin %16.7'sine karşılık gelir.
- **`TotChol 152 15.3:`** `TotChol` değişkeninde 152 eksik veri vardır ve bu, toplam verinin %15.3'üne karşılık gelir.

- **`Height 35 3.53`:** `Height` değişkeninde 35 eksik veri vardır ve bu, toplam verinin %3.53'üne karşılık gelir.
- **`Weight 7 0.78`:** `Weight` değişkeninde 7 eksik veri vardır ve bu, toplam verinin %0.78'ine karşılık gelir.

7.1.6.1 MICE (Çoklu İmputasyon) ile Eksik Veri Doldurma

```
library(mice)

# nhanes veri setinde eksik değerleri doldurma (20 imputasyon seti oluşturma)
nhanes_multiimp <- mice(nhanes3, m = 20, print = FALSE)
```

Bu kod, `mice` paketi kullanılarak **nhanes** veri setindeki eksik değerlerin doldurulması için çoklu imputasyon işlemi gerçekleştirir. Burada, `m = 20` parametresiyle eksik değerlerin 20 farklı tahmini yapılır ve her biri bir imputasyon veri seti olarak oluşturulur.

- `nhanes`: İçerisinde eksik değerler bulunan örnek bir veri seti.
- `m = 20`: Çoklu imputasyon işlemiyle 20 farklı doldurulmuş veri seti oluşturulacağını belirtir.

Yukarıdaki kod, `mice` paketini kullanarak `nhanes` veri setindeki eksik değerleri çoklu atama yöntemiyle doldurur. Bu yöntem, eksik verilerin belirsizliğini hesaba katarak daha doğru ve güvenilir analizler yapmanıza olanak tanır. Kodun doğru çalışması için `data(nhanes)` satırının eklenmesi önemlidir. Ayrıca, `seed` eklenmesi, sonuçların tekrarlanabilirliğini sağlar. Kod, `nhanes_multiimp` adında bir `mids` nesnesi oluşturur.

> **i** Çoklu Atama ve mice'ın Avantajları
>
> - **Çoklu Atama:** `mice`, eksik değerler için tek bir değer yerine birden fazla olası değer ürettiği için, eksik verilerin belirsizliğini daha iyi yansıtır. Bu, standart tek atama yöntemlerine (ortalama, medyan vb.) göre daha doğru ve güvenilir sonuçlar elde etmenizi sağlar. Tek bir değer atamak yerine, olası değerlerin bir dağılımını kullanarak, eksik veriden kaynaklanan belirsizliği modelinize dahil edersiniz.
> - **Değişkenler Arası İlişkiler:** `mice`, atama işlemi sırasında değişkenler arasındaki ilişkileri dikkate alır. Bu, eksik verilerin daha gerçekçi ve tutarlı bir şekilde tahmin edilmesini sağlar. Örneğin, yaş ve VKİ arasındaki ilişkiyi göz önünde bulundurarak, eksik VKİ değerlerini daha doğru bir şekilde tahmin edebilir.
> - **Dağılımın Korunması:** `mice`, verinin orijinal dağılımını daha iyi korur. Ortalama veya medyan ile doldurma gibi basit yöntemler, veri dağılımında bozulmalara neden olabilirken, `mice` bu etkiyi en aza indirir. Bu, analizlerinizin daha güvenilir ve anlamlı olmasını sağlar.

7.1.6.2 MICE ile Veri Setleri Üzerinde Lineer Regresyon Modeli Kurma

```
library(mice)

# Her bir atanmış veri setine lineer regresyon modeli uygula
lm_multiimp <- with(nhanes_multiimp, lm(Weight ~ Height + TotChol + PhysActive))
```

Bu kod, daha önce oluşturulan `nhanes_multiimp` adlı `mids` (multiple imputation data set) nesnesini kullanarak, her bir tamamlanmış veri setine bir lineer regresyon modeli uygular.

- **`with(nhanes_multiimp, ...)`:** Bu fonksiyon, `nhanes_multiimp` nesnesindeki *her bir* tamamlanmış veri seti üzerinde belirtilen ifadeyi uygular. Yani, 20 farklı tamamlanmış veri setin varsa, bu ifade 20 kez çalıştırılır ve her biri için ayrı bir lineer regresyon modeli oluşturulur.
- **`lm(Weight ~ Height + TotChol + PhysActive)`:** Bu, lineer regresyon modelini tanımlar. `Weight` (Kilo) bağımlı değişken, `Height` (Boy), `TotChol` (Toplam Kolesterol) ve `PhysActive` (Fiziksel Aktivite) ise bağımsız değişkenlerdir. Yani, kilonun boy, toplam kolesterol ve fiziksel aktivite ile nasıl ilişkili olduğunu inceliyoruz.

> **i** Faktör Dönüşümü
>
> **PhysActive'in Faktöre Dönüştürülmesi**: Eğer PhysActive değişkeni sayısal olarak kodlanmış bir kategorik değişken ise (örneğin, 1=Aktif, 2=Pasif ya da yes, no gibi), lineer regresyon modelinde doğru şekilde yorumlanabilmesi için bu değişkeni `factor()` fonksiyonu ile faktöre dönüştürmek çok önemlidir. Kodu bu duruma göre güncelledim. Eğer `PhysActive` zaten bir faktör ise bu satıra gerek yoktur.

7.1.6.3 MICE ile Regresyon Sonuçlarını Havuzlama

```
library(mice)

# Çoklu imputasyon veri setleri üzerindeki regresyon sonuçlarını havuzlama
lm_pooled <- pool(lm_multiimp)

# Havuzlanmış sonuçları özetleme
summary(lm_pooled)
```

```
           term    estimate  std.error  statistic         df      p.value
1   (Intercept) -93.2761794 1.88249998 -49.549100   85.53417 9.001161e-65
2        Height   0.9997873 0.01088186  91.876490   92.25780 1.680034e-92
3       TotChol   1.5587611 0.17765159   8.774259 2354.76828 3.236133e-18
4 PhysActiveYes  -5.9132808 0.38186334 -15.485332 1660.16326 1.265237e-50
```

Bu kod, `lm_multiimp` nesnesindeki çoklu imputasyon veri setleri üzerinde oluşturulan lineer regresyon modellerinin sonuçlarını birleştirir (pool). Havuzlama işlemi, eksik veriler nedeniyle ortaya çıkan belirsizliği hesaba katar ve tüm imputasyon veri setlerinden elde edilen sonuçları birleştirerek daha doğru ve güvenilir tahminler sunar.

Bu model, bağımlı değişken olan **Weight (Ağırlık)** üzerinde **Height (Boy Uzunluğu)**, **TotChol (Toplam Kolesterol)** ve **PhysActive (Fiziksel Aktivite Durumu)** değişkenlerinin etkilerini anlamlı bir şekilde açıklamaktadır. **Tüm değişkenlerin p-değerleri oldukça küçüktür ve bu değişkenlerin modelde anlamlı bir etkisi olduğunu göstermektedir.** Modeldeki katsayılar, bağımlı değişken üzerinde her bir bağımsız değişkenin etkisini istatistiksel olarak güçlü bir şekilde temsil etmektedir.

7.1.6.4 MICE ile Tamamlanmış Veri Seti

```r
library(mice)

# İlk doldurulmuş veri setini elde etme
nhanes3_completed <- complete(nhanes_multiimp)

# Doldurulmuş veri setini görüntüleme
head(nhanes3_completed)
```

```
  Weight Height TotChol PhysActive
1   87.4  164.7    3.49         No
2   87.4  164.7    3.49         No
3   87.4  164.7    3.49         No
4   17.0  105.4    3.80         No
5   86.7  168.4    6.70         No
6   29.8  133.1    4.86        Yes
```

7.1.6.5 Ham Veri ile Son Veriyi Karşılaştırma

```r
summary(nhanes3_completed)
```

```
     Weight           Height         TotChol       PhysActive
 Min.   :  2.80   Min.   : 83.6   Min.   : 1.530   No :4710
 1st Qu.: 56.10   1st Qu.:155.7   1st Qu.: 4.030   Yes:5290
 Median : 72.70   Median :165.5   Median : 4.680
 Mean   : 70.99   Mean   :159.9   Mean   : 4.813
 3rd Qu.: 88.90   3rd Qu.:174.3   3rd Qu.: 5.480
 Max.   :230.70   Max.   :200.4   Max.   :13.650
```

```r
summary(nhanes3)
```

```
     Weight           Height         TotChol       PhysActive
 Min.   :  2.80   Min.   : 83.6   Min.   : 1.530   No  :3677
 1st Qu.: 56.10   1st Qu.:156.8   1st Qu.: 4.110   Yes :4649
 Median : 72.70   Median :166.0   Median : 4.780   NA's:1674
 Mean   : 70.98   Mean   :161.9   Mean   : 4.879
 3rd Qu.: 88.90   3rd Qu.:174.5   3rd Qu.: 5.530
 Max.   :230.70   Max.   :200.4   Max.   :13.650
 NA's   :78       NA's   :353     NA's   :1526
```

7.2 Aykırı Değerler ile Çalışma

7.2.1 Aykırı Değerlerin Tespiti

- Kutu grafikleri (`ggplot2::geom_boxplot()`).
- Z-skore veya IQR yöntemleri (`scale()`, `boxplot.stats()`).

7.2.2 Aykırı Değerlerin Çıkarılması veya Düzenlenmesi

- Veri dönüşümleri ile etkilerini azaltma (ör. `log()`, `sqrt()`).
- Kategorik aykırılıkların analizi (`dplyr::filter()`).

7.3 Veri Dönüşümleri ve Standardizasyon

7.3.1 Matematiksel Dönüşümler

- Logaritmik (`log()`), karekök (`sqrt()`) dönüşümleri

7.3.2 Standardizasyon ve Normalizasyon

- `Z-skore standardizasyonu (`scale()`).`
- `Min-max normalizasyon (`scales::rescale()`, `caret::preProcess()`).`

7.4 Veri Doğrulama ve Temizleme İşlemleri

7.4.1 Veri Türlerinin Düzenlenmesi

7.4.1.1 Kategorik Değişkenler

- Kategorilere dönüştürme (`as.factor()`, `forcats::fct_reorder()`).
- Seyrek kategorilerin birleştirilmesi (`forcats::fct_lump()`).

7.4.1.2 Tarih ve Saat Verileri

- Tarih dönüşümleri (`lubridate::ymd()`, `dmy()`).

7.4.1.3 Veri Çoğaltmalarının Kaldırılması

- Benzersiz veri seçimi (`dplyr::distinct()`).

Referanslar

https://naniar.njtierney.com/

https://www.rdocumentation.org/packages/mice/versions/3.17.0/topics/mice

https://choonghyunryu.github.io/dlookr/ https://rpubs.com/chibueze99/MissingR

https://stefvanbuuren.name/fimd/

https://rmisstastic.netlify.app/tutorials/josse_bookdown_dataanalysismissingr_2020

https://rpubs.com/rpatel40/handling_missing_data_in_R

https://www.youtube.com/watch?v=Akb401i32Oc&ab_channel=yuzaRDataScience

Referanslar