R ile Veri Dönüşümü

Servet Çetin

İçindekiler

Önsöz

Ekonomik ve sosyal kalkınmayı teşvik etmek amacıyla uygulanan devlet yardımları, kamu kaynaklarının etkin ve verimli kullanımını sağlamayı hedefleyen kritik politika araçlarıdır. Ancak bu yardımların gerçekten beklenen etkileri yaratıp yaratmadığını belirlemek, yalnızca sağlanan desteklerin miktarını değerlendirmekle değil, bu desteklerin ekonomi ve toplum üzerindeki somut sonuçlarını bilimsel yöntemlerle ölçmekle mümkündür. Etki değerlendirme yöntemleri, bu doğrultuda karar alıcıların bilinçli ve veri temelli politikalar geliştirmesine olanak sağlayan güçlü bir enstrüman olarak öne çıkmaktadır.

Bu çalışma, devlet yardımlarının etki değerlendirme süreçlerinde kullanılan yöntemlere teorik ve uygulamalı bir temel sunmayı amaçlamaktadır. Etki değerlendirme süreçlerinin güvenilir ve tutarlı sonuçlar üretebilmesi için iki temel unsur büyük önem taşımaktadır:

- Kullanılan analiz yöntemlerinin bilimsel sağlamlığı
- Veri setlerinin doğruluğu, eksiksizliği ve tutarlılığı

Bu bağlamda, etki değerlendirme süreci yalnızca istatistiksel analizleri içermekle kalmayıp, aynı zamanda veri manipülasyonu, veri temizleme, eksik veri yönetimi, aykırı değer analizi ve veri dönüşümleri gibi konular da süreç içinde kritik rol oynamaktadır. Eksik veya yanlış verilerle yürütülen bir analiz, politika yapıcıları yanlış yönlendirebilmekte ve kamu kaynaklarının verimsiz kullanımına neden olabilmektedir.

Son yıllarda, etki değerlendirme alanındaki en önemli gelişmelerden biri, istatistiksel analizlerin programlama dilleri ve dinamik raporlama sistemleriyle entegre edilmesi olmuştur. Bu çerçevede, çalışmada R programlama dili ve Quarto platformu kullanılarak uygulamalı bir rehber oluşturulmuştur. R, sunduğu esneklik, geniş analiz olanakları ve güçlü veri işleme yetenekleriyle etki değerlendirme süreçlerinde yaygın olarak kullanılan bir araç haline gelmiştir. Çalışma, R içerisinde yer alan bir platform olan Quarto'nun kitap formatı özelliğiyle derlenmiş olup, hem okunması kolay bir kaynak hem de uygulamalı bir rehber olarak tasarlanmıştır.

Bu çalışma, Başkanlığımız ve diğer kamu kurumlarında yürütülen etki değerlendirme çalışmalarına destek sağlamayı hedeflemektedir. Etki değerlendirme sürecinde veri yönetimi gibi temel zorluklara yönelik çözüm önerileri sunarak, veri temelli karar alma süreçlerine katkı sağlamayı amaçlamaktadır.

Günümüzde, etki değerlendirme yalnızca uygulanan politikaların başarısını ölçmek için değil, gelecekte uygulanacak politikaların daha iyi tasarlanması için de kritik bir rol oynamaktadır. Bu çalışma ile devlet yardımlarının etkisini etkisini değerlendirmeden önce veri yönetimi ile alakalı çalışmalara katkı sunmak hedeflemektedir.

Strateji ve Bütçe Başkanlığı

Devlet Yardımları Genel Müdürlüğü

1 Giriş

1.1 Etki Değerlendirmede Verinin Önemi

Etki değerlendirmesi, belirli bir politika, program veya müdahalenin hedeflenen sonuçlara ulaşıp ulaşmadığını belirlemek amacıyla kullanılan sistematik bir analiz sürecidir. Kamu politikalarından sosyal programlara, ekonomik teşviklerden eğitim reformlarına kadar geniş bir yelpazede kullanılan etki değerlendirmesi, uygulanan müdahalelerin etkisini bilimsel yöntemlerle ölçerek politika yapıcıların bilinçli kararlar almasına olanak tanır.

Bu sürecin temel taşı veridir. Doğru, güvenilir ve kapsamlı veri olmadan, yapılan değerlendirmelerin bilimsel temeli zayıf kalır ve yanlış yönlendirmelere yol açabilir. Verinin toplanması, temizlenmesi, analiz edilmesi ve yorumlanması, etki değerlendirme süreçlerinin başarılı bir şekilde yürütülmesini sağlar.

Örneğin, bir devlet destek programının istihdam üzerindeki etkisini ölçmek için destek alan ve almayan firmaların karşılaştırılması gerekir. Ancak, bu firmalara ait eksik veya yanlış veriler kullanılırsa, programın gerçek etkisi olduğundan farklı yorumlanabilir ve politika yapıcılar yanlış çıkarımlar yapabilir. Benzer şekilde, eğitimde yapılan reformların öğrenci başarısı üzerindeki etkisini anlamak için kullanılan veriler eksik, hatalı veya yanlı olduğunda, değerlendirme süreci yanıltıcı hale gelebilir.

Etki değerlendirmesinde veri, yalnızca mevcut bir politikanın değerlendirilmesi için değil, aynı zamanda gelecekte uygulanacak politikaların tasarlanması için de kritik öneme sahiptir. Politika yapıcılar, geçmiş verileri analiz ederek benzer müdahalelerin hangi koşullarda daha etkili olduğunu belirleyebilir ve gelecekte daha iyi planlanmış projeler oluşturabilir.

Bu nedenle, etki değerlendirme süreçlerinde veri toplama, işleme, temizleme ve analiz etme aşamalarının dikkatli bir şekilde yürütülmesi gerekir. Kaliteli veri olmadan yapılan değerlendirmeler, yanlış politika kararlarına ve kamu kaynaklarının verimsiz kullanımına yol açabilir.

1.1.1 Etki Değerlendirmesinde Verinin Rolü

Etki değerlendirmesinde kullanılan veri, bir politikanın ya da programın öncesi ve sonrası durumlarını karşılaştırmak, neden-sonuç ilişkilerini ortaya koymak ve karar vericilere bilimsel temellere dayalı öneriler sunmak için kritik bir araçtır. Verinin doğru kullanımı sayesinde, politika yapıcılar hizmetlerin etkinliğini artırabilir, kaynakları verimli yönetebilir ve hedef grupların ihtiyaçlarını daha iyi karşılayabilir.

Örneğin, bir devlet destek programının istihdam üzerindeki etkisini ölçmek için iş gücü piyasası verilerine ihtiyaç vardır. Destek alan firmaların ve almayan firmaların karşılaştırılması, programın gerçekten istihdamı artırıp artırmadığını anlamamıza yardımcı olur.

1.1.2 Etki Değerlendirmede Veri Türleri ve Kaynakları

Etki değerlendirmesi için kullanılan veriler farklı kaynaklardan ve formatlardan gelebilir. Başlıca veri türleri şunlardır:

1.1.2.1 Kesitsel ve Panel Veri

- **Kesitsel Veri**: Tek bir zaman dilimindeki bireyleri, firmaları veya bölgeleri analiz eder.
- **Panel Veri**: Aynı birimleri farklı zaman dilimlerinde gözlemleyerek zaman içindeki değişimleri analiz etmeye olanak tanır.

1.1.2.2 Nicel ve Nitel Veri

- **Nicel Veriler**: Sayısal olarak ifade edilen, ölçülebilir değişkenlerdir (örneğin, gelir, istihdam oranı, üretim miktarı).
- **Nitel Veriler**: Katılımcı görüşleri, anket yanıtları veya mülakat sonuçları gibi kalitatif bilgileri içerir.

1.1.3 Etki Değerlendirme Modelleri ve Verinin Kullanımı

Etki değerlendirme sürecinde kullanılan yöntemler, verinin nasıl toplandığını ve analiz edildiğini belirler. Kullanılan yöntemler, verinin zaman içindeki değişimini ve müdahalelerin etkisini anlamamıza yardımcı olur.

- **Difference-in-Differences (DiD)**: Müdahale öncesi ve sonrası verileri karşılaştırarak etkiyi belirler.
- **Propensity Score Matching (PSM)**: Müdahale alan ve almayan bireyleri benzer özelliklere göre eşleştirerek daha doğru karşılaştırmalar yapar.
- **Instrumental Variables (IV)**: Nedenselliği ölçmek için kullanılan yöntemlerden biridir.

Bu yöntemlerin sağlıklı çalışabilmesi için doğru veri toplama, temizleme ve analiz süreçlerinin uygulanması gerekmektedir.

1.1.4 Neden Veri Olmazsa Etki Değerlendirme Yapılamaz?

Etki değerlendirmesi veriye dayalı bir süreçtir ve doğru kararlar alabilmek için güvenilir, eksiksiz ve analiz edilebilir veri setlerine ihtiyaç vardır. Eğer veri eksik, hatalı veya yanlı ise, politika değerlendirmeleri yanlış sonuçlar verebilir ve bu da yanlış politika kararlarına yol açabilir.

Veri olmadan etki değerlendirme yapılamaz çünkü:

- Müdahalenin etkisini ölçmek için karşılaştırmalar yapılamaz.
- Politika önerileri bilimsel dayanağa sahip olmaz.
- Kaynakların verimli kullanımı garanti edilemez.
- Sonuçların doğruluğu sorgulanır ve politika yapıcılar yanlış yönlendirilebilir.

Bu nedenle, veri toplama, temizleme, analiz etme ve raporlama süreçlerinin doğru yönetilmesi, etkili bir etki değerlendirmesi için kritik öneme sahiptir.

1.2 Etki Değerlendirmede Veri Seti Nasıl Olmalı?

Etki değerlendirmesi, belirli bir politika, program veya müdahalenin beklenen sonuçları üretip üretmediğini anlamak için kullanılan analitik bir süreçtir. Bu sürecin başarısı, kullanılan veri setinin doğruluğuna, kapsamına ve analiz için uygunluğuna bağlıdır. Eğer veri seti eksik, tutarsız veya hatalı ise, analizler yanlış yönlendirme yapabilir ve alınan politika kararları gerçek durumu yansıtmayabilir.

İyi bir etki değerlendirme veri seti, yalnızca mevcut bir politikayı veya müdahaleyi analiz etmekle kalmaz, aynı zamanda gelecekteki karar süreçlerine de rehberlik eder. Doğru yapılandırılmış bir veri seti, karar alıcıların ve araştırmacıların müdahale öncesi ve sonrası durumları kıyaslamasına, neden-sonuç ilişkilerini analiz etmesine ve politika önerilerini bilimsel temellere dayandırmasına olanak tanır.

Ancak, etki değerlendirme sürecinde kullanılan veri setleri çeşitli eksiklikler, hatalar ve metodolojik zorluklar içerebilir. Eksik veriler, aykırı değerler, yanlış kodlanmış değişkenler, zaman uyumsuzluğu ve örneklem seçim yanlılığı gibi sorunlar analizlerin güvenilirliğini ciddi şekilde etkileyebilir. Bu nedenle, veri setinin etki değerlendirmesi için hazırlanma süreci titizlikle yürütülmelidir.

Bu bölümde, etki değerlendirme için ideal bir veri setinin nasıl olması gerektiği, hangi bileşenleri içermesi gerektiği ve hangi kalite standartlarını karşılaması gerektiği ayrıntılı olarak ele alınacaktır. Ayrıca, etki değerlendirme yöntemleriyle uyumlu veri yapılarının nasıl oluşturulması gerektiği ve veri setlerinin güvenilir analizler için nasıl hazırlanması gerektiği açıklanacaktır.

1.2.1 Veri Setinin Temel Yapısı

Etki değerlendirmesi için kullanılan veri setleri genellikle gözlemler (satırlar) ve değişkenlerden (sütunlar) oluşur. Her satır, bir birey, firma, bölge veya politika müdahalesine tabi olan bir birimi temsil eder. Her sütun ise bu birime ilişkin farklı değişkenleri içerir.

Veri setinde şu temel bileşenler bulunmalıdır:

- **Kimlik Bilgileri (ID değişkenleri)**: Özniteliklerin eşleşmesini sağlayan benzersiz bir kimlik numarası (örn. birey ID, firma ID, bölge kodu vb.).
- **Bağımlı Değişken (Sonuç Değişkeni)**: Politika veya müdahalenin etkilediği değişken (örn. istihdam, gelir, eğitim seviyesi vb.).
- **Bağımsız Değişkenler**: Etkiyi belirleyen veya kontrol değişkeni olarak kullanılan faktörler (örn. yaş, sektör, hanehalkı büyüklüğü, ekonomik göstergeler vb.).
- **Müdahale Değişkeni (Tedavi/Müdahale Değişkeni):** Müdahaleye maruz kalıp kalmadığını gösteren değişken (örn. 1 = müdahale aldı, 0 = almadı).
- **Zaman Değişkeni:** Etki değerlendirmesi zaman serileri veya panel veri ile yapılıyorsa tarih veya yıl değişkeni.

Örnek bir etki değerlendirme veri seti (Panel veri yapısında):

ID	Yıl	Müdahale	Gelir	Eğitim Yılı	Sektör	Bölge
101	2018	0	5000	12	Hizmet	İstanbul
101	2019	1	6000	12	Hizmet	İstanbul
102	2018	0	4500	10	Tarım	Ankara
102	2019	1	5500	10	Tarım	Ankara

Bu yapıda **müdahale değişkeni**, bireyin politikadan önce ve sonra nasıl değiştiğini anlamamıza yardımcı olur.

1.2.2 Veri Setinin Kalite Kriterleri

Etki değerlendirme veri setlerinin yüksek kaliteli olması için şu kriterlere uyması gerekir:

- **Eksiksiz ve Tutarlı Olmalı**: Tüm birimler ve zaman dilimleri için eksiksiz veri içermeli, tutarsız girişlerden kaçınılmalıdır.
- **Temizlenmiş Olmalı**: Eksik veriler tamamlanmalı, aykırı değerler kontrol edilmeli ve veriler doğru formatta saklanmalıdır.
- **Tekilleştirilmiş Olmalı**: Aynı birime ait birden fazla satır veya çakışan kayıtlar olmamalıdır.
- **Standartlaştırılmış Olmalı**: Değişken isimleri ve formatları tutarlı olmalı, aynı türdeki veriler uyumlu formatta olmalıdır (örn. tarih formatı, kategorik değişkenlerin kodlanması).

- **Analiz İçin Uygun Olmalı:** Kullanılacak yöntem (Difference-in-Differences, Propensity Score Matching vb.) için gerekli değişkenleri içermelidir.

1.2.3 Veri Setinin Türü ve Yapısı

Etki değerlendirmesi için kullanılan veri setleri genellikle iki ana yapıya sahiptir:

1.2.3.1 Kesitsel Veri (Cross-Sectional Data)

- Tek bir zaman noktasındaki bireyleri veya firmaları içerir.
- Politikaların kısa vadeli etkilerini ölçmek için uygundur.
- Örnek: 2023 yılı için girişimcilik destek programına katılan firmalar ve katılmayan firmaların performansları.

Örnek Kesitsel Veri Seti:

Firma ID	Müdahale	Gelir	Çalışan Sayısı	Sektör
201	1	75000	15	Üretim
202	0	60000	12	Ticaret

1.2.3.2 Panel Veri (Panel Data)

- Aynı bireyleri veya firmaları farklı zaman dilimlerinde içerir.
- Difference-in-Differences (DiD) gibi yöntemler için gereklidir.
- Politikanın uzun vadeli etkisini incelemek için uygundur.

Örnek Panel Veri Seti (Zaman içindeki değişimi gösterir):

Firma ID	Yıl	Müdahale	Gelir	Çalışan Sayısı
201	2021	0	50000	10
201	2022	1	60000	12
202	2021	0	55000	11
202	2022	0	58000	11

Panel veri kullanıldığında müdahaleden önce ve sonra bireylerin/firma performanslarının nasıl değiştiği karşılaştırılabilir.

1.2.4 Etki Değerlendirme Yöntemleri için Veri Yapıları

Etki değerlendirmesinde kullanılan yöntemlere göre veri setinin özellikleri değişebilir:

- **Propensity Score Matching (PSM)** → Müdahale ve karşılaştırma grubu arasında benzer özelliklere sahip gözlemler bulunmalı.
- **Difference-in-Differences (DiD)** → Zaman içinde gözlemlenen aynı bireyler veya firmalar olmalı.
- **Regression Discontinuity (RD)** → Kesikli bir değişken üzerinden belirlenen bir eşik değeri ile müdahale alan ve almayan grupların ayrımı yapılmalı.

Her yöntem için veri setinin doğru formatta olması analiz sürecinin güvenilirliği açısından kritik öneme sahiptir.

1.2.5 Etki Değerlendirme için İdeal Veri Seti

Etki değerlendirme veri seti, aşağıdaki özellikleri taşımalıdır:

- **Zengin Değişken Yapısı**: Etkiyi ölçmek için bağımlı değişken, müdahale değişkeni ve kontrol değişkenleri içermelidir.

- **Doğru Veri Formatı**: Panel veri veya kesitsel veri analiz yöntemine uygun şekilde düzenlenmelidir.

- **Eksiksiz ve Temiz**: Eksik veriler doldurulmalı, aykırı değerler düzeltilmeli ve tutarsız girişler kaldırılmalıdır.

- **Zaman Boyutunu İçermeli**: Müdahale öncesi ve sonrası gözlemler mümkünse bulunmalıdır.

Özetle, iyi yapılandırılmış, eksiksiz ve doğru bir veri seti, etki değerlendirme süreçlerinin güvenilirliğini artırır ve politika yapıcılara doğru yönlendirmeler sunar.

1.3 Etki Değerlendirmede Veri Seti ile İlgili Karşılaşılan Sorunlar ve Doğurduğu Sonuçlar

Etki değerlendirme sürecinde kullanılan veri setleri, analizlerin doğruluğunu, geçerliliğini ve güvenilirliğini belirleyen en kritik bileşenlerden biridir. Verinin eksik, hatalı veya yanlış yapılandırılmış olması, değerlendirme sürecini yanıltıcı hale getirebilir ve politika yapıcıların yanlış sonuçlara ulaşmasına neden olabilir. Bu durum, etkisiz veya gereksiz politikaların sürdürülmesine ya da etkili bir müdahalenin gereğinden az desteklenmesine yol açabilir.

Etki değerlendirme sürecinde karşılaşılan veri seti sorunları veri manipülasyonu, veri dönüşümü, veri şekillendirme, metin manipülasyonu ve fonksiyonlarla veri işleme aşamalarında yapılan hatalardan kaynaklanabilir. Bu aşamalarda dikkat edilmesi gereken en önemli unsurlar, veri setinin eksiksiz, tutarlı, doğru formatlanmış ve analiz için uygun yapıda olmasıdır. Aksi halde, yapılan modellemeler güvenilirliğini kaybedebilir ve yanlış politika sonuçlarına neden olabilir.

Veri setleriyle ilgili yaşanan başlıca sorunlar; eksik veriler, aykırı değerler, örneklem seçim yanlılığı, zaman uyumsuzluğu, yanlış veri eşleşmeleri ve eksik tanımlanmış değişkenler olarak öne çıkmaktadır. Bu tür hatalar, veri analizlerinin yanıltıcı sonuçlar üretmesine, nedensellik ilişkilerinin hatalı yorumlanmasına ve yanlış politika önerilerine neden olabilir.

Bu bölümde, etki değerlendirme sürecinde veri setleriyle ilgili yaygın olarak karşılaşılan sorunlar detaylı bir şekilde ele alınacak ve bu sorunların değerlendirme süreçleri üzerindeki etkileri tartışılacaktır. Aynı zamanda, veri kalitesini artırmaya yönelik bazı temel çözümler ve iyi uygulama yöntemleri de incelenecektir.

1.3.1 Veri Hazırlama, Dönüştürme ve Şekillendirme ile İlgili Genel Sorunlar

Etki değerlendirme analizlerinde kullanılan veri setleri, ham veri olarak toplandıktan sonra çeşitli işlemlerden geçirilerek analiz edilebilir hale getirilmelidir. Ancak, yanlış veya eksik veri manipülasyonu, hatalı veri dönüşümleri, yanlış veri birleştirme işlemleri ve eksik veri temizliği gibi faktörler, sonuçların güvenilirliğini zayıflatabilir.

- Sütun Seçimi (`select()`) ve Satır Filtreleme (`filter()`) yanlış yapıldığında, analiz için gerekli olan değişkenler yanlışlıkla veri setinden çıkarılabilir veya örneklemde önemli bir kayma yaşanabilir.
- Satırların sıralanması (`arrange()`), zaman serisi analizlerinde doğru yapılmazsa, etki değerlendirme sonuçları hatalı olabilir.
- Yeni değişkenler oluşturulurken (`mutate()`) veya istatistiksel özetler hesaplanırken (`summarise()`) yanlış hesaplamalar, yanıltıcı bulgular üretebilir.
- Veri kaynaklarının birleştirilmesi (`merge()`) yanlış yapılırsa, aynı bireyler veya firmalar için tutarsız bilgiler içeren bir veri seti oluşabilir.
- Metinsel verilerin düzensizliği (yanlış kategorik değişkenler, eksik etiketler) analizleri bozabilir ve yanlış sınıflandırmalara yol açabilir.

Sonuçları:

- Yanlış veri seçimi ve filtreleme, politika etkisinin yanlış hesaplanmasına neden olabilir.
- Eksik veya yanlış dönüştürülen veriler, analiz sonuçlarını çarpıtabilir.
- Yanlış eşleşmeler ve eksik gözlemler nedeniyle model hatalı tahminlerde bulunabilir.
- Etki değerlendirmesi hatalı yapıldığında, karar alıcılar yanlış yönlendirilebilir.

1.3.2 Eksik Verilerle Çalışma ve Olası Sorunlar

Eksik veriler, etki değerlendirme süreçlerinde en yaygın karşılaşılan sorunlardan biridir. Veri setlerinde eksik değerlerin bulunması, analizlerin güvenilirliğini azaltabilir ve sonuçların yanlış yorumlanmasına yol açabilir.

1.3.2.1 Eksik Verilerin Kaynakları ve Yaygın Nedenleri

Eksik veriler, farklı nedenlerden dolayı ortaya çıkabilir:

- **Yanıt Eksiklikleri**: Anketlerde bazı sorulara verilen yanıtların eksik olması.
- **İdari Kayıtlardaki Boşluklar**: Kamu veritabanlarında bazı bireyler veya firmalar hakkında eksik bilgiler bulunması.
- **Örneklemden Kaynaklanan Eksiklikler**: Belirli bir grup bireyin veya firmanın değerlendirme dışında kalması.
- **Zaman Uyum Sorunları**: Verinin belirli dönemlerde eksik olması.

1.3.2.2 Eksik Verilerin Etkileri

Eksik veriler analiz sürecinde ciddi sorunlara yol açabilir:

- **Yanlılık Oluşabilir**: Eksik veriler sistematik bir biçimde belirli bir grupta toplanıyorsa (örneğin, düşük gelirli bireylerin anketleri tam doldurmaması), analizde yanlılık meydana gelebilir.
- **Örneklem Temsiliyetini Bozabilir**: Veri setindeki eksik değerler nedeniyle analiz edilen örneklem, evreni tam olarak temsil etmeyebilir.
- **Model Performansını Düşürebilir**: Eksik değerler yanlış tahminlere yol açabilir.

Sonuçları:

- Eksik veri sorunları uygun yöntemlerle ele alınmazsa, analiz sonuçları ciddi şekilde yanlı hale gelebilir.
- Eksik verilerin yanlış doldurulması, model performansını düşürebilir.
- Verinin kalitesini artırmak için uygun eksik veri yöntemleri uygulanmalıdır.

1.3.3 Aykırı Değerler ile Çalışma ve Olası Sorunlar

Aykırı değerler, veri setinde diğer gözlemlerden büyük ölçüde farklılık gösteren gözlemlerdir. Etki değerlendirme sürecinde, aykırı değerlerin varlığı analizleri yanıltıcı hale getirebilir.

1.3.3.1 Aykırı Değerlerin Kaynakları

- **Ölçüm Hataları:** Veri girişlerinde hata yapılması sonucu ortaya çıkabilir.
- **Doğal Aykırılıklar**: Bazı bireyler veya firmalar gerçekten aşırı uç değerlere sahip olabilir.
- **Yanlış Veri Kodlaması**: Negatif maaş değerleri gibi mantıksız veriler bulunabilir.

1.3.3.2 Aykırı Değerlerin Etkileri

- **Model Katsayılarını Çarpıtabilir**: Regresyon analizinde uç değerler tahminleri bozabilir.
- O**rtalama ve Standart Sapmayı Etkileyebilir**: Veri setinin genel istatistikleri yanıltıcı olabilir.
- **Yanlış Politikaların Önerilmesine Yol Açabili**r: Yanlış analiz sonuçları, etkisiz veya hatalı politikaların önerilmesine neden olabilir.

Sonuçları:

- Aykırı değerler düzgün ele alınmazsa, analiz sonuçları yanıltıcı olabilir.
- Yanlış politika etkileri hesaplanabilir.
- Uygun istatistiksel yöntemlerle aykırı değerler kontrol edilmelidir.

1.3.4 Veri Setinin Yanıltıcı veya Eksik Tanımlanması

Veri setinde bazı değişkenlerin eksik tanımlanması veya yanlış şekilde kategorize edilmesi, analiz süreçlerini olumsuz etkileyebilir.

1.3.4.1 Yanlış Kategorik Değişken Tanımları

- Örneğin, "Bölgesel Kalkınma Programı"na dahil olan şehirlerin yanlış kategorize edilmesi, analiz sonuçlarını bozabilir.

1.3.4.2 Zaman Uyum Sorunları

- Müdahale etkisini ölçmek için yeterli zaman geçmeden yapılan analizler, yanlış politika sonuçları çıkarılmasına neden olabilir.

Sonuçları:

- Verinin yanlış tanımlanması, analizlerin yanlış yorumlanmasına yol açabilir.
- Politikalar hakkında hatalı çıkarımlar yapılabilir.
- Etki değerlendirme süreçlerinde veri yapısının doğru tanımlanması sağlanmalıdır.

1.4 Etki Değerlendirme İçin Veri Setlerinin Hazır Hale Getirilmesi

Etki değerlendirme sürecinde kullanılan veri setlerinin doğru, eksiksiz ve analiz için uygun hale getirilmesi, elde edilecek sonuçların güvenilirliği açısından kritik bir öneme sahiptir. Veri setinde yer alan eksik gözlemler, hatalı girişler, aykırı değerler veya yanlış kodlanmış değişkenler giderilmezse, analiz sonuçları yanıltıcı olabilir ve politika yapıcıların yanlış yönlendirilmesine neden olabilir.

Etki değerlendirme için uygun bir veri seti oluşturmak, sadece veriyi toplamakla sınırlı değildir. Verinin kalite kontrol süreçlerinden geçirilmesi, eksik veya hatalı verilerin düzeltilmesi ve analiz yöntemlerine uygun şekilde dönüştürülmesi gerekir. Eksik verilerin doldurulması, aykırı değerlerin analiz edilmesi, veri manipülasyonu ve standardizasyon gibi işlemler, veri setinin analize hazır hale getirilmesi için mutlaka uygulanmalıdır.

Etki değerlendirme modelleri (Difference-in-Differences, Propensity Score Matching, Regression Discontinuity vb.) belirli veri yapıları gerektirdiğinden, veri setinin analiz yöntemlerine uygun hale getirilmesi de büyük önem taşır. Örneğin, panel veri kullanan Difference-in-Differences yöntemi için zaman boyutunun eksiksiz olması gerekirken, Propensity Score Matching yöntemi için müdahale ve kontrol gruplarının benzer özellikler taşıması sağlanmalıdır.

Bu nedenle, veri setlerini analize uygun hale getirmek için aşağıdaki işlemler titizlikle uygulanmalıdır:

- Veri manipülasyonu ve dönüştürme işlemleri (sütun seçimi, veri standardizasyonu, zaman serisi uygunluğu vb.)
- Eksik verilerin belirlenmesi ve uygun şekilde doldurulması
- Aykırı değerlerin tespiti ve uygun yöntemlerle düzeltilmesi
- Farklı veri kaynaklarının tutarlı hale getirilmesi ve birleştirilmesi
- Doğru değişken tanımlamaları yapılarak veri setinin analiz yöntemine uygun hale getirilmesi

Bu adımların her biri, analizin güvenilirliğini artırmak ve politika etkilerini doğru ölçmek için kritik rol oynamaktadır. Eksik veya hatalı veriler düzeltilmeden yapılan analizler, politika yapıcıların hatalı kararlar almasına neden olabilir. Bu bölümde, veri setinin etki değerlendirme sürecine uygun hale getirilmesi için atılması gereken temel adımlar detaylı bir şekilde ele alınacaktır.

1.4.1 Veri Manipülasyonu ve Dönüştürme İşlemleri

Veri setlerinin analiz öncesinde uygun hale getirilmesi için sütun seçimi, satır filtreleme, sıralama ve veri dönüştürme işlemlerinin doğru yapılması gerekmektedir.

- **Gereksiz Sütunların Çıkarılması**: `select()` fonksiyonu ile analizde kullanılmayacak değişkenler veri setinden temizlenmelidir.
- **Hedef Grupların Seçilmesi**: `filter()` fonksiyonu ile yalnızca analiz için gerekli bireyler veya firmalar veri setinde bırakılmalıdır.

- **Zaman Serisi Uygunluğu**: `arrange()` fonksiyonu ile zaman içindeki değişimler doğru bir sıraya konulmalıdır.
- **Veri Standartlaştırma ve Yeniden Kodlama**: `mutate()` ve `rename()` fonksiyonları kullanılarak değişken isimleri ve kategorik değişkenler düzenlenmelidir.

Etki Değerlendirme İçin Önemi:

- Müdahale ve kontrol gruplarının doğru tanımlanmasını sağlar.
- Modelin tutarlı bir veri seti üzerinden çalışmasına olanak tanır.
- Zamansal tutarsızlıkları ve yanlış filtreleme hatalarını önler.

1.4.2 Eksik Verilerle Başa Çıkma ve Tamamlama Süreci

Eksik veriler, analizlerin güvenilirliğini düşüren en büyük sorunlardan biridir. Eksik veri oranı yüksekse, analiz yanlı hale gelebilir ve yanlış politika kararları alınabilir.

- **Eksik Veri Analizi:** Eksik veri yüzdesi belirlenmeli ve sistematik bir desen olup olmadığı incelenmelidir. `naniar`, `VIM` gibi paketler ile eksik veri görselleştirilebilir.
- **Eksik Verileri Doldurma Yöntemleri:**
  - Ortalama, Medyan veya Mod ile Doldurma (Basit ama her zaman güvenilir değil)
  - Regresyon Tabanlı İmputasyon
  - Çoklu İmputasyon (MICE Paketi ile)
  - K En Yakın Komşu (KNN) Yöntemi
- **Eksik Verilerin Yönetimi:** Eğer eksik veriler belirli bir gruba özgü ise (örneğin düşük gelirli bireylerde daha fazla eksik veri olması gibi), analiz sırasında bu eksiklik dikkate alınmalıdır.

Etki Değerlendirme İçin Önemi:

- Eksik verilerin uygun şekilde doldurulması, analizlerin güvenilirliğini artırır.
- Örneklemde sistematik veri kaybı olup olmadığı belirlenebilir.
- Eksik verilerin yanlış doldurulması önlenerek modelin hatalı tahminler yapması engellenir.

1.4.3 Aykırı Değerleri Tespit Etme ve Düzenleme

Aykırı değerler, analiz sonuçlarını çarpıtabilir ve yanlış politika önerilerine yol açabilir. Bu nedenle aykırı değerlerin tespit edilmesi ve uygun şekilde ele alınması gerekmektedir.

- **Aykırı Değerleri Belirleme:**
  - **İstatistiksel yöntemlerle tespit edilmesi**: Z-skoru, IQR yöntemi, Boxplot kullanımı.
  - **Görselleştirme ile inceleme**: `boxplot()`, `ggplot2` kullanımı.

- **Aykırı Değerlerle Baş Etme Yöntemleri:**
  - **Veri Setinden Çıkarma**: Aşırı uç değerler tamamen kaldırılabilir. Ancak dikkatli olunmalıdır, çünkü bazı aykırı değerler gerçekte politika etkisini yansıtan değerler olabilir.
  - **Winsorization**: Aykırı değerlerin belirli bir sınıra çekilmesi.
  - **Robust Modeller Kullanma**: Aykırı değerlere duyarlı olmayan analiz yöntemleri tercih edilmelidir.

Etki Değerlendirme İçin Önemi:

- Aykırı değerler kontrol edilmezse, müdahale etkisi yanlış hesaplanabilir.
- Bazı grupların aşırı etkili görünüp gerçekte farklı sonuçlar doğurması önlenir.
- Politikaların uzun vadeli etkileri daha doğru bir şekilde hesaplanır.

1.5 Sonuç

Etki değerlendirme sürecinin sağlıklı bir şekilde yürütülebilmesi, kullanılan verinin kalitesiyle doğrudan ilişkilidir. Eksik, hatalı veya düzensiz bir veri seti ile yapılan analizler, yanlış sonuçlara yol açarak politika yapıcıları yanıltabilir ve kaynakların etkisiz kullanılmasına neden olabilir. Bu nedenle, veri setlerinin manipülasyon, dönüşüm, temizleme ve bütünleştirme işlemlerinden geçirilerek analiz için uygun hale getirilmesi gerekmektedir.

Eksik verilerin uygun yöntemlerle tamamlanması, aykırı değerlerin kontrol edilmesi, zaman uyumsuzluklarının giderilmesi ve veri setinin standardize edilmesi, değerlendirme sürecinin doğruluğunu artıracaktır. Ayrıca, veri türüne ve kullanılan yönteme bağlı olarak veri setinin kesitsel veya panel veri formatında düzenlenmesi, nedensellik ilişkilerinin daha sağlıklı kurulmasına olanak tanıyacaktır.

Etki değerlendirme süreçlerinde doğru veriyle çalışmak, müdahale ve kontrol gruplarının karşılaştırılmasını kolaylaştırır, nedensel ilişkileri daha iyi anlamamızı sağlar ve daha güvenilir politika önerileri geliştirilmesine katkıda bulunur. Ancak, veri setleri hazırlanırken yukarıda ele alınan hataların önlenmesi büyük önem taşımaktadır.

Bu çalışma, yukarıda sayılan konuların önemli bir kısmında yardımcı olacak niteliktedir. Veri setlerinin etki değerlendirme sürecine uygun hale getirilmesine yönelik adımlar, eksik veri yönetimi, aykırı değer analizi, veri bütünleştirme ve standardizasyon gibi kritik konulara değinilerek kapsamlı bir yol haritası sunulmuştur. Bu sayede, politika yapıcılar ve araştırmacılar, daha sağlıklı analizler yaparak karar alma süreçlerinde güvenilir bilgiye dayalı değerlendirmeler gerçekleştirebileceklerdir.

Part I

Veri Dönüşümleri

Etki değerlendirme süreçleri, bir politika ya da müdahalenin hedeflenen sonuçlara ulaşıp ulaşmadığını belirlemek amacıyla yürütülen sistematik analizleri kapsar. Bu süreçlerin güvenilir olması, doğru yöntemlerin uygulanması kadar, kullanılan veri setlerinin eksiksiz, tutarlı ve analiz için uygun hale getirilmesine de bağlıdır. Veri analizine başlamadan önce, veri manipülasyonu ve veri temizleme aşamalarının dikkatlice yürütülmesi, elde edilen sonuçların doğruluğunu artırarak karar alıcıların daha sağlıklı değerlendirmeler yapmasına olanak tanır. Eksik veya yanlış işlenmiş verilerle yürütülen analizler, politika etkilerinin hatalı hesaplanmasına ve yanıltıcı çıkarımlara neden olabilir.

Veri manipülasyonu, ham verinin analiz için uygun hale getirilmesi sürecini kapsayan işlemleri ifade eder. Sütun seçimi, satır filtreleme, değişken dönüştürme ve verinin istatistiksel olarak özetlenmesi gibi işlemler, veri setinin daha anlamlı ve analiz edilebilir hale getirilmesini sağlar. Manipülasyon aşaması, özellikle büyük veri setlerinde gereksiz veya tekrarlayan bilgilerin çıkarılması, eksik gözlemlerin yönetilmesi ve veri yapısının modelleme için optimize edilmesi gibi kritik adımları içermektedir.

Bununla birlikte, veri analizinin güvenilirliği açısından veri temizleme süreci büyük önem taşımaktadır. Veri setlerinde eksik değerler, aykırı gözlemler, yanlış formatlanmış değişkenler veya hatalı kayıtlar bulunabilir. Bu tür veriler, analiz süreçlerini olumsuz etkileyerek yanlış sonuçlara yol açabilir. Eksik verilerin doldurulması, aykırı değerlerin yönetilmesi ve verinin standardizasyonu gibi işlemler, analiz sürecinin daha sağlıklı ilerlemesini sağlar. Veri temizleme aşamasında eksik değerlerin nasıl ele alınacağı, hatalı girişlerin nasıl tespit edileceği ve veri türlerinin nasıl dönüştürüleceği gibi konular titizlikle ele alınmalıdır.

Bu çalışma, etki değerlendirme süreçlerinde veri manipülasyonu ve veri temizleme adımlarının nasıl gerçekleştirileceğini ele alarak, R programlama dili ve Quarto platformu aracılığıyla uygulanabilir bir rehber sunmayı amaçlamaktadır. Eksiksiz, temizlenmiş ve analiz için optimize edilmiş veri setleri, etki değerlendirme süreçlerinin güvenilirliğini artırarak politika yapıcıların daha sağlam kararlar almasına katkı sağlayacaktır.

2 Veri Manipülasyonu

2.1 Temel Veri Manipülasyonu İşlemleri

2.1.1 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:**

```
# 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
```

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

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

```
# 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
```

2.1.1.1 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**

```
# 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
```

2.1.1.2 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
```

2.1.1.3 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
```

2.1.1.4 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.

```r
# 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.

```r
# 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
```

2.1.2 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).

2.1.2.1 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
```

2.1.2.2 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.
```

```r
# 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**

```r
# 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
```

2.1.2.3 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.

2.1.2.4 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
```

2.1.2.5 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     70    154
2     72    164
3     67    142
```

- **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
```

2.1.3 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.

2.1.3.1 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
```

2.1.3.2 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
```

2.1.4 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
```

2.2 Veri Dönüştürme

2.2.1 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.

2.2.1.1 `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*

2.2.1.2 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
```

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
```

2.2.1.3 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
```

2.2.1.4 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
```

2.2.1.5 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
```

2.2.2 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.

2.2.2.1 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`**).

2.2.2.2 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:

2.2.3 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.

2.2.3.1 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
```

2.2.3.2 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
```

2.3 Veri Şekillendirme

2.3.1 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.

2.3.1.1 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
```

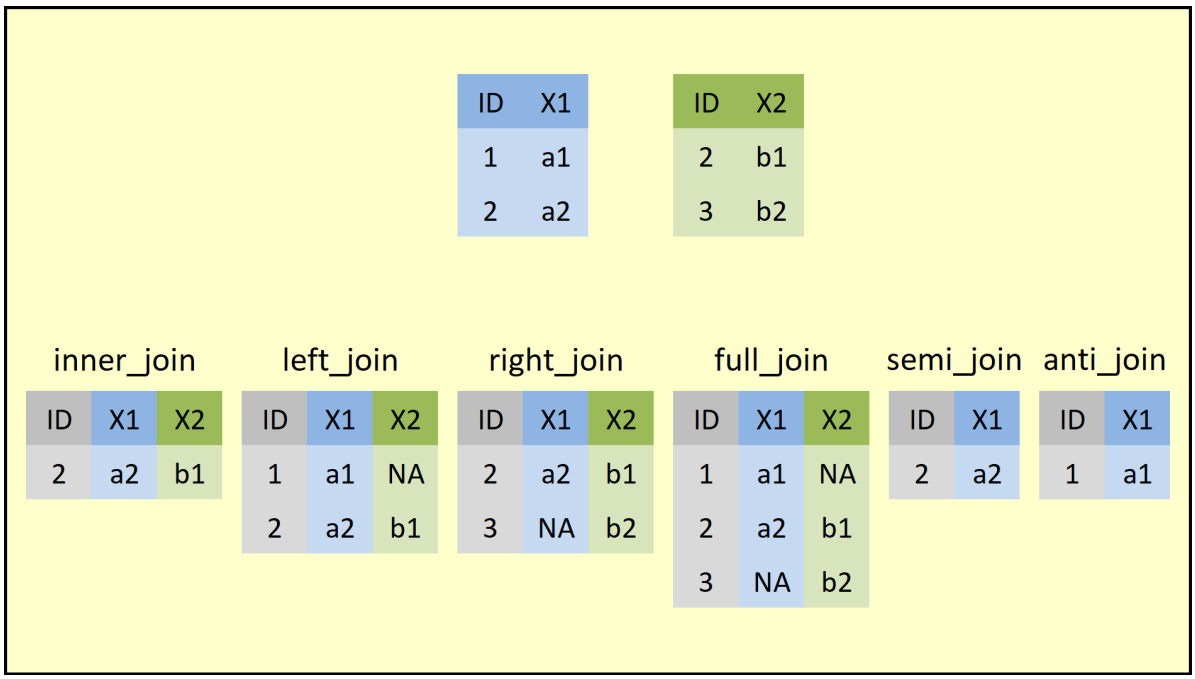
```
  ID X1
1  1 a1
2  2 a2
```

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
```

...



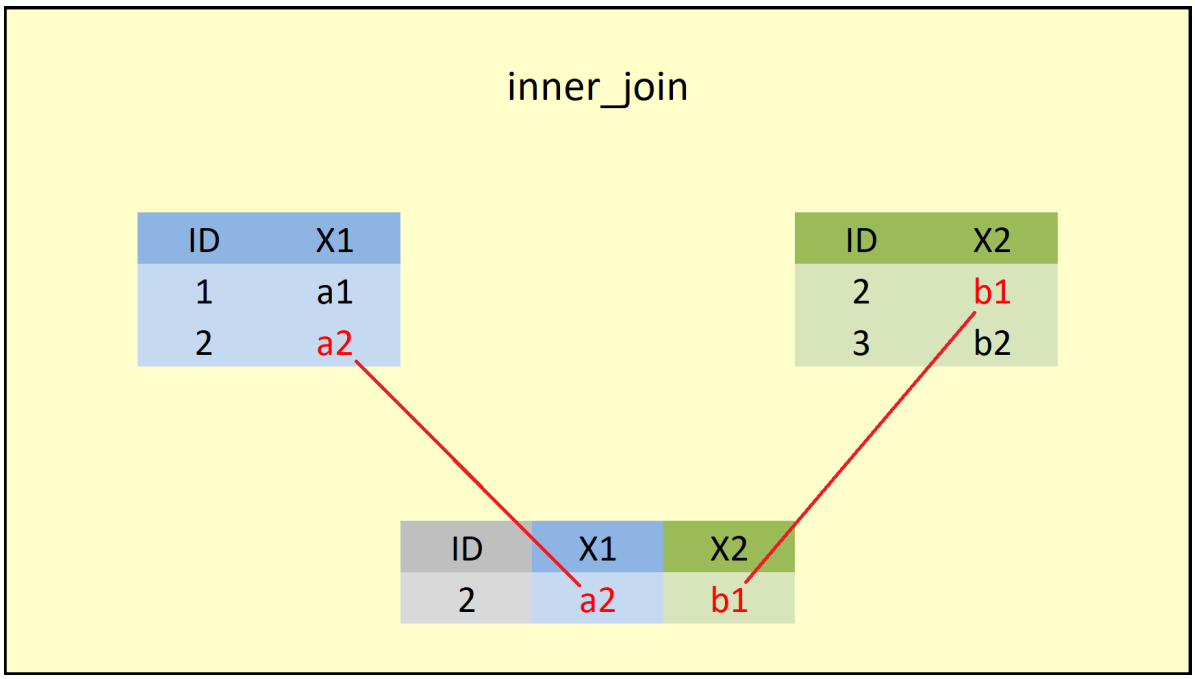
2.3.1.2 Inner Join - İç Birleştirme İşlemi

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.

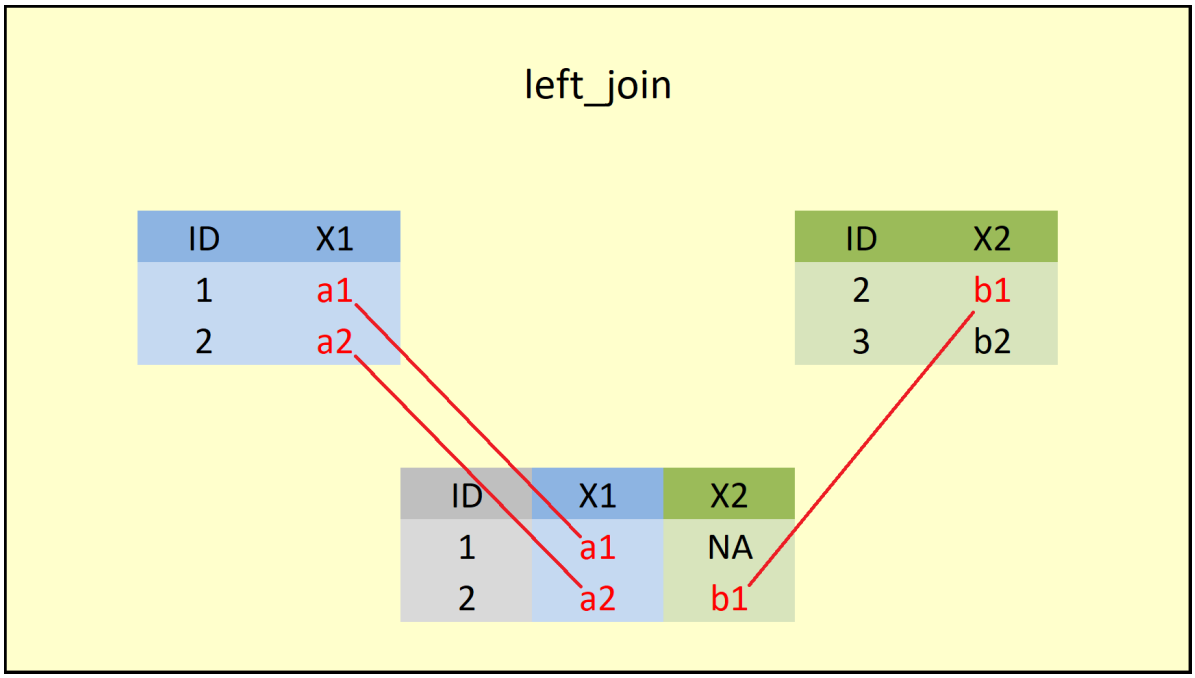
2.3.1.3 Left Join - Sol Birleştirme İşlemi

```
# 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.



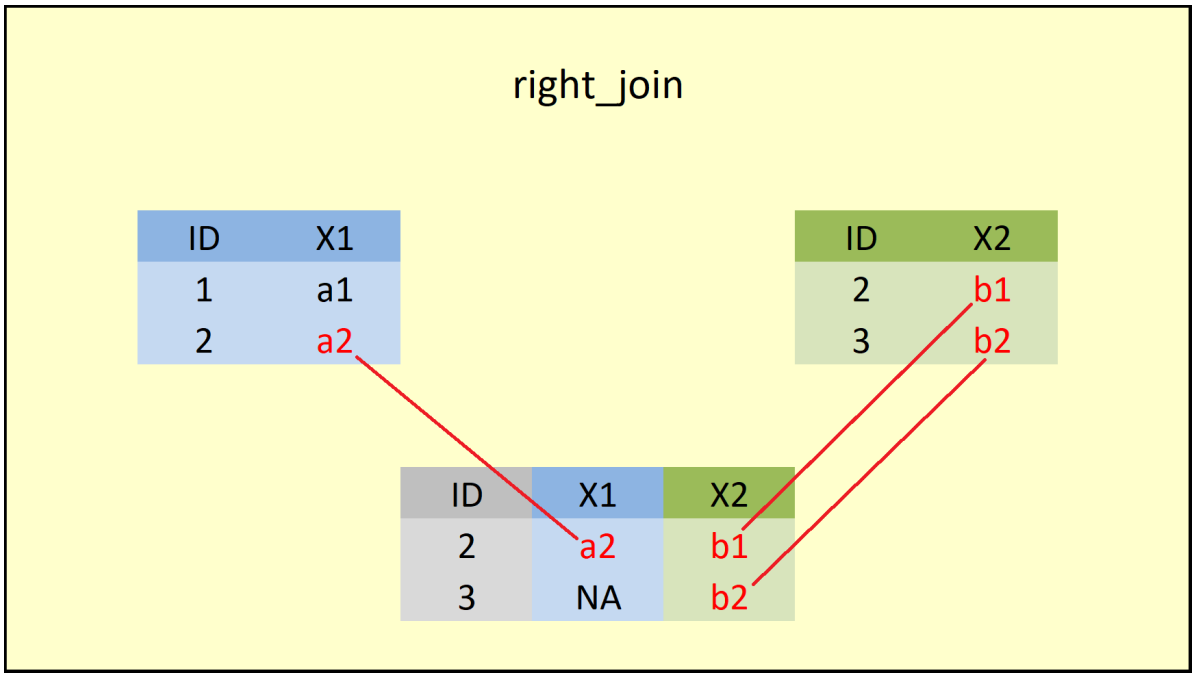
Left join, sol veri çerçevesindeki tüm satırları tutar ve sağ veri çerçevesinden yalnızca eşleşen değerleri ekler. Eşleşme yoksa eksik değerler (NA) atanır. **Inner join** ise, yalnızca her iki veri çerçevesinde **ortak olan** satırları tutar ve diğer satırları dışlar.

2.3.1.4 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.



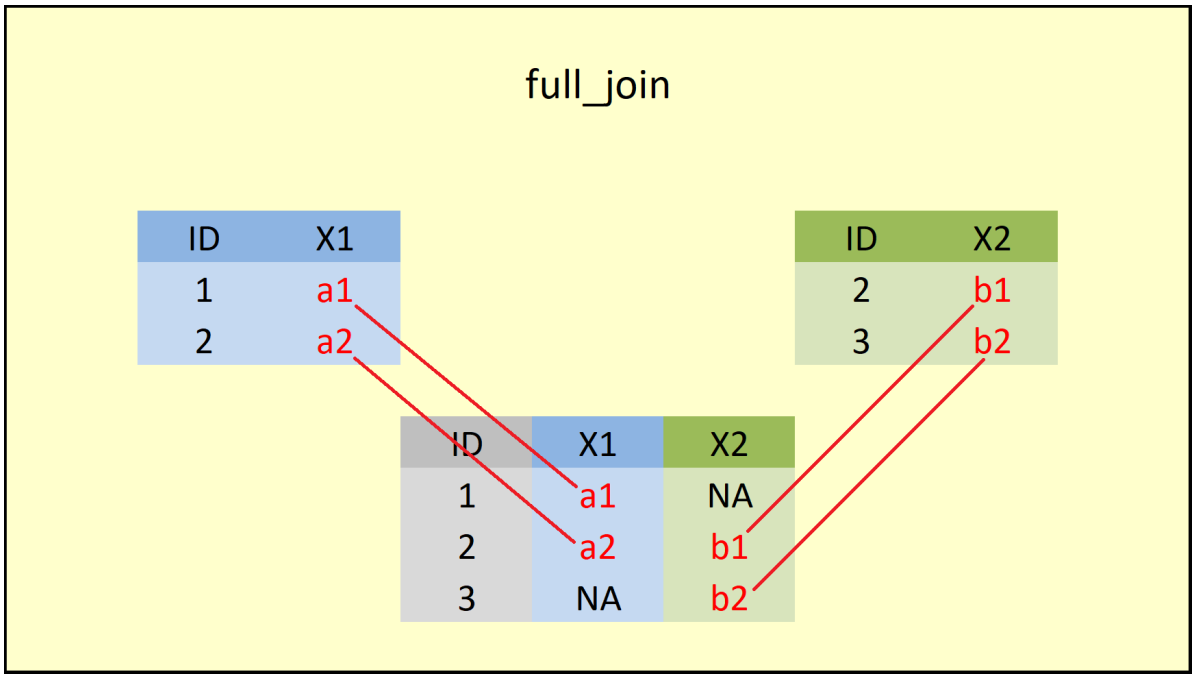
Right join, sağ veri çerçevesindeki tüm satırları korurken sol veri çerçevesinden yalnızca eşleşen değerleri ekler ve eşleşme olmayan satırlar için eksik değerler (**NA**) atar. **Left join** ile farkı, **Right join**'in sağ veri çerçevesini referans alması, **Left join**'in ise sol veri çerçevesini 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.

2.3.1.5 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.

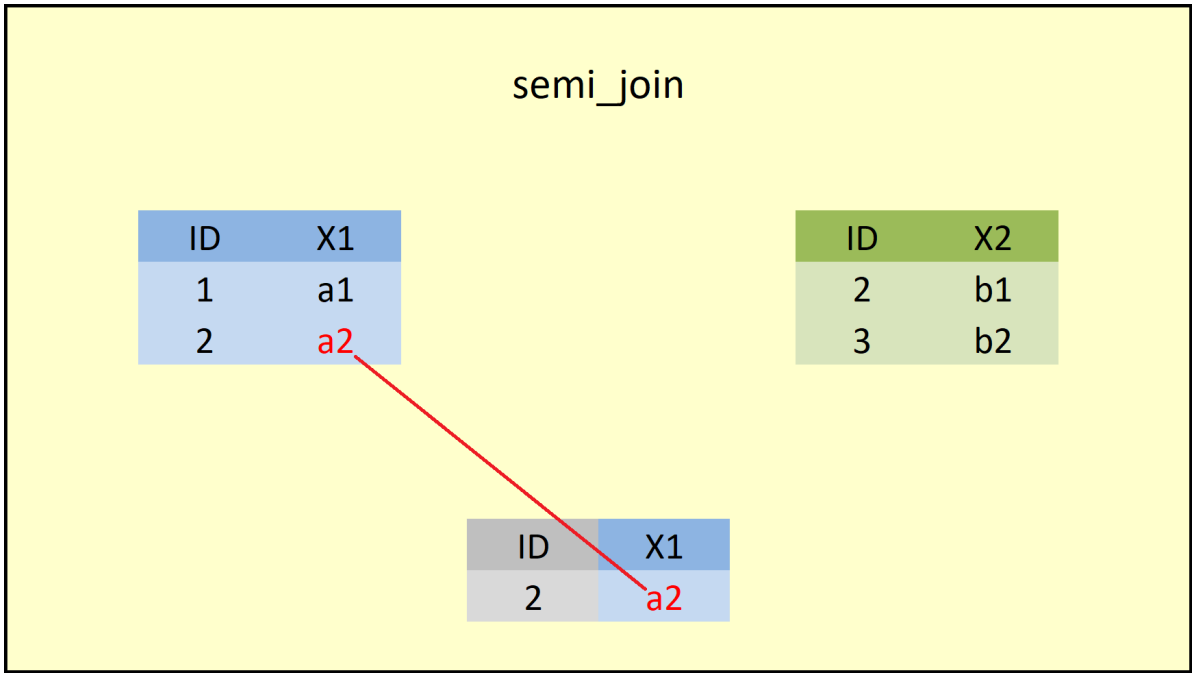
2.3.1.6 Semi Join - Yarı Birleştirme İşlemi

```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.



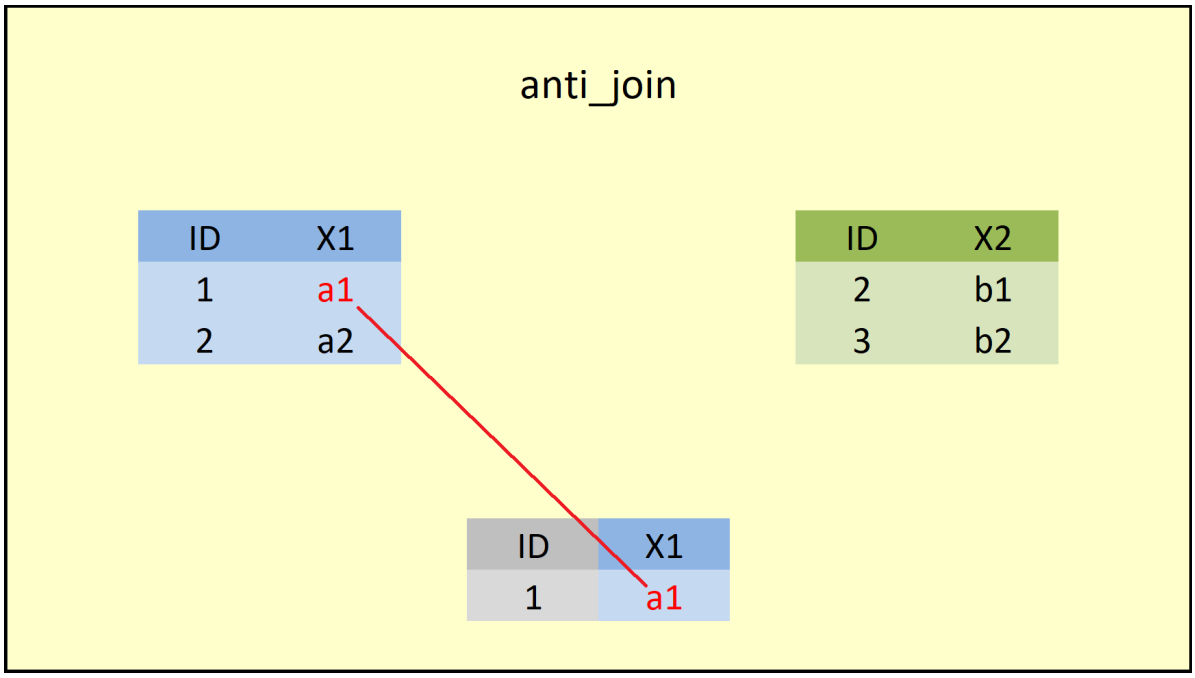
Semi join ve **Inner join** arasındaki temel fark, sonuç veri çerçevesinin içeriğidir. **Inner join**, her iki veri çerçevesinde ortak olan satırları birleştirir ve sonuç veri çerçevesine her iki veri çerçevesinin sütunlarını ekler. **Semi join** ise sadece birinci veri çerçevesindeki ortak satırları 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.

2.3.1.7 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.



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

2.3.2.1 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.

2.3.2.2 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.

2.4 Metin (String) Manipülasyonu

2.4.1 Temel Metin Manipülasyonu

2.4.1.1 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.
```

2.4.1.2 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.

2.4.1.3 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"
```

2.4.2 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
```

```
# 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.

2.4.2.1 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.

```
# 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"
```

```
# Tüm eşleşmeleri değiştirme
replaced_text2 <- str_replace_all(text2, "apple", "orange")
replaced_text2
```

```
[1] "red orange, green orange, red banana"
```

2.4.3 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

2.4.4 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
```

2.5 Fonksiyonlarla Veri Manipülasyonu

2.5.1 Fonksiyon Uygulamaları:

- `apply()`, `sapply()`, `lapply()` gibi fonksiyonlar ile veri üzerinde işlemler gerçekleştirme.

2.5.2 Gruplama İşlemleri:

- `tapply()` fonksiyonu ile gruplar üzerinde hesaplamalar yapma.

2.5.3 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

3 Veri Temizleme

3.1 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).

3.1.1 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.

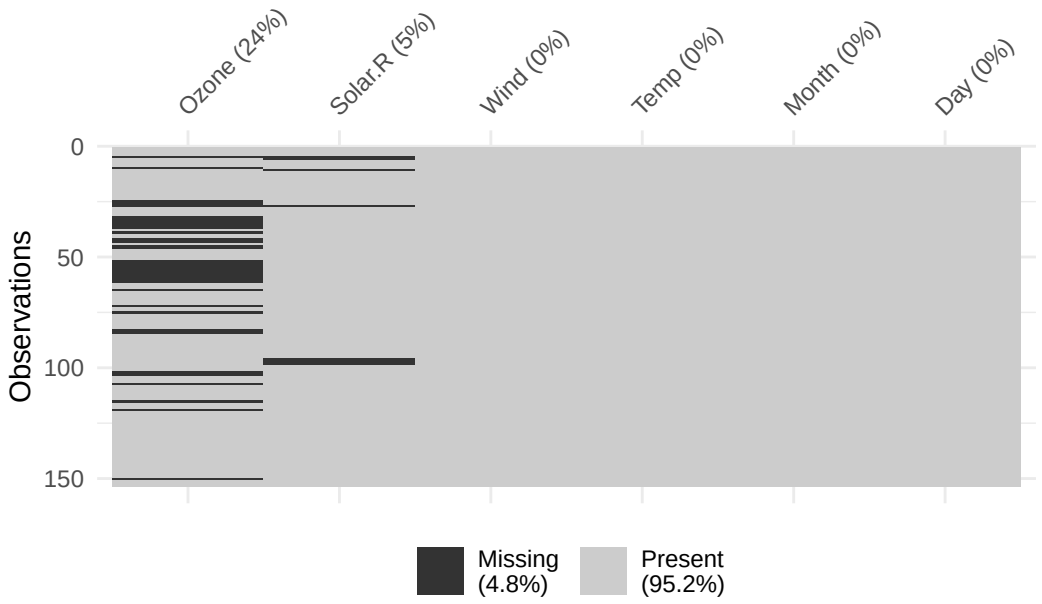
Bu nedenlerle, eksik verilerle başa çıkmak için **çoklu doldurma (multiple imputation)** veya

eksik değerlerin modelleme yöntemleriyle ele alınması gibi daha gelişmiş yöntemler genellikle listwise deletion'a tercih edilir.

3.1.2 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)
```



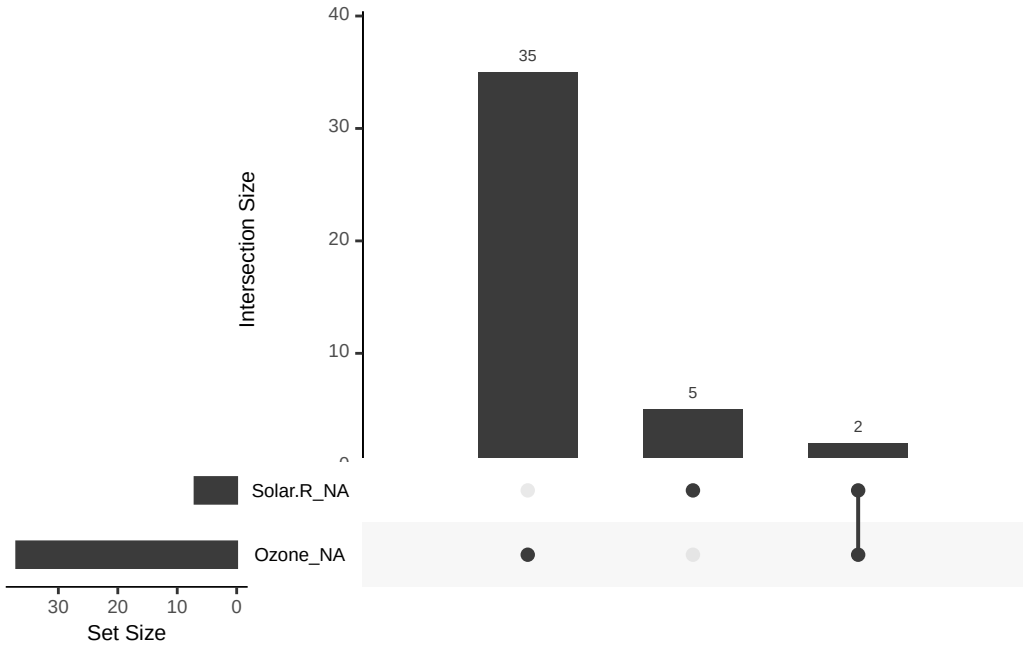
Grafik, `vis_miss()` fonksiyonu ile oluşturulmuş ve **airquality** veri setindeki eksik verilerin genel yapısını göstermektedir.

- **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.

3.1.2.1 Eksik Değerlerin Eşzamanlı Görülmesi

```
# naniar kütüphanesini yükleme
library(naniar)

# Eksik verilerin UpSet grafiği ile gösterimi
gg_miss_upset(airquality)
```



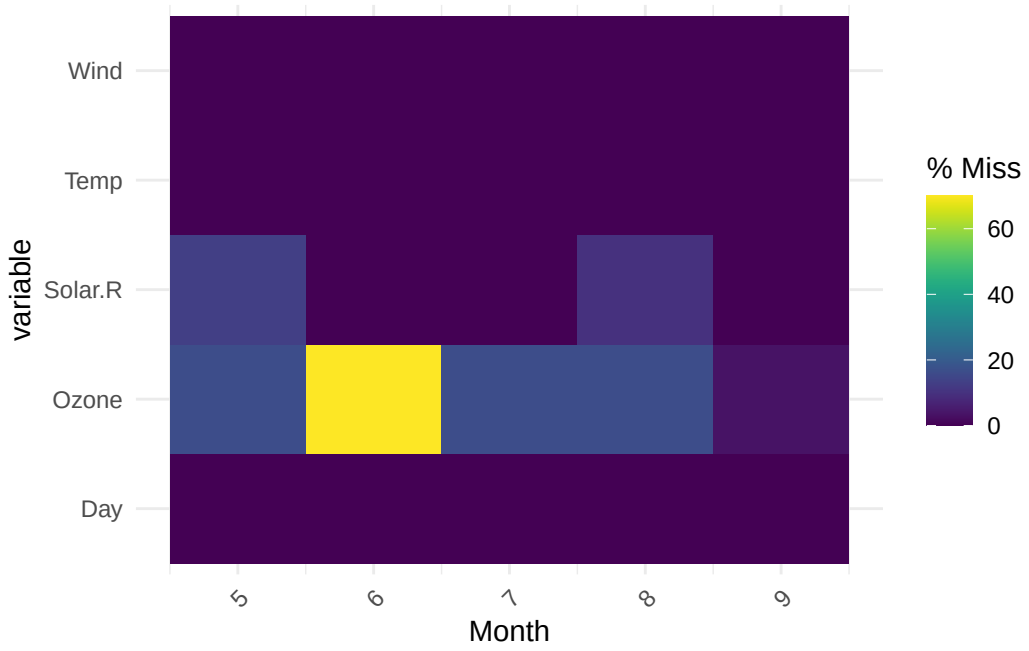
`gg_miss_upset(airquality)` fonksiyonu, `naniar` paketine ait bir fonksiyondur ve eksik değerlerin birlikteliğini (co-occurrence) görselleştirmek için UpSetR paketini kullanarak bir grafik oluşturur. Bu grafik, hangi değişkenlerin birlikte eksik olduğunu ve bu kombinasyonların ne sıklıkta görüldüğünü 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.

3.1.2.2 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:

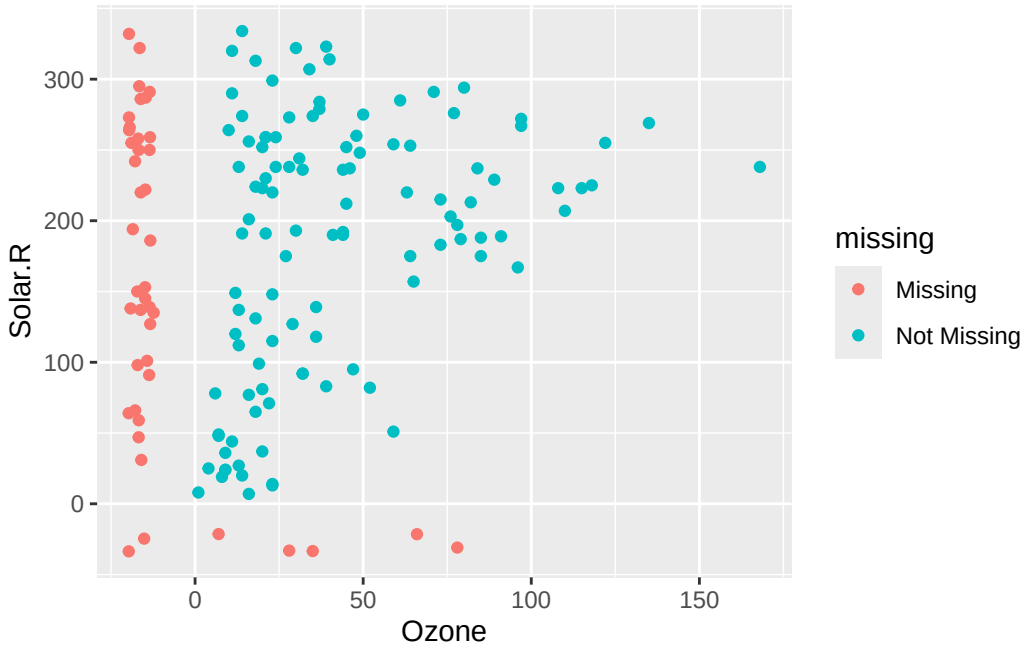
- **X ekseni (Month):** Ayları temsil eder (5 = Mayıs, 6 = Haziran, …, 9 = Eylül).
- **Y ekseni (Missing Percentage):** Eksik veri yüzdesini temsil eder.

- **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.

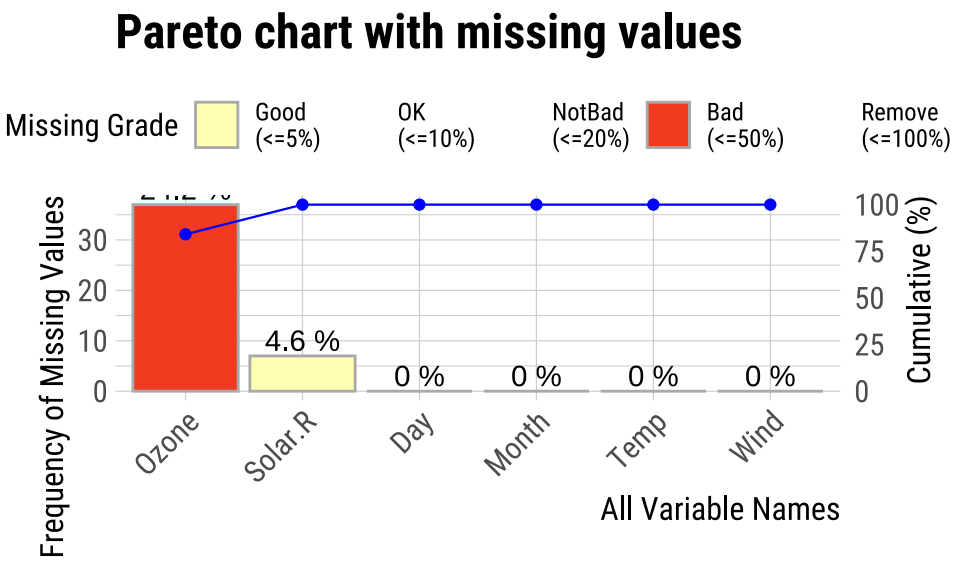
- **Eksik Verileri Faktör Düzeyine Göre Nokta Grafiği ile Görselleştirme**

```
# 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
```



3.1.2.3 `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.

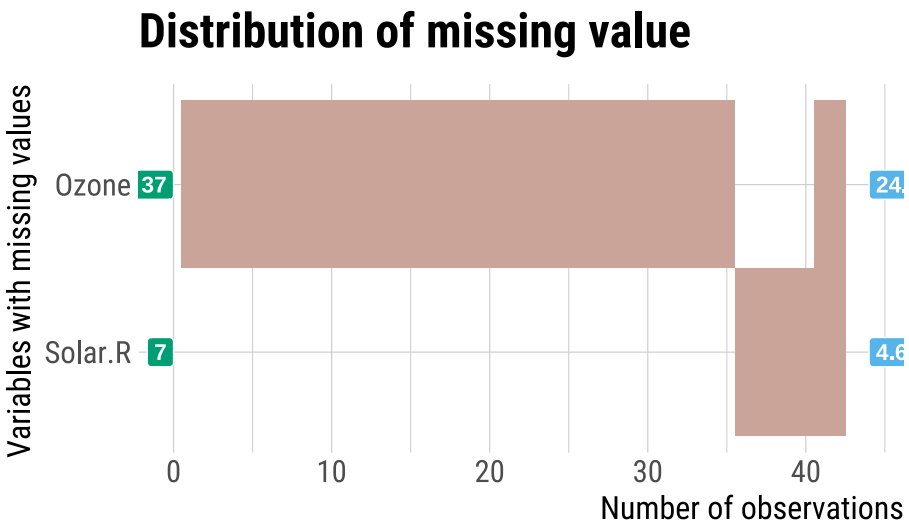
- **X ekseni:** Değişkenleri temsil eder. Değişkenler, eksik değer sayılarına göre en çoktan en aza doğru sıralanmıştır.
- **Sol Y ekseni:** Eksik değer sayısını temsil eder. Her bir değişken için bir çubuk bulunur ve çubuğun yüksekliği o değişkendeki eksik değer sayısını gösterir.
- **Sağ Y ekseni:** 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")
```

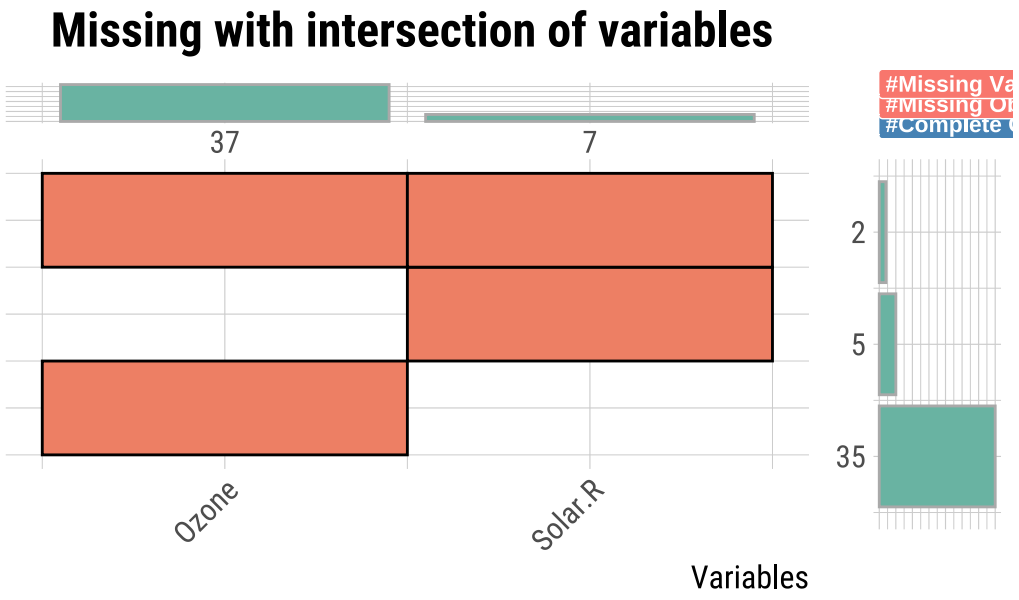


- `plot_na_hclust(airquality, main = "Distribution of missing value")` fonksiyonu, `dlookr` paketine ait bir fonksiyondur ve `airquality` veri setindeki eksik değer örüntülerini hiyerarşik kümeleme (hierarchical clustering) kullanarak görselleştirir.
- `main = "Distribution of missing value"` argümanı, 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.

3.1.3 Eksik Değerlerin Toplam Sayıları ve Oranları

`n_miss` fonksiyonu, verilerdeki tüm `NA` (yani eksik) değerlerinin toplam sayısını döndürür.

3.1.3.1 `NA` olan değerlerin sayısı için:

```
# 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.

3.1.3.2 `NA` olmayan (complete) değerlerin sayısı için:

```
# 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.

3.1.3.3 `NA` olan değerlerin oranı için:

```
# 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.

3.1.3.4 `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.

3.1.3.5 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.
```

3.1.4 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.

3.1.5 `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.

3.1.5.1 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.

3.1.5.2 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
```

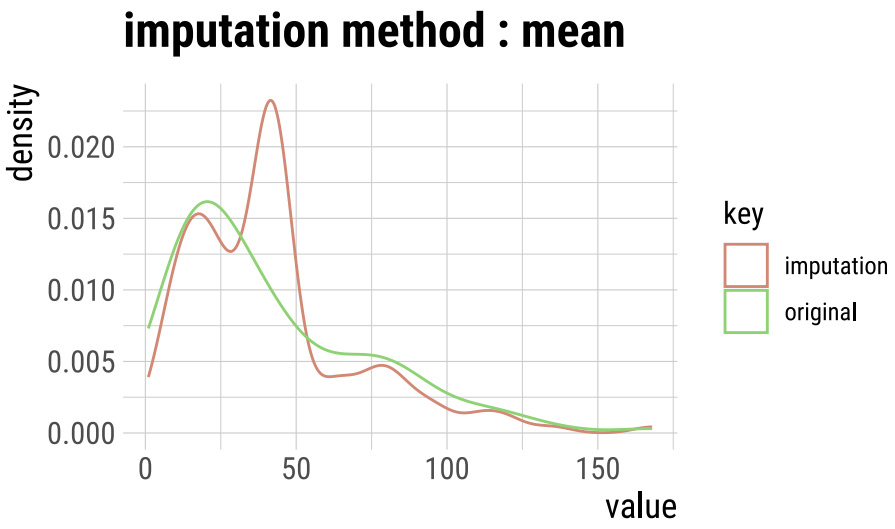
```
attr(,"var_type")
[1] "numerical"
attr(,"method")
[1] "mean"
attr(,"na_pos")
 [1]   5  10  25  26  27  32  33  34  35  36  37  39  42  43  45  46  52  53  54
[20]  55  56  57  58  59  60  61  65  72  75  83  84 102 103 107 115 119 150
attr(,"type")
[1] "missing values"
attr(,"message")
[1] "complete 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.

3.1.5.3 Ortalama (`mean`) ile doldurma öncesi ve sonrası yoğunluk dağılımları

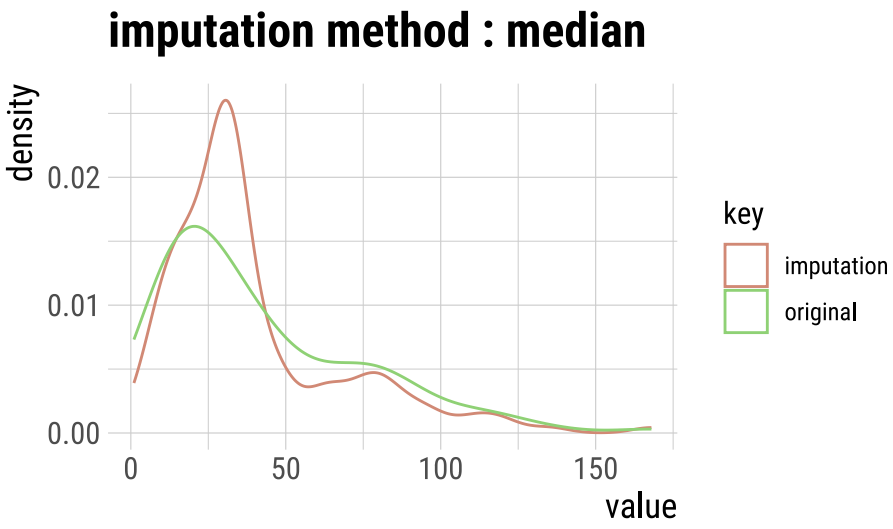
```
plot(aq_imp_ozone_mean)
```



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.

3.1.5.4 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)
```



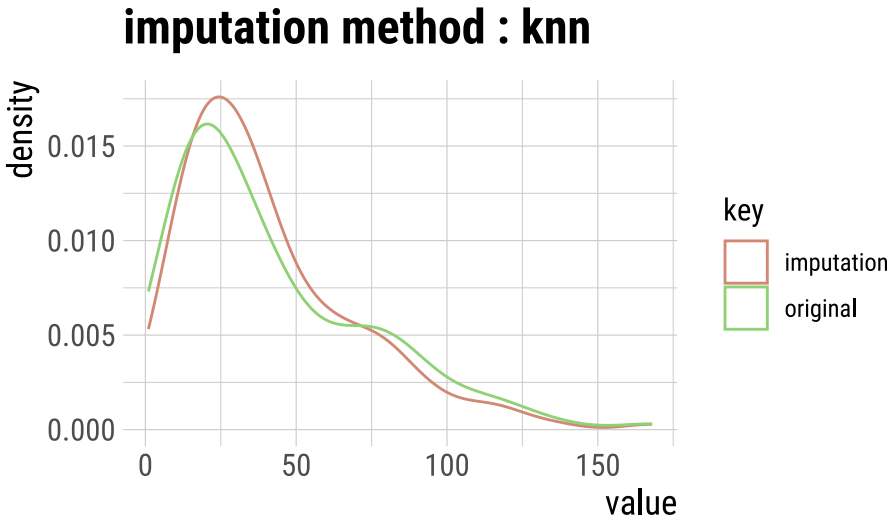
Grafikte, x ekseni vektördeki elemanların sırasını (indeks), y ekseni ise medyan ile doldurulmuş `Ozone` değerlerini gösterir. Ortalama ile doldurmaya benzer şekilde, grafikte noktaların rastgele yukarı aşağı hareket ettiğini görürsünüz. Ancak, medyan ile doldurmada, ortalama ile doldurmaya kıyasla grafikte 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.

3.1.5.5 `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)
```

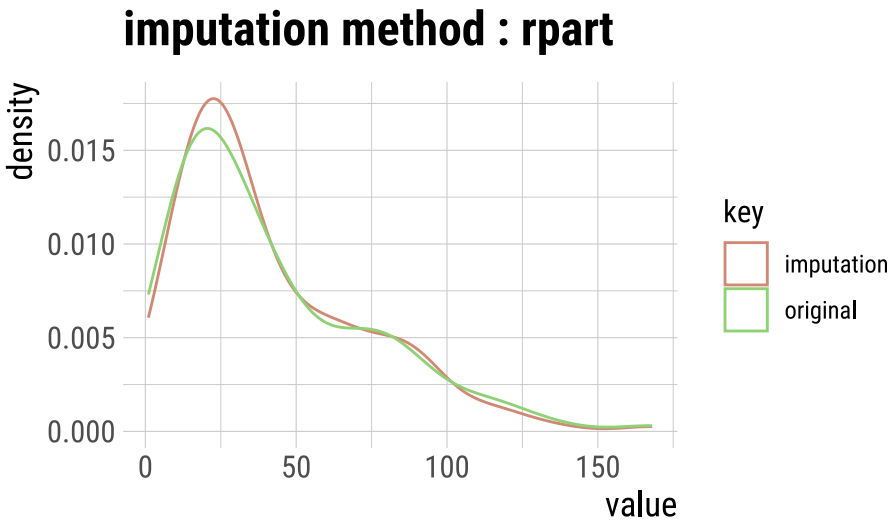


Yukarıdaki kod, `Ozone` değişkenindeki eksik değerleri ***knn*** **(k-Nearest Neighbors - k-En Yakın Komşu)** yöntemiyle dolduruyor ve ardından bu doldurulmuş değerleri `plot()` fonksiyonu ile çiziliyor.

- **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.

3.1.5.6 `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)
```

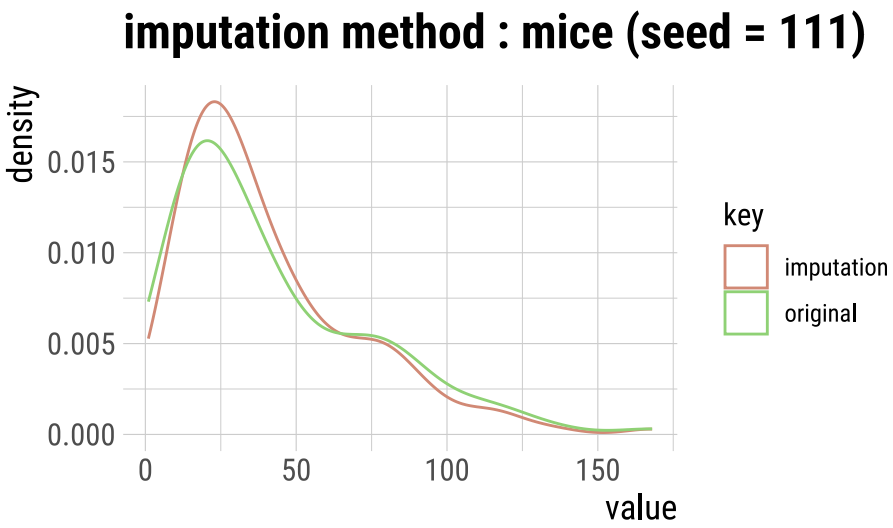


Yukarıdaki kod ile `Ozone` değişkenindeki eksik değerleri ***rpart*** **(Recursive Partitioning and Regression Trees - Özyinelemeli Bölümleme ve Regresyon Ağaçları)** yöntemiyle dolduruyor ve ardından bu doldurulmuş değerleri `plot()` fonksiyonu ile çiziyor.

- **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.

3.1.5.7 `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.

3.1.5.8 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
```

3.1.5.9 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
```

3.1.6 `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.

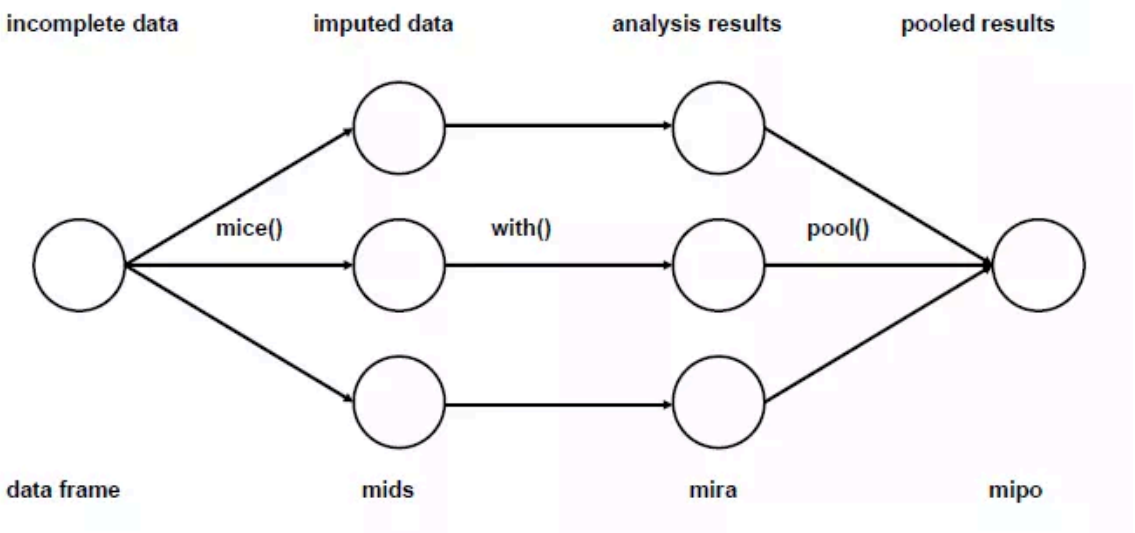
2. **Doldurulan veri setleri üzerinde analizler yapar (`with()`).**

   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.

3.1.6.1 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.

3.1.6.2 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.

> **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.

3.1.6.3 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.

3.1.6.4 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, 10)
```

```
   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    35.2  130.6    4.09        Yes
8    75.7  166.7    5.82        Yes
9    75.7  166.7    5.82        Yes
10   75.7  166.7    5.82        Yes
```

```
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
```

3.1.6.5 Ham Veri ile Son Veriyi Karşılaştırma

```
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
```

```
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
```

3.2 Aykırı Değerler ile Çalışma

3.2.1 Aykırı Değerleri Tanımlama ve İlk İnceleme

Aykırı değerler, veri analizinde istatistiksel çıkarımları bozabilecek uç değerlerdir. Bu değerleri tanımlamak ve analiz etmek için keşifçi veri analizi (EDA) kullanılmalıdır. Aşağıda, bir veri seti üzerinden aykırı değerleri tespit etmek için uygulanabilecek yöntemler sunulmuştur.

Örnek Veri Seti

Örnek olarak bir müşteri gelir veri seti oluşturalım:

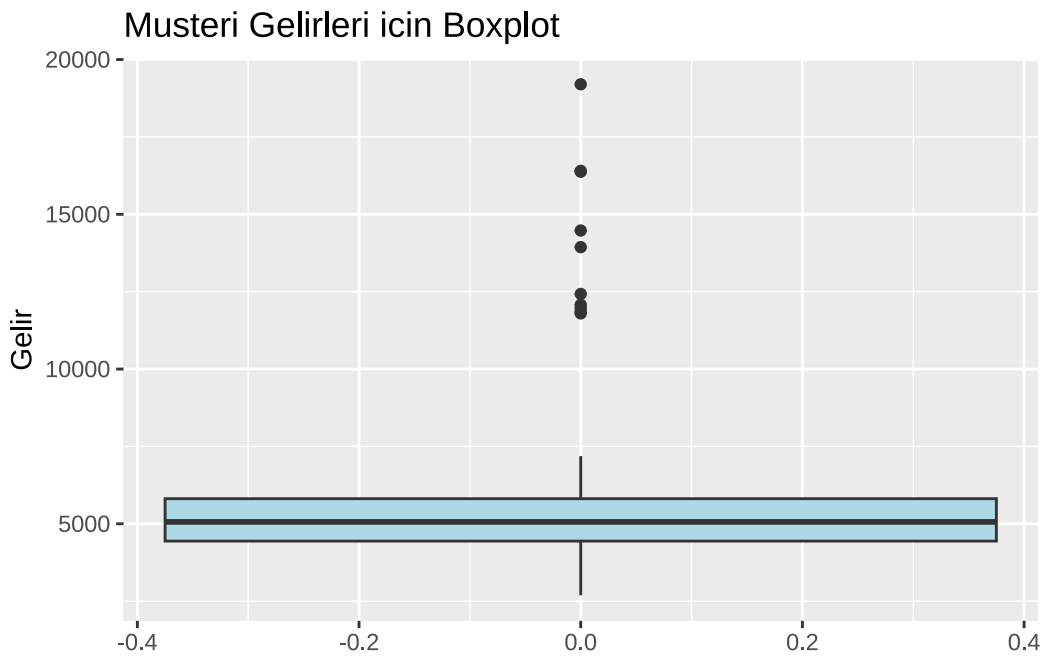
```
set.seed(123)
veri <- data.frame(
  musteri_id = 1:150,  # 150 müşteri için ID oluşturulur
  gelir = c(rnorm(140, mean = 5000, sd = 1000),
            # 140 müşteri için ortalama 5000 TL gelir
            rnorm(10, mean = 15000, sd = 2000))
            # 10 müşteri için ortalama 15000 TL gelir (Aykırı değerler)
)
```

Bu veri setinde 140 müşterinin geliri ortalama 5000 TL civarındayken, 10 müşterinin geliri 15000 TL ve üzerinde olarak ayarlanmıştır. Böylece, yüksek gelire sahip müşteriler aykırı değerler olarak tespit edilmelidir.

3.2.1.1 Boxplot ile Aykırı Değerleri Görselleştirme

Boxplot, aykırı değerleri hızlı bir şekilde görselleştirmek için kullanılır.

```
# ggplot2 kütüphanesini yükleme
library(ggplot2)

# Boxplot oluşturma
ggplot(veri, aes(y = gelir)) +  # 'veri' veri setinden 'gelir' değişkenini
                                #y eksenine yerleştir
  geom_boxplot(fill = "lightblue") +  # Boxplot çiz ve kutuyu açık mavi renk
  ggtitle("Musteri Gelirleri icin Boxplot") +  # Grafiğe başlık ekle
  ylab("Gelir")  # Y ekseninin adını "Gelir" olarak belirle
```

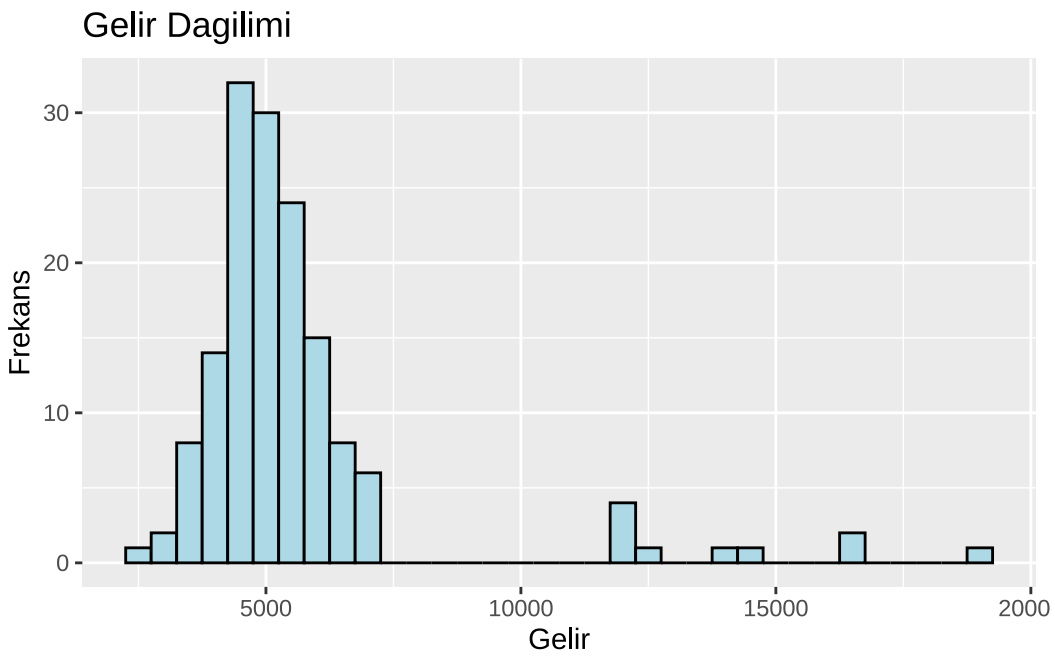


Boxplot'ta kutu dışındaki noktalar aykırı değerleri temsil eder. Üst ve alt sınırların dışında kalan noktalar potansiyel aykırı değerlerdir.

3.2.1.2 Histogram ile Dağılım Analizi

Histogram, veri setinin genel dağılımını incelemek için kullanılır. Aykırı değerler, histogramda ana dağılımın dışında kalan uç noktalarda yoğunlaşacaktır.

```
# Histogram oluşturma
ggplot(veri, aes(x = gelir)) +
  # 'veri' veri setinden 'gelir' değişkenini x eksenine yerleştir
  geom_histogram(binwidth = 500, fill = "lightblue", color = "black") +
  # Bin genişliği 500 olan histogram oluştur
  ggtitle("Gelir Dagilimi") +  # Grafiğe başlık ekle
  xlab("Gelir") +  # X ekseninin adını "Gelir" olarak belirle
  ylab("Frekans")  # Y ekseninin adını "Frekans" olarak belirle
```



Histogram, verinin çarpıklığını (skewness) gösterir. Eğer sağa veya sola çarpık bir dağılım varsa, bu genellikle aykırı değerlerin etkisiyle ortaya çıkar.

3.2.1.3 Özet İstatistiklerle İlk Değerlendirme

Aykırı değerleri tespit etmek için özet istatistiklerden de faydalanabiliriz:

```
# Gelir değişkeninin özet istatistiklerini hesaplama
summary(veri$gelir)
```

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
2691    4441    5062    5611    5811   19200
```

Bu çıktı, **minimum, maksimum, medyan ve çeyrek değerleri** içerir. Eğer **maksimum değer, üst çeyrek (Q3) değerinden çok daha büyükse**, aykırı değerlerin varlığına işaret edebilir.

Sonuç Yerine

- **Boxplot**, aykırı değerleri görselleştirmenin en hızlı yollarından biridir.
- **Histogram**, veri setinin genel dağılımını göstererek aşırı uç değerleri tespit etmeye yardımcı olur.

- **Özet istatistikler**, minimum ve maksimum değerleri analiz ederek aykırı değerleri sayısal olarak belirlemeye yardımcı olur.

Bu temel yöntemlerle aykırı değerleri belirledikten sonra, IQR, Z-Skoru veya Mahalanobis mesafesi gibi istatistiksel yöntemlerle daha detaylı bir analiz yapılabilir.

3.2.2 Tek Değişkenli Aykırı Değer Analizi

Tek değişkenli aykırı değer analizi, yalnızca **tek bir değişkenin dağılımına bakarak** uç değerlerin belirlenmesini sağlar. Aşağıda, farklı yöntemlerle aykırı değerleri tespit etmek için uygulanabilecek teknikler açıklanmaktadır.

Örnek Veri Seti Daha önce kullanılan müşteri gelir veri setini kullanmaya devam ediyoruz:

Bu veri setinde yüksek gelirli müşteriler aykırı değer olarak incelenecektir.

3.2.3 Aykırı Değer Tespiti ve Veri Setinden Çıkarılması (Tek Değişkenli)

3.2.3.1 Z-Skoru Yöntemi

Z-Skoru yöntemi, veri analistleri ve istatistikçiler tarafından özellikle normal dağılıma sahip veri setlerinde aykırı değerleri belirlemek için yaygın olarak kullanılır. Bir gözlemin ortalamadan kaç standart sapma uzaklıkta olduğunu hesaplayarak, ±3 standart sapma dışındaki değerleri aykırı kabul eder. Finans, kalite kontrol ve biyomedikal araştırmalar gibi alanlarda sıkça uygulanır. Ancak, normal dağılım varsayımına dayandığı için çarpık dağılımlarda yanıltıcı sonuçlar verebilir ve uç değerler standart sapmayı artırarak yöntemin duyarlılığını düşürebilir. Buna rağmen, özellikle simetrik dağılımlı veri setlerinde hızlı ve etkili bir aykırı değer tespit yöntemi olarak kullanılmaktadır.

Z-skoru yöntemi, bir gözlemin ortalamadan kaç standart sapma uzaklıkta olduğunu belirler. Eğer bir gözlem ±3 standart sapmadan daha uzaktaysa, bu gözlem aykırı değer olarak kabul edilir.

Z-Skoru Yönteminin Temeli Z-Skoru yöntemi, bir veri noktasının ortalamadan kaç standart sapma uzaklıkta olduğunu hesaplar.

Standart normal dağılımda (%99.7 kuralı - 3 sigma kuralı):

- Verilerin %68'i ortalama ±1 standart sapma içinde bulunur.
- Verilerin %95'i ortalama ±2 standart sapma içinde bulunur.
- Verilerin %99.7'si ortalama ±3 standart sapma içinde bulunur.
- Bu yüzden, Z-Skoru ±3'ü aşan gözlemler aykırı kabul edilir.

```r
# Gelir değişkeni için Z-Skoru yöntemi kullanılarak aykırı değerlerin tespiti

# 1. 'scale()' fonksiyonu ile gelir değişkeninin standartlaştırılması
# (Z-Skoru hesaplama)
z_scores <- scale(veri$gelir)

# 2. Z-Skoru mutlak değeri 3'ten büyük olan gözlemlerin aykırı değer olarak
# belirlenmesi
outliers_z <- veri$gelir[abs(z_scores) > 3]

# 3. Aykırı değerlerin çıktısını görüntüleme
outliers_z
```

```
[1] 16403.57 14475.61 13938.19 16375.83 19200.22
```

Neden Sadece 5 Aykırı Değer Belirlendi?

Z-Skoru yöntemi verinin normal dağıldığını varsayar.

Eğer veri seti normal dağılıma yakınsa, standart sapma istatistiği doğru şekilde çalışır ve aykırı değerleri doğru tespit edebilir. Ancak veri seti çarpık (skewed) bir dağılıma sahipse, standart sapma aşırı büyük veya küçük olabilir ve Z-Skoru bazı uç noktaları tespit edemeyebilir. Standart sapmanın büyüklüğü aykırı değer tespitini etkileyebilir.

Veri setinde büyük uç değerler varsa, standart sapma artar (çünkü standart sapma, aşırı uç değerlerden etkilenir). Standart sapma büyüdüğünde, ortalama **+3σ** eşiği de yükselir, dolayısıyla bazı yüksek değerler 3 standart sapma sınırını aşamayabilir. Bu durumda, normalde aykırı olması beklenen bazı yüksek gelirler, büyük bir standart sapma nedeniyle Z-Skoru yöntemi tarafından aykırı olarak kabul edilmeyebilir. Z-Skoru yöntemi simetrik dağılımlar için uygundur.

Eğer veri sağa çarpık (right-skewed) dağılım gösteriyorsa (yani büyük değerler normalden daha fazla bulunuyorsa), Z-Skoru yöntemi düşük hassasiyet gösterebilir. Çünkü Z-Skoru yöntemi, ortalamaya ve standart sapmaya bağımlıdır ve bu tür çarpık dağılımlarda ortalama üst tarafa kayar. Sonuç: Üst sınır daha yukarı kaydığı için bazı yüksek değerler aykırı olarak algılanmaz. Veri setinde doğal olarak geniş bir dağılım olabilir.

Örneğin, müşteri gelirleri genellikle çarpık bir dağılıma sahiptir ve bazı müşterilerin diğerlerine göre çok daha yüksek geliri olabilir. Eğer dağılım doğal olarak genişse, Z-Skoru yöntemi bu genişliği normal kabul ederek sadece en uçtaki birkaç değeri aykırı olarak seçebilir.

Sonuç

- Z-Skoru yöntemi 5 adet aykırı değer belirledi, çünkü veri seti muhtemelen normal dağılım göstermiyor ve sağa çarpık bir yapıya sahip.
- Standart sapma büyük olduğu için, bazı yüksek gelirler Z-Skoru için "ortalama civarında" kalmış olabilir.
- Eğer veri normal dağılsaydı, standart sapma daha küçük olurdu ve belki daha fazla gözlem aykırı olarak tespit edilirdi.
- Bu yüzden Z-Skoru yöntemi, normal dağılım varsayımına daha iyi uyan veri setlerinde daha doğru çalışır.

3.2.3.2 IQR (Çeyrekler Açıklığı) Yöntemi

IQR (Çeyrekler Açıklığı) yöntemi, veri analistleri ve istatistikçiler tarafından özellikle çarpık dağılımlara sahip veri setlerinde aykırı değerleri tespit etmek için sıkça kullanılır. Finans, sağlık ve makine öğrenmesi gibi alanlarda, uç değerleri belirlemek ve analizleri güvenilir hale getirmek amacıyla tercih edilir. Normal dağılım varsayımı gerektirmediği için Z-Skoru yöntemine kıyasla daha esnek bir yöntemdir. Ancak, belirlenen eşik değerlerin her veri seti için en uygun olmayabileceği unutulmamalıdır. Buna rağmen, veri temizleme ve istatistiksel analiz süreçlerinde yaygın olarak kullanılan sağlam (robust) bir yöntemdir.

IQR yöntemi, alt çeyrek (Q1) ve üst çeyrek (Q3) değerleri arasındaki farkı kullanarak aykırı değerleri belirler.

Bu yöntem, veri setinin merkezi dağılımına odaklanır ve uç değerleri belirlemek için alt (Q1) ve üst (Q3) çeyrekler aralığını kullanır.

- **Q1 (Alt Çeyrek - %25 Dilim):** Verinin %25'inin altında kaldığı değer.
- **Q3 (Üst Çeyrek - %75 Dilim):** Verinin %75'inin altında kaldığı değer.
- **IQR (Interquartile Range - Çeyrekler Açıklığı):** $IQR = Q3 - Q1$
  - Bu aralık, veri setinin orta %50'lik kısmını temsil eder.
  - Aykırı değerler, genellikle bu aralığın 1.5 katı kadar alt ve üst limitlerin dışına çıkan değerler olarak tanımlanır.

Aykırı değer sınırları şu şekilde belirlenir:

Alt Sınır $= Q1 - 1.5 \times IQR$ $Üst Sınır = Q3 + 1.5 \times IQR $

Bu sınırların dışına çıkan tüm gözlemler aykırı kabul edilir.

```r
# IQR (Çeyrekler Açıklığı) yöntemi ile aykırı değer tespiti

# 1. Gelir değişkeninin alt ve üst çeyreklerini hesapla
Q1 <- quantile(veri$gelir, 0.25)  # 1. çeyrek (Q1 - %25'lik dilim)
Q3 <- quantile(veri$gelir, 0.75)  # 3. çeyrek (Q3 - %75'lik dilim)

# 2. IQR hesapla (Q3 - Q1 farkı)
IQR <- Q3 - Q1  # Çeyrekler arası açıklık (Interquartile Range - IQR)

# 3. Aykırı değer eşiklerini belirleme
lower_bound <- Q1 - 1.5 * IQR  # Alt sınır
upper_bound <- Q3 + 1.5 * IQR  # Üst sınır

# 4. Aykırı değerleri tespit etme
outliers_iqr <- veri$gelir[veri$gelir < lower_bound | veri$gelir > upper_bound]

# 5. Aykırı değerleri görüntüleme
outliers_iqr
```

```
 [1] 16403.57 14475.61 11855.71 11970.66 11796.93 13938.19 12076.49 16375.83
 [9] 19200.22 12425.94
```

Neden Daha Fazla Aykırı Değer Bulundu?

Z-Skoru yöntemi sadece 5 değer tespit ederken, IQR yöntemi 10 değer tespit etti. Bunun sebebi nedir?

IQR yöntemi veri dağılımının şeklinden bağımsızdır.

- Z-Skoru, normal dağılıma bağımlıdır ve standart sapmayı kullanarak eşik belirler.
- IQR yöntemi ise sadece çeyrekler açıklığını dikkate alır.
- Bu nedenle, veri çarpık dağılıma sahipse veya standart sapması büyükse, IQR yöntemi daha fazla aykırı değer bulabilir.

IQR yöntemi ortalamadan etkilenmez, dağılımın orta kısmına odaklanır.

- Z-Skoru, verinin ortalamasına ve standart sapmasına bağlı olarak çalıştığı için bazı uç noktaları normal kabul edebilir.
- IQR yöntemi, verinin %50'lik merkezi kısmını esas aldığı için aşırı uçları daha kolay tespit eder.

Veri setinde sağa çarpık bir dağılım olabilir.

- Eğer veri sağa çarpık (right-skewed) ise, ortalama daha yukarı kayar ve Z-Skoru yöntemi bazı uç değerleri kaçırabilir.

- IQR yöntemi, böyle bir durumda yüksek gelire sahip daha fazla müşteriyi aykırı olarak değerlendirebilir.

Sonuç yerine

- IQR yöntemi, dağılımın orta %50'lik kısmını temel alarak uç değerleri belirler ve standart sapmadan etkilenmez.
- Z-Skoru yöntemi normal dağılım varsayar ve standart sapmaya bağlıdır, bu yüzden bazı uç değerleri normal kabul edebilir.
- Bu nedenle, IQR yöntemi Z-Skoru yöntemine göre daha fazla aykırı değer tespit etmiştir.
- Veri seti normal dağılmış olsaydı, Z-Skoru yöntemi de benzer sayıda aykırı değer bulabilirdi.
- Ancak mevcut durumda veri sağa çarpık (right-skewed) olduğu için IQR yöntemi daha hassas bir tespit yapmıştır.

IQR yöntemi, verinin merkezi dağılımını baz aldığı için özellikle sağa çarpık (asymmetric) veya geniş varyansa sahip veri setlerinde daha fazla aykırı değer tespit edebilir. Bu durumda, IQR yöntemi veriyi daha iyi temsil eden bir metot olarak değerlendirilmelidir.

3.2.3.3 MAD (Ortanca Mutlak Sapma) Yöntemi

MAD (Ortanca Mutlak Sapma) yöntemi, aykırı değerlere karşı dayanıklı (robust) bir istatistiksel yöntemdir ve özellikle çarpık dağılımlarda veya uç değerlerin yoğun olduğu veri setlerinde güvenilir sonuçlar sağlar. Medyan merkezli bir ölçüm olduğu için, ortalama ve standart sapmadan etkilenen Z-Skoru yöntemine kıyasla uç değerlerden daha az etkilenir. Finans, biyomedikal ve makine öğrenmesi gibi alanlarda, aşırı uç noktaların yanlış sınıflandırılmasını önlemek amacıyla sıkça kullanılır. Ancak, hesaplaması Z-Skoru ve IQR yöntemlerine göre daha karmaşıktır ve veri setinin dağılımına bağlı olarak eşik değerlerin dikkatle belirlenmesi gerekir. Dayanıklı yapısı nedeniyle, normal dağılım varsayımı gerektirmeyen veri setlerinde güvenilir bir aykırı değer tespit yöntemi olarak tercih edilir.

Ortanca Mutlak Sapma (MAD) yöntemi, medyanı referans alarak aykırı değerleri belirler ve aşırı uç gözlemlere karşı dayanıklıdır.

Aykırı değer sınırları: - **Alt sınır = Medyan - 3 * MAD - Üst sınır = Medyan + 3 * MAD**

```
# MAD (Ortanca Mutlak Sapma) yöntemi ile aykırı değer tespiti

# 1. Gelir değişkeninin MAD değerini hesapla
mad_value <- mad(veri$gelir)  # Ortanca mutlak sapma (MAD) hesaplanır

# 2. Aykırı değerleri belirleme kriteri:
# Bir gözlem, medyandan 3 * MAD kadar uzaksa aykırı kabul edilir
outliers_mad <- veri$gelir[abs(veri$gelir - median(veri$gelir)) > 3 * mad_value]
```

```
# 3. Aykırı değerleri görüntüleme
outliers_mad
```

```
 [1] 16403.57 14475.61 11855.71 11970.66 11796.93 13938.19 12076.49 16375.83
 [9] 19200.22 12425.94
```

IQR ve MAD Sonuçlarının Yorumu

1. **İki yöntem de aynı 10 aykırı değeri tespit etti.**

- Bu, veri setinin yapısının çarpık olmasına rağmen her iki yöntemin de güvenilir sonuçlar üretebileceğini gösteriyor.
- Eğer veri çok çarpık olsaydı, MAD yöntemi daha farklı bir sonuç verebilirdi.

2. **IQR yöntemi çeyrekler açıklığını (Q1 ve Q3) kullanırken, MAD yöntemi doğrudan medyan ve sapma ölçüsüne dayanır.**

- Eğer uç değerler çok fazlaysa, IQR yöntemi geniş bir dağılımda bazı noktaları kaçırabilir.
- MAD yöntemi, medyan etrafındaki mutlak sapmaları kullanarak daha geniş çerçevede uç değerleri değerlendirebilir.

3. **Çok uç değerler mevcut olsaydı, MAD yöntemi daha az hassas olabilirdi.**

- Çünkü MAD, medyana dayalı bir ölçeklendirme kullanır ve uç değerlerin yayılmasını daha küçük bir aralıkta tutabilir.
- IQR yöntemi ise uç değerler belirli bir aralığın dışına çıktığında daha geniş bir kapsama sahip olabilir.

Hangi Durumda Hangi Yöntem Kullanılmalı?

- Veri normal veya simetrik dağılıma sahipse: IQR yöntemi iyi bir seçim olabilir.
- Veri sağa/sola çarpık dağılıma sahipse veya çok uç değerler varsa: MAD yöntemi daha güvenilir olabilir.
- Eğer veri setinde hem çarpıklık hem de geniş varyasyon mevcutsa: Her iki yöntem birlikte kullanılabilir.
  - Bu durumda, IQR'nin belirlediği sınırları kontrol etmek için MAD yöntemi ek bir test olarak uygulanabilir.

Sonuç: IQR ve MAD Neden Aynı Sonuçları Verdi?

Bu veri setinde her iki yöntemin de aynı 10 gözlemi aykırı olarak belirlemesi, veri setinin yapısından kaynaklanmaktadır.

- Eğer veri daha çarpık olsaydı, MAD yöntemi bazı ek uç noktaları aykırı olarak belirleyebilir veya bazıları için daha dayanıklı kalabilirdi.
- Eğer veri daha homojen dağılıma sahip olsaydı, IQR yöntemi daha dar bir sınır belirleyebilir ve bazı uç noktaları dışarıda bırakabilirdi.

Sonuç olarak, veri setindeki çarpıklık düzeyi ve uç noktaların dağılımı bu iki yöntemin aynı sonucu vermesine neden olmuştur.
Eğer veri daha dengesiz bir dağılıma sahip olsaydı, bu iki yöntem arasında belirgin farklılıklar görülebilirdi.

MAD yöntemi, özellikle **veri çarpık dağılım gösterdiğinde veya aşırı uç değerler fazla olduğunda** daha güvenilir sonuçlar verir.

3.2.3.4 Yöntemlerin Karşılaştırılması

Durum	Önerilen Yöntem	R Uygulaması
Veri normal dağılmış mı?	Z-Skoru Yöntemi	`abs(scale(veri$deger)) > 3`
Veri normal dağılmamış mı?	IQR Yöntemi	`Q1 - 1.5*IQR, Q3 + 1.5*IQR`
Veri aşırı uç değerlere dayanıklı bir ölçüme mi ihtiyaç duyuyor?	MAD Yöntemi	`median(abs(veri$deger - median(veri$deger)))`

Her yöntemin **avantaj ve dezavantajları bulunur**, bu yüzden **veri setinin yapısına en uygun olan yöntem seçilmelidir.**

3.2.3.5 Genel Karşılaştırma Tablosu

Yöntem	Ne Zaman Kullanılır?	Avantajları	Dezavantajları
Z-Skoru	Veri normal dağılıyorsa	Kolay uygulanabilir, farklı ölçeklerde çalışır	Normal dağılım varsayımı gerektirir, uç değerlerden etkilenir
IQR (Çeyrekler Açıklığı)	Veri normal dağılmıyorsa	Aykırı değerlere dayanıklı, uygulanması kolay	Bazı önemli uç değerleri silebilir

Yöntem	Ne Zaman Kullanılır?	Avantajları	Dezavantajları
MAD (Ortanca Mutlak Sapma)	Aykırı değerlere dayanıklı analiz gerekiyorsa	Uç değerlere duyarlı değil, çarpık dağılımlar için uygun	Hesaplaması biraz daha karmaşıktır

Sonuç: Hangi Yöntemi Seçmeli?

Aykırı değer analizi yaparken hangi yöntemin seçileceği, veri setinin yapısına ve dağılım özelliklerine bağlıdır. Z-Skoru yöntemi, normal dağılıma sahip veri setlerinde hızlı bir analiz için uygundur. Ancak, gerçek dünya verilerinde normal dağılım her zaman sağlanamayacağı için, IQR ve MAD gibi uç değerlere dayanıklı (robust) yöntemler daha sık tercih edilmektedir.
IQR yöntemi, çarpık dağılım gösteren verilerde güvenilir bir seçimdir ve veri bilimciler tarafından en yaygın kullanılan yöntemlerden biridir. Bunun nedeni, hesaplamasının kolay olması, aşırı uç değerlerden etkilenmemesi ve dağılım varsayımı gerektirmemesidir. Özellikle finans, ekonomi, sağlık ve makine öğrenmesi gibi alanlarda, verilerin çoğu çarpık dağılım gösterdiğinden IQR yöntemi daha güvenilir bir seçim olarak öne çıkar.
MAD yöntemi ise, aykırı değerlere karşı en dayanıklı yöntem olup, aşırı uç verilerin fazla olduğu ve çarpıklığın aşırı yüksek olduğu durumlarda tercih edilir. Ancak, hesaplaması IQR'ye göre daha karmaşıktır ve belirli senaryolarda daha az hassas olabilir.
Öneri: Eğer veri setinin dağılımı bilinmiyorsa, önce histogram veya boxplot ile dağılım incelenmeli ve en uygun yöntem seçilmelidir. Genellikle IQR yöntemi, pratikliği ve güvenilirliği nedeniyle en sık tercih edilen aykırı değer tespit yöntemidir.
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.

3.2.4 Çok Değişkenli Aykırı Değer Analizi

Çok değişkenli aykırı değer analizi, bir gözlemin yalnızca tek bir değişken bazında değil, diğer değişkenlerle olan ilişkisine göre de aykırı olup olmadığını belirlemeyi amaçlar.

Ne zaman kullanılır?

- Eğer iki veya daha fazla değişken arasındaki ilişkilere göre aykırı değerleri tespit etmek istiyorsak.
- Eğer bir gözlem tek değişken bazında normal görünüyorsa ama diğer değişkenlerle birlikte değerlendirildiğinde aykırıysa.

Örnek Veri Seti

Aşağıda, ihracat hacmini etkileyen değişkenleri içeren bir veri seti (`data_multiple`) oluşturulmuştur.

```
set.seed(123)
library(tidyverse)


data_multiple <- data.frame(
  firma_id = 1:150,
  ihracat = c(pmax(rnorm(120, mean = 500000, sd = 100000), 0),
              pmax(rnorm(10, mean = 1000000, sd = 200000), 0),
              pmax(rnorm(20, mean = 100000, sd = 50000), 0)),
  devlet_yardimi = as_factor(sample(c(0, 1), 150, replace = TRUE,
                                    prob = c(0.7, 0.3))),
ciro = c(pmax(rnorm(130, mean = 2000000, sd = 500000), 0),
         pmax(rnorm(10, mean = 5000000, sd = 2000000), 0),
         pmax(rnorm(10, mean = 800000, sd = 100000), 0)),
ihracat_ulkesi = c(sample(10:50, 120, replace = TRUE),
                   sample(30:80, 20, replace = TRUE),
                   sample(1:10, 10, replace = TRUE)),
calisan_sayisi = c(round(pmax(rnorm(140, mean = 150, sd = 50), 5)),
                   round(pmax(rnorm(5, mean = 400, sd = 150), 5)),
                   round(pmax(rnorm(5, mean = 20, sd = 5), 5))),
  faaliyet_yili = c(sample(1:50, 130, replace = TRUE),
                    sample(50:100, 20, replace = TRUE))
)

head(data_multiple)
```

```
  firma_id  ihracat devlet_yardimi    ciro ihracat_ulkesi calisan_sayisi
1        1 443952.4              1 2029875             21            222
2        2 476982.3              0 1647702             20            202
3        3 655870.8              1 1641391             16            172
4        4 507050.8              1 2442325             12            186
5        5 512928.8              0 1492204             32            196
6        6 671506.5              0 2977647             30             17
  faaliyet_yili
1            43
2            45
3            25
4            36
```

```
5             25
6             50
```

Bu `data_multiple` setinde ihracat (bağımlı değişken) ve bağımsız değişkenler olarak devlet yardımı, ciro, ihracat yapılan ülke sayısı, çalışan sayısı ve faaliyet yılı bulunmaktadır.

```r
summary(data_multiple)
```

```
   firma_id         ihracat        devlet_yardimi      ciro
 Min.   :  1.00   Min.   :      0   0:106          Min.   :   68204
 1st Qu.: 38.25   1st Qu.: 407386   1: 44          1st Qu.: 1639746
 Median : 75.50   Median : 482660                  Median : 2011415
 Mean   : 75.50   Mean   : 478313                  Mean   : 2190169
 3rd Qu.:112.75   3rd Qu.: 563527                  3rd Qu.: 2439966
 Max.   :150.00   Max.   :1368772                  Max.   :10142916
 ihracat_ulkesi  calisan_sayisi  faaliyet_yili
 Min.   : 1.00   Min.   :  5.0   Min.   : 1.00
 1st Qu.:18.00   1st Qu.:120.2   1st Qu.:14.00
 Median :30.00   Median :154.0   Median :28.50
 Mean   :30.52   Mean   :156.6   Mean   :31.82
 3rd Qu.:40.00   3rd Qu.:187.8   3rd Qu.:43.00
 Max.   :78.00   Max.   :526.0   Max.   :98.00
```

- **İhracat (Bağımlı Değişken):** Büyük bir dağılım var, bazı firmaların ihracat değeri oldukça düşükken bazıları aşırı yüksek olabilir (potansiyel aykırı değerler var).
- **Devlet Yardımı (Kategori):** Çoğu firma devlet yardımı almıyor, ancak yaklaşık %30'u destek alıyor.
- **Ciro (Bağımsız Değişken):** Ciro ile ihracat arasında büyük farklar olabilir. Bazı firmaların cirosu aşırı yüksek (potansiyel aykırı değerler var).
- **İhracat Yapılan Ülke Sayısı:** Çoğu firma 30-40 ülkeye ihracat yapıyor, ancak 80 ülkeye ihracat yapan firmalar uç değer olabilir.
- **Çalışan Sayısı:** Çalışan sayısında büyük bir değişkenlik var, bazı firmalar küçük ölçekli iken bazıları oldukça büyük.
- **Faaliyet Yılı:** Firma yaşlarında ciddi bir farklılık var. Uzun süredir faaliyet gösteren firmalar ile yeni kurulan firmalar arasında ciddi fark olabilir.

3.2.5 Aykırı Değer Tespiti ve Veri Setinden Çıkarılması (Çok Değişkenli)

Aykırı değerleri tespit etmek için **Mahalanobis Mesafesi, İzolasyon Ormanı ve DBSCAN yöntemleri** kullanılabilir.

3.2.5.1 Mahalanobis Mesafesi

Mahalanobis mesafesi, çok değişkenli aykırı değerleri belirlemek için kullanılan istatistiksel bir yöntemdir. Bu yöntem, bir gözlemin tüm değişkenler açısından veri setinin merkezinden (ortalama vektöründen) ne kadar uzak olduğunu ölçer. Öklidyen mesafeye benzer ancak verinin kovaryans yapısını dikkate alarak ölçek bağımsız ve yön duyarlı bir mesafe ölçüsü sağlar.

Bu mesafe, bir gözlemin merkezden ne kadar farklı olduğunu ölçmek için kullanılır. Değeri ne kadar büyükse, gözlem diğerlerinden o kadar farklıdır ve aykırı olma olasılığı artar.

```
# Gerekli paketi yükleyelim (eğer yüklü değilse)
if (!requireNamespace("MASS", quietly = TRUE)) {
  install.packages("MASS")
}

# Gerekli kütüphaneyi çağıralım
library(MASS)  # Mahalanobis mesafesi hesaplaması için gerekli paket

# Gerekli değişkenleri seçelim
data_multiple_mahal <- data_multiple[, c("ihracat", "ciro", "ihracat_ulkesi",
                                         "calisan_sayisi", "faaliyet_yili")]

# Mahalanobis mesafesini hesaplayalım
mean_vector <- colMeans(data_multiple_mahal)
cov_matrix <- cov(data_multiple_mahal)
mahal_dist <- mahalanobis(data_multiple_mahal, mean_vector, cov_matrix)

# Kritik eşik belirleme ( ² testi, df = değişken sayısı, %95 güven aralığı)
thresh <- qchisq(0.95, df = ncol(data_multiple_mahal))

# Aykırı değerleri belirleyelim (eşik değerini aşan gözlemler aykırıdır)
data_multiple_mahal$aykiri_mahal <- mahal_dist > thresh

# Aykırı gözlemleri içeren veri setini oluşturalım
data_multiple_mahal_outlier <-
   data_multiple_mahal[data_multiple_mahal$aykiri_mahal == TRUE, ]

# Aykırı gözlemleri ana veri seti ile eşleştirerek devlet yardımı değişkenini ekleyelim
data_multiple_mahal_result <- data_multiple[data_multiple$firma_id
                                            %in% rownames(data_multiple_mahal_outlier), ]

# Yeni veri setini tibble formatında görüntüleyelim
as_tibble(data_multiple_mahal_result)
```

```
# A tibble: 21 x 7
   firma_id  ihracat devlet_yardimi      ciro ihracat_ulkesi calisan_sayisi
      <int>    <dbl> <fct>              <dbl>          <int>          <dbl>
 1      121 1023529. 0                1123381.             37            198
 2      123  901889. 0                1714075.             68            111
 3      125 1368772. 1                1910047.             73            128
 4      132  122575. 0                5822860.             76            145
 5      133  102062. 0                4933928.             64             91
 6      134   78875. 1                  68204.             37            175
 7      135       0  0               10142916.             68             98
 8      136  156567. 0                4589401.             78            139
 9      137   26968. 0                6302387.             51            169
10      139  195455. 0                7049346.             43            179
# i 11 more rows
# i 1 more variable: faaliyet_yili <int>
```

1. **Mahalanobis Mesafesi İçin Değişken Seçimi**

```
data_multiple_mahal <- data_multiple[, c("ihracat", "ciro", "ihracat_ulkesi",
"calisan_sayisi", "faaliyet_yili")]
```

- Bu değişkenler, çok değişkenli aykırı değer tespiti için neden seçildi?

Mahalanobis mesafesi, birden fazla değişken arasındaki ilişkileri dikkate alarak aykırı gözlemleri belirler. Bu nedenle, **doğrudan sayısal değer içeren** ve işletmenin büyüklüğü veya performansını gösteren değişkenler seçilmiştir.

Değişken	Açıklama	Neden Seçildi?
`ihracat`	Firmanın toplam ihracat miktarı	İşletmenin finansal büyüklüğünü gösterir
`ciro`	Firmanın toplam geliri (cirosu)	Genellikle ihracat ile ilişkilidir, finansal durumu gösterir
`ihracat_ulkesi`	Firmanın ihracat yaptığı ülke sayısı	İşletmenin uluslararası faaliyet kapsamını gösterir
`calisan_sayisi`	Firmanın çalışan sayısı	Firmanın ölçeği ve büyüklüğü hakkında bilgi verir
`faaliyet_yili`	Firmanın kaç yıldır faaliyette olduğu	İşletmenin deneyim seviyesi ve sürdürülebilirliği hakkında bilgi verir

- Neden `devlet_yardimi` Seçilmedi?
  - Devlet yardımı (0 veya 1) gibi kategorik değişkenler, Mahalanobis mesafesi hesaplamalarında anlamlı değildir.

- Mahalanobis mesafesi kovaryans matrisine dayandığı için sürekli (sayısal) değişkenler gereklidir.
- Eğer kategorik değişkenler kullanılmak istenirse, önce one-hot encoding (dummy değişkenler) gibi dönüşümler yapılmalıdır.

2. **Mahalanobis Mesafesi: Veri Noktalarının Merkezden Uzaklığını Ölçme**

- Ortalama Vektörü Hesaplanıyor

```
mean_vector <- colMeans(data_multiple_mahal)
```

  - Her değişkenin (sütunun) ortalama değeri hesaplanıyor.
  - Veri setinin merkez noktası belirleniyor.
  - Bu ortalama vektör, gözlemlerin merkeze ne kadar uzak olduğunu hesaplamak için kullanılacak.

- Kovaryans Matrisi Hesaplanıyor

```
cov_matrix <- cov(data_multiple_mahal)
```

  - Veri setindeki değişkenlerin birbirleriyle ilişkileri ölçülüyor.
  - Değişkenler arasındaki kovaryans hesaplanarak ölçek farkları dengeleniyor.
  - Öklidyen mesafeden farklı olarak, Mahalanobis mesafesi değişkenlerin kovaryans yapısını dikkate alıyor.

- Mahalanobis Mesafesi Hesaplanıyor

```
mahal_dist <- mahalanobis(data_multiple_mahal, mean_vector,
cov_matrix)
```

  - Her gözlemin (satırın) veri setinin merkezine (ortalama vektörüne) olan Mahalanobis mesafesi hesaplanıyor.
  - Bu mesafe ne kadar büyükse, gözlem veri setinin genel yapısından o kadar farklıdır ve aykırı olma ihtimali yüksektir.

3. **Mahalanobis Mesafesi İçin Ki-Kare Eşik Değeri Hesaplama**

Bu kod bloğunda, **Mahalanobis mesafesi için kritik eşik belirleniyor**. İşlemler adım adım şu şekilde gerçekleşiyor:

- Ki-kare Dağılımı Kullanılarak Eşik Değeri Hesaplanıyor

```
thresh <- qchisq(0.95, df = ncol(data_multiple_mahal))
```

  - `qchisq(0.95, df)` fonksiyonu, ki-kare dağılımının %95'lik yüzdelik dilimini hesaplar.
  - Eğer bir gözlemin Mahalanobis mesafesi bu eşik değerinden büyükse, aykırı kabul edilir.
  - %95 güven aralığı, gözlemlerin %5'inin aykırı kabul edileceğini gösterir.

- Serbestlik Derecesi (df) Belirleniyor

  ```
  df = ncol(data_multiple_mahal)
  ```

  - df (derece serbestlik) = Veri setindeki değişken (sütun) sayısıdır.
  - Her eklenen değişken, gözlemlerin çok değişkenli dağılımdaki konumunu etkiler.
  - Bu yüzden df, değişken sayısına eşit alınır.

Sonuçların Değerlendirilmesi:

Şu anki veri setimizde **18 tane aykırı değere sahip firma** tespit edildi ve devlet yardımı alan firmalardan 6 tanesi de aykırı olarak tespit edildi.

Bu durum analiz sonuçlarını etkileyebilir çünkü:

- Eğer bu firmalar aşırı yüksek ihracat yapan firmalar ise, devlet yardımı alan firmaların ihracatlarının genelde arttığına dair yanlış bir çıkarım yapabiliriz.
- Tam tersi, eğer bu firmalar aşırı düşük ihracat yapan firmalar ise, devlet yardımı alan firmaların başarısız olduğu gibi hatalı bir sonuç çıkabilir.
- Bu yüzden, devlet yardımı alan firmalar içindeki aykırı gözlemleri temizleyerek analiz yapmak daha doğru olacaktır.

Aykırı Olmayan Gözlemler (Mahalonobis)

```
# Aykırı olmayan gözlemleri filtreleyerek temiz bir veri seti oluşturma
mahal_data_clean <- data_multiple[!data_multiple$firma_id %in% rownames(data_multiple_mahal_

# Yeni veri setini tibble formatında görüntüleyelim
as_tibble(mahal_data_clean)
```

```
# A tibble: 129 x 7
   firma_id ihracat devlet_yardimi     ciro ihracat_ulkesi calisan_sayisi
      <int>   <dbl> <fct>             <dbl>          <int>          <dbl>
 1        1 443952. 1              2029875.             21            222
 2        2 476982. 0              1647702.             20            202
 3        3 655871. 1              1641391.             16            172
 4        4 507051. 1              2442325.             12            186
 5        5 512929. 0              1492204.             32            196
 6        6 671506. 0              2977647.             30             17
 7        7 546092. 0              1954840.             30            206
 8        8 373494. 0              2107269.             46            126
 9        9 431315. 0              1630736.             11            162
10       10 455434. 0              1712806.             18            135
# i 119 more rows
```

```
# i 1 more variable: faaliyet_yili <int>
```

Sonuç: Mahalanobis Mesafesi Kullanmalı mıyım?

Avantajları	Dezavantajları
Çok değişkenli aykırı değerleri belirler.	Veri normal dağılmıyorsa yanlış sonuçlar verebilir.
Ölçek bağımsızdır, farklı ölçeklerdeki değişkenleri dengeler.	Çok değişkenli normal dağılım varsayımı yapar.
Değişkenler arasındaki ilişkileri dikkate alır.	Kovaryans matrisi tekillik sorunu yaşayabilir.
Hızlı hesaplanabilir ve yorumlanabilir.	Büyük veri setlerinde hesaplama maliyeti artabilir.

Öneri:

Eğer veri normal dağılıma yakınsa ve doğrusal ilişkiler varsa, Mahalanobis mesafesi güvenilir bir yöntemdir. Ancak, veri çarpık dağılım gösteriyorsa veya çok karmaşıksa, İzolasyon Ormanı (Isolation Forest) veya DBSCAN gibi yöntemler daha uygun olabilir.

Uygunluk Değerlendirmesi:

Mahalanobis Mesafesi, eğer veri normal dağılıma yakınsa ve değişkenler arasındaki ilişki doğrusal ise oldukça etkili olabilir. Ancak, bizim veri setimizde değişkenlerin dağılımı normal olmayabilir ve ciro, ihracat gibi değişkenler oldukça değişkenlik gösterebilir. Bu durumda Mahalanobis Mesafesi yanıltıcı sonuçlar verebilir.

3.2.5.2 İzolasyon Ormanı (Isolation Forest)

Isolation Forest, aykırı değerleri tespit etmek için karar ağaçlarına dayalı bir makine öğrenmesi yöntemidir. Bu yöntem, rastgele alt kümeler alarak veri noktalarını izole eder ve bir gözlemin ne kadar hızlı izole edilebildiğini ölçer. Aykırı gözlemler, diğer gözlemlerden daha az bölme (izolasyon) gerektirdiği için, daha düşük derinlikte daha erken izole edilir.

Bu yöntem, verinin dağılımına bağımlı değildir ve büyük veri setlerinde ölçeklenebilir bir yapı sunar. Aykırılık skoru 0 ile 1 arasında değişir ve 1'e yakın değerler gözlemin aykırı olma olasılığını artırır. Özellikle karmaşık veri setlerinde, doğrusal olmayan ilişkileri tespit etmek için kullanışlıdır.

```
# Gerekli değişkenleri seçelim
data_multiple_iso <- data_multiple[, c("ihracat", "ciro", "ihracat_ulkesi",
                                       "calisan_sayisi", "faaliyet_yili")]

# Gerekli paketi yükleyelim (eğer yüklü değilse)
```

```r
if (!requireNamespace("isotree", quietly = TRUE)) {
  install.packages("isotree")
   }

# Gerekli kütüphaneyi çağıralım
library(isotree)  # Isolation Forest modeli için gerekli paket

# Isolation Forest modelini oluşturalım
iso_model <- isotree::isolation.forest(data_multiple_iso, ntrees = 100)

# Aykırılık skorlarını tahmin edelim
outlier_scores <- predict(iso_model, data_multiple_iso)

# Aykırı gözlemleri belirleyelim (skoru 0.6'dan büyük olanlar aykırıdır)
data_multiple_iso$aykiri_iso <- outlier_scores > 0.6

# Aykırı gözlemleri içeren veri setini oluşturalım
data_multiple_iso_outlier <-
   data_multiple_iso[data_multiple_iso$aykiri_iso == TRUE, ]

# Aykırı gözlemleri ana veri seti ile eşleştirerek devlet yardımı
# değişkenini ekleyelim
data_multiple_iso_result <- data_multiple[data_multiple$firma_id %in%
                                          rownames(data_multiple_iso_outlier), ]

# Yeni veri setini tibble formatında görüntüleyelim
as_tibble(data_multiple_iso_result)
```

```
# A tibble: 5 x 7
  firma_id  ihracat devlet_yardimi      ciro ihracat_ulkesi calisan_sayisi
     <int>    <dbl> <fct>              <dbl>          <int>          <dbl>
1      125 1368772. 1               1910047.             73            128
2      135       0  0               10142916.            68             98
3      139  195455. 0                7049346.            43            179
4      143   21393. 1                 705459.             7            526
5      145   19923. 1                 753896.             9            476
# i 1 more variable: faaliyet_yili <int>
```

- **Aykırı Değer Tespiti İçin Gerekli Değişkenlerin Seçilmesi**

  ```r
  data_multiple_iso <- data_multiple[, c("ihracat", "ciro",
  "ihracat_ulkesi", "calisan_sayisi", "faaliyet_yili")]
  ```

  - Analizde kullanılacak sayısal değişkenler seçiliyor.
  - Kategorik değişkenler dahil edilmeden sadece sayısal değişkenlerle işlem yapılıyor.

- **Isolation Forest Modelinin Kurulması ve Aykırı Değerlerin Belirlenmesi**

  `library(isotree) iso_model <- isotree::isolation.forest(data_multiple_iso, ntrees = 100)`

  - Isolation Forest modeli oluşturuluyor.
  - 100 ağaç (ntrees = 100) kullanılarak model eğitiliyor.

- `outlier_scores <- predict(iso_model, data_multiple_iso) data_multiple_iso$aykiri_iso <- outlier_scores > 0.6  # Skoru 0.6'dan büyük olanlar aykırı`

  - Model, her gözlem için bir "aykırılık skoru" hesaplıyor.
  - Skoru 0.6'dan büyük olan gözlemler aykırı olarak kabul ediliyor.

i Ağaç sayısı neye göre belirlenir

Isolation Forest modelinde `ntrees = 100` genellikle hız ve doğruluk arasında iyi bir denge sağladığı için tercih edilir. Her bir ağaç, veri setinden rastgele örnekler alarak gözlemleri izole etmeye çalışır ve bu süreç ne kadar az bölme ile gerçekleşirse, gözlem o kadar aykırı kabul edilir. Daha fazla ağaç kullanmak, modelin daha tutarlı ve stabil sonuçlar vermesini sağlar çünkü her ağaç farklı alt kümelerle eğitildiğinden, tek bir ağacın etkisi azalır ve genel eğilim daha güvenilir hale gelir. Ancak, ağaç sayısının fazla olması işlem süresini artırırken, çok az olması modelin kararsız olmasına yol açabilir. Küçük veri setlerinde `ntrees = 50` genellikle yeterli olurken, orta ölçekli veri setlerinde `ntrees = 100 - 200`, büyük veri setlerinde ise `ntrees = 200+` kullanmak daha iyi sonuç verir. Eğer hız ön planda ise daha az ağaç kullanılabilir, ancak daha hassas aykırı değer tespiti yapılmak isteniyorsa ağaç sayısının artırılması önerilir. Bu nedenle, 100 ağaç genellikle güvenilir ve dengeli bir seçim olarak kabul edilir.

i 0,6 Ne Anlama Geliyor ve Neden Seçildi

Isolation Forest modelinde `predict()` fonksiyonu, her gözlem için bir "aykırılık skoru" (`outlier_scores`) üretir. Bu skor, bir gözlemin model tarafından ne kadar hızlı izole edilebildiğini gösterir. Skor değeri 0 ile 1 arasında değişir ve şu şekilde yorumlanır:

- 0'a yakın skorlar → Normal gözlemler (izole edilmesi zor, yani veri kümesine iyi uyuyor).
- 1'e yakın skorlar → Aykırı gözlemler (çok hızlı izole edilebiliyor, yani genel dağılımdan çok farklı).

Bu şu anlama gelir: Eğer bir gözlemin Isolation Forest tarafından hesaplanan skoru `0.6`'dan büyükse, bu gözlem aykırı kabul edilir.
Neden `0.6` Seçildi? Daha Fazla veya Daha Az Olamaz mıydı?

> Genellikle `0.6`, normal gözlemler ile aykırı gözlemleri ayırt etmek için dengeli bir eşik değeridir. Ancak, analiz yapılan veri setine bağlı olarak bu değer değiştirilebilir. Eğer veri setinde çok fazla aykırı değer olduğu düşünülüyorsa, eşik değeri artırılabilir (0.7 veya 0.8 yapılabilir). Eğer daha fazla gözlem aykırı kabul edilmek isteniyorsa, eşik değeri düşürülebilir (0.5 veya 0.55 yapılabilir).

- **Aykırı Olan Gözlemlerin Seçilmesi**

  ```
  data_multiple_iso_outlier <- data_multiple_iso[data_multiple_iso$aykiri_iso
  == TRUE, ]
  ```

  - Aykırı olarak belirlenen gözlemler ayrı bir veri setine kaydediliyor.

- **Aykırı Gözlemleri Ana Veri Seti ile Eşleştirerek Devlet Yardımı Bilgisinin Eklenmesi**

  ```
  data_multiple_iso_result <- data_multiple[data_multiple$firma_id
  %in% rownames(data_multiple_iso_outlier), ] as_tibble(data_multiple_iso_result)
  ```

  - Aykırı gözlemlerin `firma_id`'si kullanılarak ana veri setindeki devlet yardımı değişkeniyle eşleştirilmesi yapılıyor.
  - Böylece devlet yardımı alan firmalar arasında aykırı olanlar analiz edilebilir.

Sonuçların Değerlendirilmesi:

Şu anki veri setimizde 9 tane aykırı değere sahip firma tespit edildi ve bu firmalardan 3 tanesi devlet yardımı almış firmalar arasındadır.

Bu durum analiz sonuçlarını etkileyebilir çünkü:

- Eğer bu firmalar aşırı yüksek ihracat yapan firmalar ise, devlet yardımı alan firmaların ihracatlarının genelde arttığına dair yanlış bir çıkarım yapabiliriz.
- Tam tersi, eğer bu firmalar aşırı düşük ihracat yapan firmalar ise, devlet yardımı alan firmaların başarısız olduğu gibi hatalı bir sonuç çıkabilir.
- Bazı firmalar hiç ihracat yapmadığı halde model tarafından aykırı olarak belirlenmiştir, bu da sonuçları yanıltabilir.

Bu yüzden, devlet yardımı alan firmalar içindeki aykırı gözlemleri temizleyerek analiz yapmak daha doğru olacaktır. Böylece, aykırı gözlemlerden kaynaklanan sapmalar engellenerek, devlet desteğinin ihracat üzerindeki etkisi daha sağlıklı bir şekilde ölçülebilir.

Aykırı Olmayan Gözlemler (Isolation Forest)

```
# Aykırı olmayan gözlemleri filtreleyerek temiz bir veri seti oluşturma
iso_data_clean <- data_multiple[!data_multiple$firma_id
                                %in% rownames(data_multiple_iso_outlier), ]
```

```r
# Yeni veri setini tibble formatında görüntüleyelim
as_tibble(iso_data_clean)
```

```
# A tibble: 145 x 7
   firma_id ihracat devlet_yardimi     ciro ihracat_ulkesi calisan_sayisi
      <int>   <dbl> <fct>             <dbl>          <int>          <dbl>
 1        1 443952. 1               2029875.             21            222
 2        2 476982. 0               1647702.             20            202
 3        3 655871. 1               1641391.             16            172
 4        4 507051. 1               2442325.             12            186
 5        5 512929. 0               1492204.             32            196
 6        6 671506. 0               2977647.             30             17
 7        7 546092. 0               1954840.             30            206
 8        8 373494. 0               2107269.             46            126
 9        9 431315. 0               1630736.             11            162
10       10 455434. 0               1712806.             18            135
# i 135 more rows
# i 1 more variable: faaliyet_yili <int>
```

Sonuç: Isolation Forest Kullanmalı mıyım?

Avantajları	Dezavantajları
Aykırı değerleri belirlemek için dağılım varsayımına ihtiyaç duymaz, veri normal dağılmak zorunda değildir.	Hiperparametre ayarlaması gerektirir, özellikle ağaç sayısı (`ntrees`) ve eşik değeri (`threshold`) analiz sonuçlarını etkileyebilir.
Karmaşık ve doğrusal olmayan ilişkileri tespit edebilir.	Aykırı değerlerin neden aykırı olduğunu doğrudan açıklamaz, yalnızca skorlara dayalı karar verir.
Büyük veri setlerinde ölçeklenebilir, veri hacmi arttığında bile verimli çalışır.	Çok küçük veri setlerinde güvenilirliği azalabilir, az sayıda gözlem olduğunda daha fazla hata yapabilir.
Hızlı çalışır ve hesaplama maliyeti düşüktür, yüksek boyutlu veri setlerinde de uygulanabilir.	Nihai sonuçlar modelin eğitimine bağlıdır, farklı çalıştırmalarda küçük farklılıklar olabilir.

Öneri:

Eğer veri normal dağılmıyorsa, değişkenler arasındaki ilişki karmaşıksa ve büyük veri setleriyle çalışıyorsanız, Isolation Forest güçlü bir alternatiftir. Özellikle yoğunluk bazlı veya doğrusal olmayan aykırı değerleri yakalamak için Mahalanobis mesafesinden daha etkili olabilir.

Uygunluk Değerlendirmesi:

Isolation Forest bizim veri setimiz için oldukça uygundur. Firmalar arasındaki büyüklük farklılıklarını ve doğrusal olmayan ilişkileri yakalayabilir. Ayrıca verinin normal dağılım gösterip göstermediği önemli olmadığı için, değişkenler arasındaki ilişkilere dayanarak iyi bir aykırı değer analizi yapabilir. Ancak, eşik değeri (`threshold`) dikkatli seçilmeli ve sonuçlar diğer yöntemlerle karşılaştırılmalıdır.

3.2.5.3 DBSCAN (Yoğunluk Bazlı Kümeleme)

DBSCAN, yoğunluk tabanlı bir kümeleme algoritmasıdır ve yoğunluk düşük alanlarda kalan gözlemleri aykırı olarak kabul eder. Bu yöntem, veri noktalarının yoğunluklarını ölçerek kümeler oluşturur ve belirlenen bir `eps` mesafesi içinde en az `minPts` kadar komşuya sahip olmayan noktaları aykırı değer olarak işaretler.

DBSCAN, önceden belirlenmiş küme sayısına ihtiyaç duymadan çalışır ve karmaşık şekilli kümeleri tespit edebilir. Ancak, veri setine uygun `eps` ve `minPts` değerlerinin dikkatli seçilmesi gereklidir. Küme numarası 0 olan gözlemler aykırı olarak kabul edilir ve bu gözlemler veri setinin genel yoğunluğundan farklı alanlarda bulunur.

```r
# Gerekli değişkenleri seçelim
data_multiple_dbscan <- data_multiple[, c("ihracat", "ciro", "ihracat_ulkesi",
                                          "calisan_sayisi", "faaliyet_yili")]

# Gerekli paketi yükleyelim (eğer yüklü değilse)
if (!requireNamespace("dbscan", quietly = TRUE)) {
  install.packages("dbscan")
}

# Gerekli kütüphaneyi çağıralım
library(dbscan)  # DBSCAN kümeleme yöntemi için gerekli paket

# Veriyi ölçekleyelim
scaled_data <- scale(data_multiple_dbscan)

# DBSCAN modelini oluşturalım
db_model <- dbscan::dbscan(scaled_data, eps = 1, minPts = 5)

# Aykırı gözlemleri belirleyelim (küme numarası 0 olanlar aykırıdır)
data_multiple_dbscan$aykiri_dbscan <- db_model$cluster == 0

# Aykırı gözlemleri içeren veri setini oluşturalım
data_multiple_dbscan_outlier <-
   data_multiple_dbscan[data_multiple_dbscan$aykiri_dbscan == TRUE, ]
```

```r
# Aykırı gözlemleri ana veri seti ile eşleştirerek
# devlet yardımı değişkenini ekleyelim
data_multiple_dbscan_result <-
   data_multiple[data_multiple$firma_id %in% rownames(data_multiple_dbscan_outlier), ]

# Yeni veri setini tibble formatında görüntüleyelim
as_tibble(data_multiple_dbscan_result)
```

```
# A tibble: 26 x 7
   firma_id  ihracat devlet_yardimi     ciro ihracat_ulkesi calisan_sayisi
      <int>    <dbl> <fct>             <dbl>          <int>          <dbl>
 1        6  671506. 0              2977647.             30             17
 2       72  269083. 0              2933426.             22            145
 3      121 1023529. 0              1123381.             37            198
 4      122  810505. 0              2049664.             38             77
 5      123  901889. 0              1714075.             68            111
 6      124  948782. 0              1512995.             56            166
 7      125 1368772. 1              1910047.             73            128
 8      126  869610. 0              2507472.             64            219
 9      127 1047077. 0              1003626.             30            184
10      128 1015592. 0              1786360.             65            154
# i 16 more rows
# i 1 more variable: faaliyet_yili <int>
```

- **Aykırı Değer Tespiti İçin Gerekli Değişkenler Seçiliyor**

  ```r
  data_multiple_dbscan <- data_multiple[, c("ihracat", "ciro",
  "ihracat_ulkesi", "calisan_sayisi", "faaliyet_yili")]
  ```

  - Analizde kullanılacak sayısal değişkenler seçiliyor.
  - Kategorik değişkenler dahil edilmeden sadece sayısal değişkenler kullanılıyor.

- **Veri Ölçekleniyor**

  ```r
  scaled_data <- scale(data_multiple_dbscan)
  ```

  - DBSCAN yöntemi, değişkenlerin farklı ölçeklere sahip olması durumunda doğru çalışmayabilir.
  - Bu yüzden, tüm değişkenler `scale()` fonksiyonu ile standartlaştırılıyor (ortalama = 0, standart sapma = 1 olacak şekilde dönüştürülüyor).

- **DBSCAN Modeli Oluşturuluyor ve Aykırı Gözlemler Belirleniyor**

  ```r
  db_model <- dbscan::dbscan(scaled_data, eps = 1, minPts = 5)
  ```

  - DBSCAN algoritması çalıştırılıyor.

- `eps = 1`, bir gözlemin kümeye dahil olması için gereken maksimum mesafeyi belirler.
- `minPts = 5`, bir küme oluşturmak için en az kaç noktanın birbirine yakın olması gerektiğini tanımlar.

> **Parametreler Neye Göre Seçildi**
>
> DBSCAN algoritmasında **`eps = 1`**, bir noktanın kümeye dahil olabilmesi için gereken maksimum mesafeyi belirler. Küçük `eps` değeri fazla aykırı değer tespitine, büyük `eps` ise kümelerin birleşmesine neden olabilir. **`minPts = 5`**, bir küme oluşturmak için gereken minimum nokta sayısını tanımlar. Çok küçük `minPts` yanlış kümeler oluşturabilir, çok büyük `minPts` ise küçük kümeleri göz ardı edebilir. Bu değerler genellikle veri setine göre optimize edilmelidir.

- DBSCAN, küme numarası `0` olan gözlemleri aykırı olarak kabul eder.

  ```
  data_multiple_dbscan$aykiri_dbscan <- db_model$cluster == 0
  ```

  - DBSCAN çıktısından aykırı olan gözlemler (`cluster == 0`) belirleniyor.

    Bu kod, DBSCAN algoritmasının çıktılarını kullanarak aykırı değerleri belirliyor. DBSCAN kümeleme yönteminde küme numarası `0` olan gözlemler aykırı olarak kabul edilir. `db_model$cluster` değişkeni, her gözlem için hangi kümeye ait olduğunu gösterir.

    Kodda, `db_model$cluster == 0` ifadesi küme numarası `0` olan gözlemleri `TRUE` olarak işaretleyerek aykırı değer olarak belirler ve bu bilgiler `data_multiple_dbscan$aykiri_dbscan` değişkenine kaydedilir. Böylece, aykırı olan ve kümelere dahil edilemeyen gözlemler veri setinde işaretlenmiş olur.

- **Aykırı Gözlemler İçin Yeni Bir Veri Seti Oluşturuluyor**

  ```
  data_multiple_dbscan_outlier <- data_multiple_dbscan[data_multiple_dbscan$aykiri_dbs
  == TRUE, ]
  ```

  - Sadece aykırı olan gözlemlerden oluşan bir veri seti oluşturuluyor.

- **Aykırı Gözlemleri Ana Veri Seti ile Eşleştirerek Devlet Yardımı Bilgisi Ekleniyor**

  ```
  data_multiple_dbscan_result <- data_multiple[data_multiple$firma_id
  %in% rownames(data_multiple_dbscan_outlier), ]
  ```

  - Aykırı firmaların `firma_id`'leri kullanılarak, ana veri setindeki devlet yardımı bilgisiyle eşleştirme yapılıyor.

- **Sonuçlar Tibble Formatında Görüntüleniyor**

  ```
  as_tibble(data_multiple_dbscan_result)
  ```

– Aykırı değerlerin tibble formatında okunaklı bir şekilde görüntülenmesi sağlanıyor.

Sonuçların Değerlendirilmesi: DBSCAN Yöntemi ve Aykırı Değerler

Şu anki veri setimizde DBSCAN yöntemiyle 28 tane aykırı değere sahip firma tespit edilmiştir ve bu firmalardan 5 tanesi devlet yardımı almış firmalar arasındadır.

DBSCAN yöntemi, yoğunluk tabanlı bir kümeleme algoritması olduğu için, veri setinde düşük yoğunlukta kalan firmaları aykırı olarak belirlemiştir. Bu durum özellikle küçük ölçekli firmalar, sınırlı sayıda ülkeye ihracat yapanlar ve ciro açısından diğerlerinden belirgin şekilde farklı olan firmalar için etkili olabilir. Ancak, DBSCAN veri yoğunluğu düşük bölgelerde kalan firmaları doğrudan aykırı kabul ettiği için, iş modelinden kaynaklı olarak farklı özellikler gösteren bazı firmaları da yanlışlıkla aykırı olarak işaretleyebilir.

Bu durum analiz sonuçlarını etkileyebilir çünkü:

- Eğer tespit edilen aykırı firmalar aşırı yüksek ihracat yapan firmalar ise, devlet yardımı alan firmaların ihracatlarının genelde arttığına dair yanıltıcı bir çıkarım yapılabilir.
- Eğer aykırı firmalar düşük ihracat yapan veya ihracatı olmayan firmalar ise, devlet yardımı alan firmaların başarısız olduğu gibi hatalı bir sonuca varabiliriz.
- Bazı firmalar model tarafından sadece yoğunluk farkından dolayı aykırı belirlenmiş olabilir, bu da analiz sonuçlarının yanlış yorumlanmasına neden olabilir.
- Özellikle büyük cirolara sahip ancak düşük ihracat yapan firmalar model tarafından aykırı olarak işaretlenmiş olabilir, bu da firmanın iş modeline göre farklı değerlendirilmesi gerektiğini gösterir.

DBSCAN yöntemi küme sayısını önceden belirlemeye gerek duymadığı için esnek bir model sunarken, `eps` ve `minPts` parametrelerinin uygun şekilde ayarlanması büyük önem taşır. Eğer `eps` değeri çok küçük seçilmişse çok fazla firma aykırı olarak belirlenebilir, `minPts` değeri çok büyükse bazı aykırı firmalar göz ardı edilebilir.

Aykırı Olmayan Gözlemler (DBSCAN)

```
# Aykırı olmayan gözlemleri filtreleyerek temiz bir veri seti oluşturma
dbscan_data_clean <- data_multiple[!data_multiple$firma_id
                                   %in% rownames(data_multiple_dbscan_outlier), ]

# Yeni veri setini tibble formatında görüntüleyelim
as_tibble(dbscan_data_clean)
```

```
# A tibble: 124 x 7
   firma_id ihracat devlet_yardimi     ciro ihracat_ulkesi calisan_sayisi
      <int>   <dbl> <fct>             <dbl>          <int>          <dbl>
 1        1 443952. 1              2029875.             21            222
 2        2 476982. 0              1647702.             20            202
```

```
 3        3 655871. 1              1641391.              16             172
 4        4 507051. 1              2442325.              12             186
 5        5 512929. 0              1492204.              32             196
 6        7 546092. 0              1954840.              30             206
 7        8 373494. 0              2107269.              46             126
 8        9 431315. 0              1630736.              11             162
 9       10 455434. 0              1712806.              18             135
10       11 622408. 0              1341492.              31             194
# i 114 more rows
# i 1 more variable: faaliyet_yili <int>
```

Sonuç: DBSCAN Kullanmalı mıyım?

Avantajları	Dezavantajları
Önceden belirlenmiş küme sayısına ihtiyaç duymaz, veri yapısına göre esnek çalışır.	Parametre seçimi (`eps` ve `minPts`) sonuçları büyük ölçüde etkiler, yanlış seçilirse ya çok fazla ya da çok az aykırı değer belirlenebilir.
Yoğunluk bazlı olduğu için farklı şekillerdeki kümeleri tespit edebilir ve doğrusal olmayan ilişkileri yakalayabilir.	Düşük yoğunluklu veri setlerinde veya eşit dağılıma sahip verilerde hatalı sonuçlar verebilir, çünkü aykırı değerleri yalnızca yoğunluk farklarına dayalı olarak belirler.
Aykırı değerleri, düşük yoğunluklu bölgelerde kalan noktalar üzerinden belirler, böylece aşırı uç noktaları daha doğal bir şekilde tespit edebilir.	Aykırı değerlerin neden aykırı olduğu konusunda doğrudan bir yorum yapmaz, yalnızca yoğunluk bazlı aykırılık tespiti yapar.
Büyük veri setlerinde iyi çalışır ve kümeler içindeki aykırı değerleri belirlemede başarılıdır.	Yüksek boyutlu veri setlerinde performansı düşebilir, çünkü mesafe hesaplamaları daha karmaşık hale gelir.

Öneri

Eğer veri yoğunluk farklılıkları gösteriyorsa ve kümeler arasında doğal ayrımlar varsa, DBSCAN etkili bir aykırı değer tespit yöntemi olabilir. Ancak, `eps` ve `minPts` değerlerinin dikkatlice belirlenmesi gerekir, aksi takdirde model gereğinden fazla veya yetersiz aykırı değer belirleyebilir. Yoğunluk bazlı aykırılık analizi yapmak istiyorsanız, Mahalanobis mesafesi veya Isolation Forest yerine DBSCAN tercih edilebilir.

Veri Setimiz İçin Uygunluk Değerlendirmesi

DBSCAN, yoğunluk bazlı çalıştığı için firmalar arasında belirgin yoğunluk farkları varsa iyi bir sonuç verebilir. Ancak, firmaların büyüklükleri ve iş modelleri doğal olarak farklı olduğu için, DBSCAN'in bazı firmaları yanlışlıkla aykırı olarak belirleme riski vardır.

Sonuç olarak, DBSCAN yöntemi veri setimiz için doğrudan uygun olmayabilir. Eğer firmalar arasında net bir yoğunluk farkı yoksa, Isolation Forest veya Mahalanobis Mesafesi gibi yöntemler daha güvenilir sonuç verebilir.

3.2.5.4 DBSCAN, Isolation Forest ve Mahalanobis Mesafesi Karşılaştırması

Kriter	DBSCAN	Isolation Forest	Mahalanobis Mesafesi
Dağılım Varsayımı	Yok, yoğunluk tabanlı çalışır.	Yok, veri yapısından bağımsız çalışır.	Normal dağılım varsayar, çarpık veri setlerinde sorun olabilir.
Doğrusal Olmayan İlişkileri Yakalama	Yüksek, farklı küme yapılarını tanıyabilir.	Yüksek, karmaşık ilişkileri yakalayabilir.	Düşük, doğrusal ilişkilere dayalıdır.
Büyük Veri Setlerinde Performans	Orta, yüksek boyutlu veri setlerinde yavaşlayabilir.	Yüksek, büyük veri setleri için ölçeklenebilir.	Orta, yüksek boyutlarda kovaryans matrisinin hesaplanması zorlaşabilir.
Aykırı Değer Tanımlama Yöntemi	Düşük yoğunlukta kalan noktaları aykırı kabul eder.	İzolasyon skoru ile veri noktalarını sıralar.	Verinin merkezine olan uzaklığı ölçerek belirler.
Açıklanabilirlik	Düşük, neden aykırı olduğunu doğrudan açıklayamaz.	Orta, skor üretir ama sebebi doğrudan açıklayamaz.	Yüksek, matematiksel olarak yorumlanabilir.
Parametre Hassasiyeti	Yüksek, `eps` ve `minPts` yanlış seçilirse yanıltıcı olabilir.	Orta, `ntrees` ve `threshold` ayarlanmalıdır.	Orta, eğer kovaryans matrisinde tekillik varsa yanlış sonuç verebilir.

Devlet yardımlarının ihracata etkisini analiz etmek için **en doğru yöntemin seçilmesi gerekir**, çünkü yanlış belirlenen aykırı değerler analiz sonuçlarını saptırabilir.

- **DBSCAN yöntemi**, firmalar arasında yoğunluk farkına dayalı olarak aykırı değerleri belirlediği için, devlet yardımı alan ve almayan firmaların doğal yoğunluk farklarını yanlış bir şekilde aykırı değer olarak işaretleyebilir. Bu yüzden DBSCAN bizim analizimiz için uygun değildir.
- **Mahalanobis Mesafesi yöntemi**, çok değişkenli normal dağılım varsayımı ile çalışır, ancak bizim veri setimizde değişkenlerin normal dağılmaması bu yöntemin güvenilirliğini düşürebilir. Ayrıca, iş dünyasında büyüklük farkları doğrusal olmayabileceği için Mahalanobis Mesafesi bazı büyük ölçekli firmaları hatalı aykırı olarak belirleyebilir.

- **Isolation Forest**, veri dağılımına bağımlı olmadan, karmaşık ilişkileri tespit edebilir ve büyük veri setlerinde ölçeklenebilir bir yöntemdir. Firmaların farklı büyüklüklerde olmasını dikkate alarak aykırı değerleri belirler. Bu nedenle, devlet yardımlarının ihracata etkisini değerlendirirken en uygun yöntem olarak öne çıkmaktadır.

3.2.6 Aykırı Değerlerle Ne Yapılmalı?

Aykırı değerler tespit edildikten sonra şu kararlar alınmalıdır:

3.2.6.1 Aykırı Değerlerin Dönüştürülmesi

Ne zaman kullanılır?

- Eğer **aykırı değerleri tamamen silmek istemiyorsak** ama etkilerini azaltmak istiyorsak.
- Eğer **veri sağa çarpık ise (pozitif çarpıklık varsa)**.

Uygulanacak Yöntemler:

Durum	Önerilen Yöntem	R Uygulaması
Veri sağa çarpık mı?	Log dönüşümü	`log(veri$deger)`
Veri orta derecede çarpık mı?	Karekök dönüşümü	`sqrt(veri$deger)`
Aykırı değerler belli bir aralıkta tutulmalı mı?	Winsorization	`Winsorize(veri$deger, probs=c(0.05, 0.95))`

3.2.7 Aykırı Değerlerin İmpute Edilmesi

Ne zaman kullanılır?

- Eğer **veri kaybetmek istemiyorsak**.
- Eğer **eksik veri yönetimi stratejileri ile uyumlu bir yaklaşım gerekiyorsa**.

Uygulanacak Yöntemler:

Durum	Önerilen Yöntem	R Uygulaması
Verinin merkezi eğilimi korunmalı mı?	Medyan ile doldurma	`median(veri$deger, na.rm = TRUE)`
Verinin değişkenliği korunmalı mı?	Random Forest Imputation	`missForest::missForest(veri)`

Durum	Önerilen Yöntem	R Uygulaması
Veriye yakın değerler baz alınmalı mı?	kNN İmputation	`DMwR::knnImputation(veri)`

3.2.8 Sonuç ve Özet

1. Öncelikle veriyi inceleyin (Boxplot, Histogram, Özet İstatistikler).
2. Eğer tek değişkenli analiz gerekiyorsa, IQR, Z-Skoru veya MAD yöntemlerini kullanın.
3. Eğer çok değişkenli analiz gerekiyorsa, Mahalanobis Mesafesi, İzolasyon Ormanı veya DBSCAN yöntemlerini tercih edin.
4. Aykırı değerleri temizlerken, silme, dönüştürme veya impute etme seçeneklerini duruma göre belirleyin.

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

https://ravenfo.com/2021/02/11/aykiri-deger-analizi/