R ile Uygulamalı Analiz Yöntemleri – I

Book · June 2018

CITATIONS
0

READS
37,071

1 author:

Selahattin Murat Sirin
Ivey Business School
21 PUBLICATIONS **229** CITATIONS

SEE PROFILE

R ile UYGULAMALI ANALİZ YÖNTEMLERİ I

İstatistiğe Giriş ve Açıklayıcı Veri Analizi

SELAHATTİN MURAT ŞİRİN

2018

R ile Uygulamalı Analiz Yöntemleri-I

İstatistiğe Giriş ve Açıklayıcı Veri Analizi

Selahattin Murat ŞİRİN

Eser Adı: R ile Uygulamalı Analiz Yöntemleri - I
Alt Eser Adı: İstatistiğe giriş ve açıklayıcı veri analizi
Yazar: Selahattin Murat ŞİRİN
Yayının Tarihi: 2018
ISBN: 978-605-245-779-5
Konu: Akademik Yayımlar / Genel Konular (Bilgi, Kitap, Veri İşleme) / Bilgisayar Programlama

Ayşegül, Furkan Aras ve Nermin'e

İçindekiler

Tablo Listesi

Şekil Listesi

Önsöz

Son dönemde bilgisayar yazılımlarındaki en önemli atılımlardan birisi açık kaynaklı programların kullanımının hızlı bir şekilde yaygınlaşması olmuştur. Her ne kadar açık kaynaklı programlar yeni bir gelişme olmasa da, internet kullanımının yaygınlaşması ile birlikte paket programlar için milyonlarca doları bulan satın alma veya güncelleme maliyetlerine katlanmak yerine herkesin ücretsiz ve kolaylıkla erişebileceği ve katkıda bulunabileceği programlar hem kapsam hem de kullanıcı sayısı olarak ücretli programlarla yarışır hale gelmiş, hatta bazı alanlarda ücretli programları geçerek temel programlar haline gelmişlerdir. Mesela Harvard Üniversitesi'nin meşhur CS50 ve CS109 derslerinde Python eğitimi verilmekte, doktora programlarında R ve Python, Matlab ve Stata gibi programlarla birlikte öğretilmektedir. Ayrıca Coursera, EdX gibi ücretsiz eğitim veren platformlarda da finansal analizden programcılığa kadar çeşitli alanlarda açık kaynak yazılımları tercih edilmektedir.

Yurtdışındaki bu gelişmelerin ülkemize yansıması olmakla birlikte yeterli seviyeye ulaşamamıştır. Maalesef kamu kurumlarında ve üniversitelerde hala binlerce dolara ulaşan maliyetlerle istatistik/ekonometri programları satın alınmakta veya daha ucuz olan ancak kısıtlı fonksiyonlar sunan öğrenci/akademisyen versiyonları kullanılmaktadır. Buna karşılık, açık kaynaklı programlar gerek eğitimcilerin azlığı gerekse bazı önyargılardan dolayı istenilen seviyede kullanılmamaktadır.

Bu nedenle, bu çalışma, Türkçe yazına katkı sağlamak ve açık kaynak kullanımının yaygınlaşmasını sağlamak amacıyla lisans öğrencilerinden doktora öğrencilerine ve kamu veya özel sektörde çalışanlara kadar geniş bir kesime hitap edecek şekilde R programına girişten başlayarak en yaygın kullanılan analiz yöntemlerini sunmaktadır. Söz konusu konuları tek kitapta kapsamak mümkün olmadığından, bu kitap daha uzun süreli bir çalışmanın ilk basamağı olarak hazırlanmış olup,bu kitapta istatistiğegiriş ve açıklayıcı veri analizleri anlatılacaktır.

Kitabın ilk kısmı R programına giriş ve grafik hazırlama konularını anlatmaktadır. İlk bölümde temel fonksiyonlardan daha karmaşık fonksiyonlara kadar geniş yelpazede günlük kullanımda en çok karşılaşılan konular değerlendirilmiş, ikinci bölümde ise temel grafiklerle birlikte yayın kalitesinde grafik hazırlanması için gereken fonksiyonlar anlatılmıştır.

Kitabın ikinci kısmında ise temel istatistik konuları işlenilmiş ve açıklayıcı veri analizine

ilişkin uygulamalardan bahsedilmiştir. Bu bölüm özellikle devam kitaplarda kullanılacak yöntemler için de temel teşkil etmektedir.

Son olarak, bu kitabın hazırlanmasında her zaman yanımda olan eşim Nermin ve bilgisayarın başında beni bir an yanlız bırakmayan çocuklarım Ayşegül ve Furkan olmak üzere aileme, birikimlerinden önemli ölçüde yararlandığım Orta Doğu teknik Üniversitesi İşletme bölümündeki değerli hocalarıma ve Enerji Piyasası Düzenleme Kurumunda bir dönem birlikte çalıştığım ve zor dönemlerde de yanımdan ayrılmayan değerli arkadaşlarıma teşekkürü bir borç bilirim.

Kısım I

R'ye Giriş

Bölüm 1

R ile Programlamanın Temelleri

Sosyal bilimlerdeki çalışmaların önemli bir kısmı ampirik araştırmaya dayanmaktadır. Ampirik araştırma kısaca gözlemlere ve tecrübeye dayanarak bilgilerin elde edilmesi ve bu bilgilerin bilimsel yöntemlerle değerlendirilerek cevaplanmak istenilen sorulara yanıt aranması olarak tanımlanabilir. Ampirik araştırmalar öncelikle sorunun tanımlanması ile başlar. Karşılaşılan sorunların anlaşılması için öncelikle sorunun ne olduğu ve sorunu oluşturan unsurların tespit edilmesi için önemli mesai ve kaynak harcanması gereklidir. Sorunların tespit edilmesinin ardından söz konusu soruna ilişkin teorilerin araştırılması ve teoriden faydalanarak modelin oluşturulması gereklidir. Her ne kadar teorik çalışmalar sadece üniversitelerin görevi gibi değerlendirilse de şirketlerin veya kamu kurumlarının karşılaştıkları sorunların çözümünde teorilerin kullanılarak sorunun çözümüne ilişkin temellerin oluşturulması sonraki aşamalarda çok önemli kolaylıklar sağlamaktadır. Burada karşılaşılan en önemli sorunlardan birisi teoriler kapsamında kullanılacak verilerin tespit edilmesi, toplanması ve kullanılabilir hale getirilmesidir. Sonraki aşamada ise veri kullanılarak modeller ve tezler değerlendirilir ve elde edilen sonuçların tutarlılığı test edilir. Son aşamada ise elde edilen sonuçlar sunuma hazır hale getirilir ve paydaşlara veya kamuoyuna sunulur. Bu aşamalarda en önemli kısıt gerek maddi gerekse maddi olmayan kaynaklardır. Çalışmaların asgari kaynakla bitirilmesi gerek kamu kurumları gerekse eğitim kurumları için başka alanlarda kullanılabilecek kaynak anlamına gelmektedir. Bu noktada R çok büyük miktarda verinin hazırlanması, modeller oluşturulması ve sonuçların sunulmasında çok büyük kolaylıklar ve diğer programlara göre önemli avantajlar sağlamaktadır. Nitekim son dönemde büyük veri (big data) üzerine çalışan şirketler R'yi mutlaka bilinmesi gereken bir program olarak değerlendirmekte ve ortalama olarak daha yüksek ücret ödemektedir. Kısaca R, gerek maddi gerekse güncellik bakımından kullanıcılarına çok önemli avantajlar sağlamaktadır.

1.1 R Nedir?

R, 1970lerin sonunda Bell laboratuvarlarında geliştirilen S programının devamı niteliğinde

olan ücretsiz, kod esaslı, açık kodlu ve çoğunlukla istatistiki analiz ve veri madenciliği alanında kullanılan bir programdır. 1991 yılında Ross Ihaka ve Robert Gentleman tarafından geliştirilen program, 1993 yılında kullanıma sunulmuş, 1995 yılında ise açık kaynak lisansı almıştır (GNU General Public License). 2000 yılında v.1.0.0 versiyonu yayımlanarak geniş kitlelerin kullanımına sunulmuştur.

R, açık kaynaklı ve ücretsiz olmasının yanı sıra çok derece esnek bir program olması sebebiyle her türlü alanda kullanılmakta, bu sayede farklı alanlardan onbinlerce programcının kullandığı ve sürekli geliştirilen bir program haline gelmiştir. R'ın biyolojiden fiziğe finanstan meteorolojiye çok farklı alanlarda kullanılmasıyla geliştirilen paketler sayesinde teknik olarak çok zengin bir analiz ortamına imkan tanımaktadır. Burada en önemli katkı ise, disiplinlerarası etkileşim sayesinde bir disiplinde kullanılan yöntemler çok kolaylıkla diğer disiplinlere de aktarılabilmektedir. Mesela uydu görüntülerinin incelenmesi için geliştirilen paketler son dönemde iktisadi kalkınma ve ülkelerin ekonomik büyümelerinin incelenmesi için kullanılmaya başlanılmıştır.

1.2 R Kaynakları ve R'nin Yüklenmesi

R temel olarak bir istatistik programı olarak geliştirildiğinden orjinal versiyonu sadece kodların yazılmasına imkan tanımaktadır. Bununla birlikte son dönemde kullanıcı arayüzleri (GUI) ve tümleşik geliştirme ortamları[1] (IDE) geliştirilmiş ve R'nin kullanımı hem kolaylaşmış hem de orjinal versiyonda olmayan fonksiyonlar eklenerek kullanıcılara farklı kullanım imkanları tanınmıştır.

R'nin yüklenmesinde temel olarak (1) R programının yüklenmesi, (2) Kullanıcı arayüzünün (GUI) yüklenmesi anlaşılır. R programı ve paketler, **The Comprehensive R Archive Network (CRAN)** cran.r-project.org adresinden indirilebilir.

Buradan ayrıca R ile ilgili makalelerin yayımlandığı **R Statistical Journal** ve diğer yardım sayfalarına da erişilebilir. Microsoft 2015 yılında Revolution Analytics'i satın alarak Revolution R Open olarak sunulan programı Microsoft R Open (MRAN) mran.microsoft.com olarak ücretsiz kullanıcıların hizmetine sunmaya başlamıştır. Her iki program arasında yüklü olarak gelen paketler ve veri analizi için sunulan bazı hizmetler haricinde fark bulunmamaktadır.

Tümleşik geliştirme ortamları ve Kullanıcı arayüzünde ise çeşitli seçenekler mevcuttur. R'nin en temel özellğı Stata, Eviews ve benzeri programlardan farklı olarak sadece kodlarla çalışabilmesidir. Bu özellik R'yi yeni öğrenenler için öğrenme sürecini güçleştirse de belirli bir eşikten sonra analiz konusunda çok büyük esneklik sağlamaktadır. Bu nedenle, yeni başlayan kullanıcılar zorlukları dikkate alarak kısayolları kullanan IDE/GUI'leri kullanmak yerine kodlarla çalışabileceği arayüzleri tercih etmelidirler.

En yaygın kullanılan ortamların/arayüzlerin başında RStudio.com gelmektedir. Ücretsiz ve

[1] Tümleşik geliştirme ortamları asgari düzeyde kod yazım editörü, derleyici, yorumlayıcı, hata ayıklayıcı ve kodların derlenmesi ve çalıştırılması için çeşitli araçları içeren yazılımlardır.

ücretli iki versiyonu bulunan RStudio, son dönemlerde hazırladığı bazı uygulamalar (Markdown, Shiny vb.) ile kullanıcıların başka programlarda bulamayacağı kolaylıklar sunmaktadır. Ücretsiz versiyonu akademik çalışmalar ve özel kullanım için yeterli olup, paralı versiyonunu çok büyük şirketler ve veri analiz şirketleri kullanmaktadır.

1.2.1 R'nin Yüklenmesi

R'nin yüklenmesi için cran.r-project.org sayfasına girildiğinde "Download and Install R" yazılı bir ekran çıkacaktır. Burada işletim sistemine göre yüklenecek tür seçildikten sonra gelen ekranda, ilk defa yüklenecekse **base** yazısına veya **install R for the first time** yazısına tıklanarak gelen ekrandan R'ın en son versiyonu yüklenebilir.

RStudio'nun yüklenmesi için de RStudio.com adresine girilerek **Products** kısmından **RStudio** seçilir ve ücretsiz olan **RStudio Desktop FREE** seçilerek yükleme yapılır.

RStudio'da dört temel bölüm bulunmaktadır. Bunların konumları kullanıcılar tarafından ayarlanabilmekle birlikte, kullanıcıya göre sol üst tarafta kod yazım editörünün bulunduğu `Source`, sağ üst tarafta kodların değerlendirilmesini ve sonuçları gösteren `Console`, sol alt tarafta veri setlerinin ve değişkenlerin bulunduğu `Environment` ve sağ alt tarafta grafikler, paketler ve yardım dosyalarına erişim imkanının olduğu `Help` bölümleri bulunmaktadır.

1.2.2 R Paketlerinin Yüklenmesi

R'yi diğer çoğu programdan ayıran ikinci özellik paketlere dayalı çalışmasıdır. Paketler , kısaca kullanılacak fonksiyonların yüklü olduğu programlar olarak tanımlanabilir. R haricinde Python gibi programlar da paketlere dayalı geliştirilmekte, bu sayede programın yükleme büyüklüğü düşük tutulmakta ve kullanıcıların istedikleri paketi yükleyerek çalışmalarına imkan tanımaktadır. Ayrıca, şirketler tarafından geliştirilen programlara kıyasla, paketler en son teorik gelişmelerin hızla uygulanmasına imkan tanımakta ve bu sayede geliştirme maliyetlerini düşük tutmaktadır. 2017 sonu itibariyle istatistikten finansa yaklaşık 12 bin paket CRAN'da kullanıma hazır bulunmaktadır. Sosyal bilimlerde yaygın olarak kullanılan paketler ise şu şekilde sıralanabilir:

- **R Programlama** : installr, magrittr, reinstallr,devtools,foreach
- **Grafik**: ggplot2, ggthemes, GGalys, graphics, lattice, prettyGraphs, plotly
- **Veri düzenleme**: data.table, tidyr, dplyr, reshape2, stringr, stargazer, xtable,lubridate, tidyverse
- **Regresyon Analizi**: stats, car, AER, plm, forecast, lme, nls, quantreg, sandwich
- **Açık veri kaynaklarına erişme**: WDI, eurostat, OECD, imfr
- **Finansal Analiz**: financial, quantmod, tis, zoo
- **Veri yükleme**: openxlsx, readr, googlesheets, readxl, RMySQL, XML, httr, rvest
- **Harita ve GIS**: choroplethr

Paketler üç sekilde sisteme yüklenebilir. Birincisi CRAN sayfasından söz konusu paketin ".zip" dosyası bilgisayara indirilip, R Kütüphanesine (~/R/library/) yüklenebilir. İkinci olarak

RStudio veya benzeri GUI de **Packages** kısmında yükle komutu kullanılarak veya üçüncü olarak konsola paketlerin yüklenmesi için `install.packages()` fonksiyonu ile yükleme yapılabilir. Mesela, en yaygın kullanılan grafik paketi olan **ggplot2** paketinin yüklenmesi için yazılacak komut şu şekildedir:

```
install.packages(ggplot2)
```

-Mevcut paketlerin tamamının veya birkaçının güncellenmesi için `update.packages()` fonksiyonu kullanılır.

```
update.packages()
update.packages(ggplot2)
```

-Yüklenilen paketin kullanılması için **Packages** kısmında paket seçilebileceği gibi, `library()` veya `require()` fonksiyonları ile paketler kullanılabilir.

```
library(ggplot2)
require(ggplot2)
```

Bazı durumlarda aynı isimde fakat farklı paketlerde bulunan fonksiyonlar çakışabilir. Paketler yüklenirken buna ilişkin uyarı mesajları da verilmekle birlikte çakışma halinde fonksiyonun paket adından sonra gelecek iki adet iki nokta üst üste ile kullanılması gereklidir.

```
dplyr::mutate()
```

Son olarak, eğer ortak proje yürütülüyorsa veya yazılan kodlar paylaşılacaksa karşı tarafın paket yüklemesini kolaylaştırmak için aşağıdaki şekilde kısa bir ekleme yapılabilir. Bu sayede eğer o paket karşı tarafın bilgisayarında yüklü değilse bu otomatik olarak yüklenmesini sağlayacaktır.

```
if (!require("ggplot2")) install.packages("ggplot2")
require(ggplot2)
```

1.3 Oturum Bilgileri

R'nin açık kaynaklı olması ve ücretli bir ürün olmaması nedeniyle karşılaşılan sorunlar için genellikle forumlardan veya ilgili internet sitelerinden faydalanılır. Özellikle R'nin yeniden başlatılmasına neden olabilecek hatalar ile karşılaşıldığında oturum bilgilerinin olması sorunun daha kolay çözülmesini sağlayacaktır. R'de oturum bilgilerine `sessionInfo()` fonksiyonu ile erişilebilir.

```
sessionInfo() # oturum bilgilerini verir
```

Mevcut oturumda kullanılan veriler ve nesneler hakkında bilgi almak için `ls()` fonksiyonu kullanılır. Komutun içine paket adı yazıldığında paket tarafından kullanılan nesneler gösterilecektir. Eğer söz konusu veriler veya nesneler kaldırılmak istenirse `rm()` fonksiyonu veya

`rm(list = ls())` komutu kullanılır.

```
ls() # Kullanılan veri ve nesneleri listeler
ls("package:ggplot2") # Pakette kullanılan nesneleri listeler.
ls.str() # Kullanılan nesnelerin özelliklerini gösterir.
rm() # Parantez içerisinde belirtilen nesneyi kaldırır.
rm(list = ls()) #Tüm nesneleri kaldırır.
```

Yukarıdaki örneklerde görüldüğü üzere '#' işaretinden sonra yazılanlar R tarafından değerlendirilmemektedir. '#' komutu programa yorum yazılmasına imkan tanıdığından, uzun kodlu çalışmalarda programın takip edilmesini ve anlaşılmasını kolaylaştırmaktadır.

RStudio'da kullanılması tavsiye edilen başka bir kolaylık `RStudio> File > New File > RScript` ile yeni bir script (taslak) dosya açılarak komutların buradan çalıştırılmasıdır. RScript kodların yazıldığı ".Rmd" uzantılı bir dosya olup, aynı komutun konsola tekrar tekrar yazılmasına gerek bırakmadan çalışma yapılmasına imkan tanımaktadır.

R'de yardım dosyasına ulaşmak için `?aranacaknesne` veya `??aranacaknesne` komutları kullanılır. "?" fonksiyonlar için yardım dosyalarını açarken "??" tüm belgelerde söz konusu nesneyi arar. Ayrıca **RStudio Help** bölümüne aranacak kelime yazılarak yardım kullanılabilir.

```
?t.test     # T testi ile ilgili yardım dosyasını açar.
??t.test    # T testi ile ilgili tüm dökümanları gösterir.
```

Son olarak, R çok amaçlı bir program olduğundan birden fazla projede kullanılması halinde çalışma klasörünün farklı tanımlanması dosyaların karışmasını engelleyecektir. Çalışma kitabının tanımlanması için `RStudio -> Tool -> Global Option -> General R` kısmından klasör tanımlanabileceği gibi `setwd("C:/Program Files/R/...")` komutu ile çalışma klasörü tanımlanabilir. Başka bir yöntem de **File -> New project** ile yapılacak her bir çalışma için yeni bir proje oluşturmaktır. Bu sayede söz konusu çalışmaya ilişkin tüm veriler ve grafikler belirlenen klasörde tutulacak ve farklı çalışmalar farklı klasörlerde tutuluğundan çalışmalara erişmek kolaylaşacaktır. Burada dikkat edilmesi gereken hususlardan birisi Windows kullanımının aksine R'da klasör ayraçları "/" şeklindedir.

1.4 R Nesneleri

Yaygın olarak kullanılan çoğu program gibi R de nesne yönelimli (object-oriented) bir programlama dilidir. Bu tür programların temel mantığı, programın etkileşim içerisinden olan birimler veya nesneler kümesinden oluşturulmasıdır. Her nesne bir sınıfa ait olup (class-based) kullanılan fonksiyonlar sınıflara göre işlem yapmaktadır.

R'nin nesne yönelimli üç sistemi bulunmaktadır: S3, S4 ve RC (reference class). Her bir sistemde kendine ait sınıf ve yöntem (method) bulunmaktadır. Sınıflar nesnelerin özelliklerini tanımlarken, yöntem ise belirli bir nesne ile ilgili fonksiyon olarak tanımlanabilir. S3 en temel ve basit sistem olup çok sayıda paketin hazırlanmasında kullanılmaktadır. Bununla

Tablo 1.1: Veri Sınıfları

Sınıf	Açıklama	Sınıf	Açıklama
Character	"a", "swc"	Complex	1+2i
Double	1.0, 2.5, 4.5	Integer	1L, 2L, 4L
Numeric	1.0, 2.5, 4.5	Logical	TRUE, FALSE

birlikte yapılacak analizlerin karmaşıklaşması ile birlikte S4 sistemini kullanan paketlerde oluşturulmaktadır.

R'de her bir öğe bir nesne olup bir sınıfa aittir. Programlamada bunun sağladığı en büyük esneklik her bir öğenin veya fonksiyonun bir nesne olarak kaydedilebilmesidir. R'de temel 10 veri sınıfı bulunmaktadır: `character` , `complex` , `double` , `expression` , `integer` , `list` , `logical` , `numeric` , `single` , `raw` . Bu sınıfların bazıları tablo 1.1 ile gösterilmekte olup, sınıflar verilerin özellikleri ve kullanılabileceği fonksiyonları belirlemektedir. Bu sınıflarla birlikte en yaygın `summary()` , `print()` , `plot()` ve `str()` fonksiyonları kullanılmaktadır.

R'nin gelişmesiyle temel sınıflara yeni sınıflar eklenmiştir. **Methods** paketi ile bu sınıflar ve özellikleri incelenebilir. Bir nesnenin sınıfını görmek için `class()` veya `is.class()` fonksiyonları kullanılırken, nesnenin sınıfını değiştirmek için `as.class()` fonksiyonu kullanılır.

```
a <- 5
b <- "a"
c <- TRUE
A <- c(1:5)
B <- c("a", "b", "c")

class(a)
```

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

```
class(A)
```

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

```
class(b)
```

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

```
class(B)
```

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

```
class(c)
```

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

```
is.numeric(c)
```

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

Tablo 1.2: R Tarih Özellikleri

Format	Açıklaması
%d	Gün
%m	Ay
%b	Yazı ile kısa ay
%B	Yazı ile ay
%y	İki haneli yıl
%Y	Dört haneli yıl
%h-%m-%s	Saat-Dakika-Saniye

```
is.logical(c)
```

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

```
c <- as.numeric(c)
```

R'de kullanılan önemli bir başka sınıf ise Tarih (`Date`) sınıfıdır. R tarih formatının hazırlanmasında esnek olup tarih değerleri birçok formatta hazırlanabilir. `as.Date()` fonksiyonu ile tarih sınıfı tanımlanabilir ve *format* öğesi ile tarihin türü ayarlanabilir. Tarih özellikleri seçilirken Tablo 1.2 ile gösterildiği şekilde "%d" (Gün), "%m" (Ay), "%b" (yazı ile kısa ay), "%B" (Yazı ile ay), "%y" (İki haneli yıl), "%Y" (dört haneli yıl) komutları tarih formatı belirlenebilir. Saat için "%h" (Saat), "%m" (Dakika), "%s" Saniye) kullanılır.

Date sınıfı haricinde `POSIXlt` ve `POSIXct` sınıfları da bulunmaktadır. Eğer bölgeler arası zaman farkları dikkate alınacaksa bu iki sınıf kullanılabilir.

Temel fonksiyonlara ek olarak **lubridate**, **anytime**, **flipTime** paketleri ile tarih ve zaman sınıflarına ilişkin çeşitli fonksiyonlar sunulmaktadır. Özellikle veri analizi için kullanılacaksa bu fonksiyonlar temel fonksiyonlardan daha kullanışlı olup, ilgili paketlerin yardım sayfasına bakılabilir.

```
date1 <- as.Date("10/23/2016", format = "%m/%d/%Y")
date2 <- as.Date("23 October 10", format = "%d %B %y")
date3 <- as.Date("11/10/2016", format = "%m/%d/%Y")
date1 - date2
weekdays(date2)
as.numeric(date3)
```

Son olarak her ne kadar bir nesnenin sınıfı olsa da bu nesnenin saklanma modunu görmek için `mode()` fonksiyonu kullanılır. Bu konuya ilerleyen bölümlerde detaylı olarak değinilecektir.

1.5 R'de Fonksiyonlar

Diğer tüm programlarda olduğu gibi R de fonksiyonlar kullancının atadığı değerler ile çıktı oluşturan programlardır. Bir fonksiyon başlıca şu unsurlardan oluşur:

1) Fonksiyonun adı ve özet bilgileri
2) Fonksiyonun nasıl kullanılacağına ilişkin bilgiler (Usage)
3) Fonksiyonun hangi öğeleri (veri, değişken, vb.) kullandığı (Arguments)
4) Fonksiyonun kullanımına ilişkin detay bilgileri (Details)
5) Fonksiyonun çıktıları (Value)
6) (Varsa) Referanslar

Bu unsurlar tüm fonksiyonlada aynı olup, kullanıcılar tarafından yazılan ama paketlerde sunulmayan fonksiyonlarda da bu hususlara dikkat edilmesi gereklidir. Mesela `sum()` fonksiyonun özellikleri şu şekildedir:

1) Fonksiyon adı ve Özet Bilgileri

```
Sum (from base v3.4.3, by R-core R-core\@R-project.org)
sum returns the sum of all the values present in its arguments.
```

2) Kullanımı (Usage)

```
sum(..., na.rm = FALSE)
```

3) Öğeler (Arguments)

```
... numeric or complex or logical vectors.

na.rm logical. Should missing values (including NaN) be removed?
```

4 Detay Bilgiler

```
This is a generic function: methods can be defined
for it directly or via the Summary group generic....
```

4) Çıktı

```
The sum. If all of ... are of type integer or logical...
```

1.6 Temel Matematik İşlemleri

R yalnızca istatistiki analiz değil matematiksel işlemler içinde yaygın olarak kullanılan bir programdır. Basit matematiksel işlemlerden diferansiyel denklemler ve doğrusal ve doğrusal olmayan programlamaya kadar çeşitli paketler yardımıyla matematiksel hesaplamalar yapılabilmektedir.

En temel düzeyde sunulan matematiksel işlemler için R (epey) gelişmiş bir hesap makinesi olarak düşünülebilir. Cebir işlemleri R tarafından temel fonksiyon olarak sunulmaktadır .

Aşağıda yer alan örneklerde, kullanılan bazı aritmetik işlemler sunulmaktadır. Kullanılabilecek fonksiyonlar bunlarla da sınırlı olmayıp, trigonometrik ve matris işlemleri de olmak üzere çok sayıda işlem yapılabilmektedir.

```
5 + 5 # Toplama
5 * 5 # Çarpma
5 / 5 # Bölme
0 / 0 # Bölme, Not a number (NaN) sonucunu verir.
pi # Pi sayısı
sqrt(5) # Karekök alma
5^2 # 2. kuvvetini hesaplar
5**2 # 2. kuvvetini hesaplar
5 %/% 2 # Bölme işlemi sonucunu tamsayı olarak verir.
5%%2 # mod (5 mod 2)
abs(-5) # Mutlak değeri hesaplar
log(5) # ln(5) değerini hesaplar
exp(5) # e^5 değerini hesaplar
log10(5) # 10 tabanlı logaritmik değeri hesaplar
log2(5) # 2 tabanlı logaritmik değeri hesaplar
round(5/2, digits = 2) # Virgülden sonra iki hane gösterir
signif(12345.6789, digits = 5) # Yuvarlama yaparak ilk beş rakamı gösterir
ceiling(5/2) # En büyük tamsayıyı verir
floor(5/2) # En küçük tamsayıyı verir
trunc(5.48) # Kesir öncesi yer alan tam sayıyı verir
```

Bu fonksiyonlar haricinde **dplyr** paketi ile sunulan `lead()` ve `lag()` fonksiyonları ile sırasıyla bir dizinin önceki veya gecikmeli değerleri alınabilir.

```
require(dplyr)
x <- seq(1:10)
lag(x,1)
```

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

```
lag(x,2)
```

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

```
lead(x,1)
```

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

```
lead(x,2)
```

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

Dizinin tamamının toplamı veya çarpımının gerekli olduğu durumlarda temel fonksiyonlardan `cumsum()` (kümülatif toplam), `cumprod()` (kümülatif çarpım), `cummax()` (maksimum), `cummin()` (minimum) fonksiyonları kullanılır. Ayrıca **dplyr** paketinde yer alan `cummean()`

fonksiyonu bir dizinin sırasıyla toplam değerlerinin ortalamasını verir.

```
require(dplyr)
x <- seq(1:10)
cumsum(x)
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

```
cumprod(x)
```

```
##  [1]       1       2       6      24     120     720    5040   40320
##  [9]  362880 3628800
```

```
cummax(x)
```

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

```
cummin(x)
```

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

```
cummean(x)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

Dizide veya vektörde sıralama değerleri için **dplyr** paketinde yer alan `first()` , `last()` , ve `nth()` fonksiyonları kullanılabilir. Bu fonksiyonlar sırasıyla vektördeki birinci, sonuncu ve n inci değeri verecektir. Ayrıca temel fonksiyonlardan `range()` fonksiyonu da vektörün aralığını (en küçük ve en büyük değerleri) verecektir.

```
require(dplyr)
x <- 5:20
first(x)
```

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

```
last(x)
```

```
## [1] 20
```

```
nth(x,9) #Vektördeki 9. değeri verecektir
```

```
## [1] 13
```

```
range(x)
```

```
## [1]  5 20
```

1.7 Temel R İşleçleri

Tablo 1.3: Temel R İşleçleri

İşleç	Özellikleri	İşleç	Özellikleri
<-	Değer atama	=	Değer atama
:	Aralık değeri atama	c()	Çoklu değer atama
<	Küçüktür	>	Büyüktür
<=	Küçük eşittir	>=	Büyük eşittir
==	Kesinlikle aynıdır	&	Ve
|	Veya	!=	Eşit değildir

Tablo 1.3 R'de en yaygın kullanılan işleçleri göstermektedir. İlk olarak `<-` işleci değer atamak için kullanılmaktadır. 2005 yılından itibaren `=` işareti de değer atama işleci olarak kullanılmaya başlanılmıştır. Ancak, çoğu kullanıcı orjinale sadık kalarak `<-` işlecini kullanmaya devam etmektedir. Bu işleci yazmak için "<" ve "-" tuşları kullanılabileceği gibi `Alt + -` yazarak yani Alt ve eksi tuşlarına birlikte basarak da işleç yazılabilir.

Yaygın kullanılan diğer bir işleç `:` işlecidir. Bu, özelikle sayılar oluşturulurken seri oluşturmak amacıyla kullanılır, yani 1'den 10'a kadar rakamları yazmak istediginizde `1:10` yazarak sayılar oluşturulur. Yalnız `:` işleci sıralı olarak ve birer arttırarak sayıları oluşturmaktadır. Eğer küsüratlı bir sayı verildiyse söz konusu aralıkta ilk sayıya birer ekleyerek diğer sayıları oluşturur.

Yaygın kullanılan başka bir işleç de `c()` (Combine) fonksiyonudur. Parantez içinde yer alan ifadeleri vektör veya liste haline dönüştürmek için veya birden fazla değer kullanılması gerektiğinde bu işleç tercih edilmektedir.

```
a <- 5
b = 5
a*b
c <- 1:10
c
c / (a*b)
d <- 0.25:10.50
d
e <- c(1,2,3,7,9,"a","t")
e
ls()
```

Aritmetik işlemler haricinde, mantıksal işleçlerde kullanılmaktadır. Bu işleçlerden en çok kullanılanları "Küçüktür" (<), "Büyüktür" (>), "Küçük eşittir" (<=), "Büyük eşittir" (>=), "Kesinlikle aynıdır" (==), "Ve" (&), "Veya" (|), "Eşit değildir" (!=), "Değişken değildir" (!x) gibi işleçlerdir.

1.8 Matris İşlemleri

Tablo 1.4: Temel Matris Fonksiyonları

Fonksiyon	Kullanımı
matrix()	Matris tanımlama
rbind()	Satır birleştirme
cbind()	Sütun birleştirme
ncol()	Sütun sayısını verir
nrow()	Satır sayısını verir
t()	Matrisin devriğini verir
det()	Matrisin determinantını hesaplar
eigen()	Karakteristik Kökleri hesaplar
solve()	Matrisin tersini hesaplar
diag()	Birim matris oluşturur
A%*%B	İç çarpım- Dot product
A%o%B	Dış çarpım -Outer product
crossprod(A,B)	A'*B değerini verir
kronecker()	Kronecker çarpımını verir

R'de temel fonksiyon olarak matris işlemleri de yapılabilmektedir. Temel özellikleri MATLAB kadar çeşitli olmasa da geliştirilen **Matrix** gibi paketler ile matris işlemleri çeşitlendirilmiştir.

R'de matris işlemlerine başlamak için `matrix(X, nrow ="")` fonksiyonu ile kullanılacak verinin matris olarak tanımlanması gerekir. Matris olarak seçilecek veri herhangi bir veri seti olabileceği gibi kullanıcı tarafından da veriler girilerek matris oluşturulabilir. Eğer kullanıcı tarafından veri girişi yapılıyorsa veriler ilk sütunun değerlerinden başlayacaktır. Eğer çok sayıda veri girişi varsa, her bir satır veya sütün değerleri girilerek satır için `rbind()` veya sütun için `cbind()` komutları birleştirilebilir. Matris işlemlerinde `nrow()` matristeki satır sayısını, `ncol()` ise matristeki sütun sayısını verir. Yeni bir matris oluşturuluyorsa satır veya sütun sayısının belirtilmesi gerekmektedir. Matris tanımlandıktan sonra `[i,j]` fonksiyonu matrisin satır ve sütun numaralarını gösterir. mesela 3x3'lük A matrisi var ise `A[1,3]` birinci satırın üçüncü sütunundaki veriyi verecektir.

Temel matris işlemlerinde devrik matris (transpose) için `t(A)`,determinant için `det(A)` , karakteristik kökler (eigenvalue) için `eigen(A)` fonksiyonları kullanılır. A matrisinin tersi `solve(A)` fonksiyonu ile hesaplanır. Birim martis için `diag(n)` fonksiyonu ile $n \times n$ boyutunda birim matris oluşturulabilir.

Matriste çarpma işlemleri için `A%*%B` veya `crossprod(A,B)` ile matris çarpımı (dot product) fonksiyonları kullanılır. Ayrıca `A%o%B` fonksiyonu ile dış çarpım (outer product) elde edilir. `kronecker()` fonksiyonu ile iki matrisin Kronecker çarpımları elde edilir.

Temel fonksiyonlara ek olarak **psych** paketinde yer alan `tr()` fonksiyonu ile matrisin izi (trace) ve **Matrix** paketinde yer alan `rankMatrix()` fonksiyonu ile matrisin rankı hesaplanabilir (Revelle 2016; Bates ve Maechler 2016).

Aşağıda sunulan örnekte 3x3'lük A matrisi oluşturulmuş ve buna ilişkin işlem örnekleri verilmiştir.

Tablo 1.5: Verileri Niteliklerine Göre Sınıflandırma

Boyut sayısı	Aynı Tip Veri	Karma Veri
1 Boyutlu	Atomic.vector	List
2 Boyutlu	Matrix	Data frame
N boyutlu	Array	

```
# 3x3lük A ve B matrisleri oluşturuldu.
A <- matrix(c(2,5,7,1,6,12,3,2,2), nrow=3)
a <- c(2,5,7)
b <- c(1,6,12)
c <- c(3,2,2)
B <- rbind(a,b,c)

t(A) # Devrik matris verir
solve(A)
eigen(A) # Özdeğerleri verir.
diag(4) # 4x4 lük birim matris oluşturur.
# A matrisinin ikinci satır üçüncü sütundaki veriyi verir.
A[2,3]
# A ve B matrislerinin Kronecker çarpımını hesaplar
kronecker(A,B)
```

Yukarıda belirtilen fonksiyonlar haricinde **matrixcalc** ve **expm** paketlerinde yer alan fonksiyonlar ile de matris işlemleri yapılabilmektedir.

1.9 Veriler

R'de veri birimleri boyutlarına göre veya aynı tipte olup olmadıklarına göre sınıflandırılabilir.Tablo 1.5 gösterildiği üzere veriler aynı sınıf (tip) veri içerip içermediklerine göre veya boyutlarına göre farklı birimlere ayrılmakta ve bu birimlere göre farklı fonksiyonlarla kullanılmaktadır (Wickham 2014).

R'de en temel veri birimi vektördür . Vektörler (1) Atomik vektörler (atomic.vector) ve (2) Listeler (lists) olmak üzere ikiye ayrılır.Her bir vektörün türü (`typeof()`) , uzunluğu (`length()`) ve özellikleri (`attributes()`) olmak üzere üç karakteri bulunmaktadır. Atomik vektörlerin elemanlarının aynı sınıftan olması gerekli iken liste elemanlarının farklı sınıflardan olması mümkündür.

R farklı sınıflardan elemanlar içeren vektörleri ortak bir sınıf oluşturarak işleme koyar. Mesela `c <- c("a","b","c",1,2,3)` olan bir c vektörü oluşturulduğunda R bunu altı elemanlı bir `character` sınıfı vektör olarak değerlendirir. Eğer söz konusu vektör `numeric` sınıfına çevrilirse (`c <- as.numeric(c)`) program uyarı mesajı verir ve c'nin değeri (NA NA NA 1

2 3) olarak gösterilir. `c <- list(1,2,3,"a","b","c")` yazıldığında ise bir liste oluşturulur ve `class(c)` veya `typeof(c)` yazıldığında `list` sınıfında olduğu belirtilir. Vektörlerin türü ise `is.atomic()` veya `is.list()` fonksiyonları ile sorgulanabilir. Burada önemli bir husus, `list` tipindeki verilerinin `list()`fonksiyonu ile oluşturulması gerekmektedir, aksi takdirde bu `vector` olarak değerlendirilir.

Atomik vektörlerden en yaygın kullanılanları `integer`, `numeric (double)`, `character` ve `logical` sınıflarıdır . Bu vektörlerin sınıflarını daha detaylı sorgulamak için `typeof()` fonksiyonunun yanında `is.numeric()` veya `is.character()` şeklinde sorgulamalar yapılabilir. Bu sınıflardan `integer` ve `numeric` sırasıyla tam sayıları ve reel sayıları, `character` karakter ve simgeleri `logical` ise doğru (TRUE) ve yanlış (FALSE) seçenekli mantıksal nesneleri barındıran sınıflardır. Eğer `logical` vektör numeric veya integer sınıfına çevrilirse doğru değeri 1, yanlış değeri ise 0 değerini alacaktır.

```
a <- c(FALSE, FALSE,TRUE,TRUE)
typeof(a)
```

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

```
is.logical(a)
```

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

```
as.numeric(a)
```

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

Bazen kullanılan nesnelere bazı bilgilerin eklenmesi (metadata) gerekebilir. Bu durumda `attributes()` fonksiyonunun kullanılması gerekir. Temel özellikler adlar - `names()` , boyut - `dim()` ve sınıf- `class()` olarak üç tanedir. Vektörlerin isimleri `names()` fonksiyonu ile belirlenir.Mesela, yukarıda oluşturulan "a" vektörünün isimlerini düzenlemek için `names(a) <- c("a","b","c","d")` komutu yazılarak her bir değere sırasıyla harfler isim olarak atanır.

Bunlar haricinde önemli bir diğer sınıf da kategorik verilerin kaydedildiği *factor* sınıfı verilerdir. Bu sınıf verileri oluşturmak için `as.factor()` fonksiyonu kullanılabilir. Bu fonksiyonda x öğesi kullanılacak veriyi, *levels* kategorileri, *labels* kategori adlarını tanımlamak için kullanılır.

```
factor(x = character(), levels, labels = levels,
       exclude = NA, ordered = is.ordered(x), nmax = NA)
```

Burada dikkat edilmesi gereken husus, *levels* ve *label* komutlarında sıraya göre sınıflandırma yapılmaktadır. Aşağıda yer alan örnekte 10'ar adet 1 ve 2 den oluşan değişken tanımlanmış, sonra kadın için 1, erkek için 2 değeri atanmıştır. Eğer `attributes()` fonksiyonu ile örnek veri incelenirse sınıfa (class) ve düzeye (levels) ilişkin bilgilerin olduğu görülebilir. `Factor` sınıfı verileirn en önemli avantajı kategori oluşturulduğunda, eğer gözlem olmasa bile kategori gösterilmeye devam edecek ve sıfır değeri gösterilecektir. Uygulamada karşılaşılan sorunlardan birisi de diğer programlardan veri aktarırken bazı `character` veya `numeric` sınıfı öğelerin `factor` sınıfında kaydedilmesi veya tam tersi durumla karşı kalınmasıdır. Bu nedenle veriler aktarıldıktan sonra `str()` ile kontrol edilmesi ve yapılacak analize göre sınıfının değiştirilmesi

gerekebilir.

```
# Kategorik verilerin factor fonksiyonu ile tanımlanması
# ve özelliklerinin incelenmesi
veri <- rep(c(1,2), length.out = 20)
summary(veri)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     1.5     1.5     2.0     2.0
```

```
veri <- factor(veri,levels = c(1,2),labels = c("kadin","erkek"))
summary(veri)
```

```
## kadin erkek
##    10    10
```

```
attributes(veri)
```

```
## $levels
## [1] "kadin" "erkek"
##
## $class
## [1] "factor"
```

```
# Gözlem olmayan factor
cins <- c(rep("M",10))
summary(factor(cins,levels = c("F","M"),labels = c("Female","Male")))
```

```
## Female   Male
##      0     10
```

Faktörlerin düzenlenmesi için kullanılabilececek diğer bir yol **forcats** paketi ile sunulan fonksiyonlardan faydalanmaktır. Bu pakette yer alan `fct_relevel()`, `fct_inorder()`, `fct_reorder()` ve `fct_recode()` temel fonksiyonları ile faktörlerin düzeylerinin belirlenmesi ve sıralanması yapılabilir (Wickham 2017a). Bu paket faktörler için hazırlandığından temel faktör işlemlerine ek çok sayıda fonksiyon ile faktör düzeylerine ekleme, çıkarma, sayma, eksik verileri düzenleme, bütünleştirme gibi çok sayıda fonksiyonda sunulmaktadır.

Matris sınıfına ilişkin temel işlemler bir önceki başlıkta bahsedildiğinden tekrar değinilmeyecektir. Matris sınıfına benzer bir şekilde atomik vektörlerin çok boyutlu olmasını sağlayan başka bir fonksiyon da `array()` fonksiyonudur. Aslında "matrix" sınıfı bunun özel bir versiyonu olmasına rağmen uygulamada yaygın olarak "matrix" sınıfı kullanılır. Bir önceki bölümde belirtilenlere ek olarak `rownames()` ve `colnames()` fonksiyonları ile matrisin satır ve sütunlarını da isimlendirmek mümkündür.

```
# A matrisinin satır ve sütunlarının isimlendirilmesi
A <- matrix(c(2,5,7,1,6,12,3,2,2), nrow = 3)
rownames(A) <- c("A","B","C")
colnames(A) <- c("X","Y","Z")
```

```
A
```

```
##   X  Y Z
## A 2  1 3
## B 5  6 2
## C 7 12 2
```

`Data.frame` sınıfı nesne verilerin saklanması için kullanılan ve verilerin analizini çok kolaylaştıran genel bir nesnedir. Diğer nesnelerin aksine `data.frame` hem `matrix` hem de `list` özelliklerini taşır, bu sayede farklı sınıflardan nesnelerin bir arada bulunmasını ve işlem yapılmasını sağlar. Diğer vektörlerde kullanılan özellikler `data.frame` sınıfında da kullanılabilir. Ancak matris sınıfından farklı olarak, `length()` fonksiyonu `ncol()` fonksiyonu gibi sütun sayısını, `nrow()` ise satır sayısını vermektedir. `Data.frame` nesnesi haricinde **tibble** paketi ile sunulan *tibble* nesneleri de son dönemde kullanılmaya başlanılmıştır. Sunduğu fonksiyonlar `data.frame` nesnesinden az olmakla birlikte kolaylık açısından bazı durumlarda tercih edilmektedir.

Veriler `data.frame` olarak kaydedilecekse `data.frame()`fonksiyonunda *row.names* değişken isimleirni tanımlamak için, *stringsAsFactors* = *FALSE* ise `caharcter`vektörünün `factor` olarak kaydedilip kaydedilmeyeceğini belirtmek için kullanılır.

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE, fix.empty.names = TRUE,
           stringsAsFactors = default.stringsAsFactors()
```

Veriler ile uğraşırken belirli format değişiklikleri yapılması gerekebilir. Bu durumda `formatC()` fonksiyonu verinin istenilen şekilde düzenlenmesini sağlar. Bu fonksiyonda x düzenlenecek veriyi, *digits* ondalık işaretinden sonraki basamak sayısını, *format* veri türünü (d: tamsayı, g:gerçel sayı, e:bilimsel gösterim), *mode* verinin türünü (real,integer,character), *decimal.mark* ondalık işaretini (“.” veya “,”), *big.mark* binlik işaretini belirlemek için kullanılır.

```
# formatC foksiyonu
formatC(x, digits = NULL, width = NULL,
        format = NULL, flag = "", mode = NULL,
        big.mark = "", big.interval = 3L,
        small.mark = "", small.interval = 5L,
        decimal.mark = getOption("OutDec"),
        ...)
```

readr paketi ile sunulan `parse_x` fonksiyonları da verilerin düzenlenmesi için kullanılabilir(Wickham, Hester, ve Francois 2017). Bu fonksiyonlar da `as.X()` fonksiyonlarına benzer bir şekilde verilerin düzenlemesini sağlar. Söz konusu fonksiyonlar için paketin yardım sayfasına başvurulabilir.

```
#parse fonksiyonları
require(readr)
parse_logical(x, na = c("", "NA"), locale = default_locale())
parse_integer(x, na = c("", "NA"), locale = default_locale())
```

```
parse_double(x, na = c("", "NA"), locale = default_locale())
parse_character(x, na = c("", "NA"), locale = default_locale())

parse_date(x, format = "", na = c("", "NA"), locale = default_locale())
parse_time(x, format = "", na = c("", "NA"), locale = default_locale())

parse_factor(x, levels, ordered = FALSE, na = c("", "NA"),
  locale = default_locale(), include_na = TRUE)
```

1.10 Veri Girişi

Bir çok istatistik programının aksine R'de veri editörü veya tabloları kullanarak doğrudan veri girişi yapılamamaktadır. Son dönemde geliştirilen bazı açık kaynak uygulamalarda bu özellik sağlansa da R'da veri girişi yukarıda belirtilen sınıflar yoluyla kullanıcı tarafından veya başka programlar vasıtasıyla hazırlanan veriler yüklenerek (.xls, .csv vb.) kullanılmaktadır.

Bu her ne kadar olumsuz bir özellik olarak görülse de son dönemde geliştirilen uygulamalar ile R doğrudan internet sitelerinden veya açık veri kaynaklarından doğrudan verileri alabilmektedir. Ayrıca Excel vb. uygulamalarda yapılan değişiklikler genelde takip edilememekte ve yapılan çalışmaların tekrarı güçleşmektedir. Bu nedenle, akademik çalışmalarda ham verilerin düzenlenmesi ve bu düzeltme işlemlerine ilişkin kodların da yapılan çalışma ile birlikte sunulması istenildiğinden, R kullanıcılarına bu noktada da avantaj sağlamaktadır.

R'de kullanıcı tarafından veri girişi bir nesne oluşturarak ve bu nesneye ait elemanları belirterek yapılır. Belirli bir dizilimi takip eden veri girişi yapılması gerektiğinde `rep()` ve `seq()` fonksiyonları kullanılarak veri girişi kolaylaştırılabilir. `seq()` fonksiyonu belirli aralıklarla sayı serisi elde etmeye imkan tanırken, `rep()` fonksiyonu ile dizilimin tekrarlanması sağlanır. `seq()` fonksiyonunda başlangıç (from), bitiş (to), aralık (by) veya istenilen seri uzunluğu (length.out) değerleri belirlenebilirken, `rep()` fonksiyonunda tekrar edilecek seri (x), tekrar sayısı (times) veya istenilen seri uzunluğu (length.out) ve tekrar sayısı (each) belirtilerek seri oluşturulur.

```
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL)

rep(x, times = 1, length.out = NA, each = 1)
```

Aşağıda yer alan örneklerde çeşitli yöntemlerle veri girişleri gösterilmektedir. Örnekte yer alan `rnorm()` fonksiyonu normal dağılımlı bir seri elde etmeye imkan tanıyan bir fonksiyon olup ilerleyen bölümlerde detaylı olarak anlatılacaktır.

```
veri <- data.frame( x = c(1:5), y = rnorm(5,0,0.5))
veri.2 <- data.frame(a = c(1:10), b = rnorm(10, 0, 0.5), c = rep(c(1:2)))
```

```
a <- matrix(c(1, 2, 4, 2, 6, 3, 5, 8, 1),nrow=3)
b <- rep(c(1:5), times=5)
c <- rep(c(1:5), each=5)
d <- rep(c(1:5), length.out=10)
e <- seq(0, 100, length.out = 10)
f <- seq(0, 50, by=5)
veri.3 <- data.frame(b, c, d, e, f)
```

Eğer kullanıcılar SPSS veya Stata gibi programların sunduğu veri düzenleyicilerden faydalanmak isterse `data.entry()` fonksiyonu ile önceden tanımlanmış bir veri setine veya değişkene ait gözlemler düzenlenebilir ve değiştirilebilir. Ancak özellikle çalışmanın tekrar edilebilirliği açısından bu yöntem genelde tercih edilmemektedir.

Kullanıcı tarafından veri girişi haricinde diğer programlarda hazırlanan veriler de yüklenerek kullanılabilir. Ancak, bu amaçla çok sayıda paket geliştirildiğinden sunulan özellikler ve fonksiyonlar farklı olup, kullanıcıların buna dikkat etmesi gerekmektedir.

RStudio kullanılarak veri girişi için `File -> Import Dataset` kısmından Excel, csv, Stata, SPSS ve SAS dosyaları indirilebilir. Excel dosyaları seçildiğinde dosyaların yüklenmesi için yükleme ekranı çıkacak ve bu ekranda dosyanın bulunduğu klasör veya internet bağlantısının adresi seçilerek, yüklenecek veri setine verilecek isim (name), excel kitabının kaçıncı sayfasında olduğu (sheet), veri için ilk satırın dosyanın kaçıncı satırından itibaren başlayacağı (skip), veri için ilk satırda yer alan bilgilerin değişken adı olarak kullanılıp kullanılmayacağı (First Row as Names) ve eksik verilerin nasıl gösterileceği belirtilir. Yan tarafta yer alan ufak ekranda ise dosyanın yükleme komutu gösterilmektedir. Eğer veri seti uzun süreli çalışmalarda kullanılacaksa bu alanda yer alan kodun kopyalanarak RScript'e eklenmesi verinin otomatik olarak yüklenmesini sağlar.

Eğer kullanıcı arayüzü kullanılmadan dosyalar indirilmek istenirse kullanılabilecek ilk fonksiyon `read.table()` fonksiyonudur. Bu fonksiyon ".txt" dosyalarının yüklenmesini sağlar. Eğer aksi belirtilmemişse bu fonksiyon dosyaları `setwd()` fonksiyonu ile tanımlanmış çalışma klasöründen alacaktır. *header = TRUE* olarak yazılırsa ilk satırda yer alan değerler değişken isimleri olarak belirlenir. *sep ="/"* ise bölümler arasında kullanıan ayracı belirtmeye yarar. Eğer "" olarak tanımlanırsa bölümler arasında boşluk, ";" olarak tanımlanırsa noktalı virgül kullanıldığı belirtilir. *strip.white* öğesi ise ayraç tanımlandığında karakterlerin başında ve sonunda yer alan beyaz boşlukların ortadan kaldırılmasını (TRUE) seçildiğinde sağlar.

```
veri <- read.table("DOSYAADI.txt",
                   header = FALSE,
                   sep="/",
                   na.strings = "EMPTY"
                   strip.white=TRUE)
```

CSV dosyasının yüklenmesi için `read.csv()` ve `read.csv2()` fonksiyonları kullanılabilir. Buradaki fonksiyon özellikleri `read.table()` fonksiyonunda olduğu gibidir. Farklı olarak *quote* karakterleri tanımlamak için kullanılan sembolleri belirtirken, "stringsAsfactors" satırlarda yer alan karakterlerin faktör sınıfı olarak tanımlanıp tanımlanamayacağını belirtir.

```
veri <- read.csv("DosyaADI.UZANTISI",
                 header = TRUE,
                 quote="\"",
                 stringsAsFactors= TRUE,
                 strip.white = TRUE)
```

Verilerin yüklenmesi için kullanılabilecek üçüncü fonksiyon grubu `read.delim()` ve `read.delim2()` fonksiyonlarıdır. Bu fonksiyonlar yukarıda belirtilen fonksiyonların daha da geliştirilmiş halidir. Yukarıda belirtilenlere ek olarak *dec* ondalık ayıracını (virgül veya nokta), *row.names* satır isimlerini, *fill = TRUE* satırların aynı uzunlukta olmaması halinde boşlukların nasıl değerlendirileceğini (TRUE olursa boş alanlar eklenir), *na.strings ="EMPTY"* karakter içeren satırların hangilerinin eksik veri olarak değerlendirileceğini *as.is* karakter vektör içeren hangi sütun(ların) olduğu gibi bırakılacağını ve başka bir sınıfa dönüştürülmeyeceğini, *nrows* toplamda kaç satırın veri olarak yükleneceğini ve *skip* ise baştaki kaç satırın veri olarak değerlendirilmeyeceğini belirlemeye imkan tanır.

```
veri <- read.delim("DosyaADI.UZANTISI",
                   header = TRUE,
                   sep = "/",
                   quote  "\"",
                   dec = ".",
                   row.names = c(),
                   fill = TRUE,
                   strip.white = TRUE,
                   stringsAsFactors = TRUE,
                   na.strings = "EMPTY",
                   as.is = 3,
                   nrows = 5,
                   skip = 2)
```

Bu fonksiyonlar `read.table()` fonksiyonunun bir uzantısı olarak **utils** paketi ile yüklü olarak sunulmaktadır (Bengtsson 2016). Bunlar haricinde **XLConnect**, **gdata** paketlerinde de Excel dosyalarının yüklenmesine imkan tanıyan fonksiyonlar bulunmaktadır (Studer 2016; Warnes vd. 2016). **XLConnect** paketi yalnızca verilerin yüklenmesi değil, aynı zamanda interaktif bir şekilde Excel dosyasında düzenleme yapılması ve verilerin Excel formatında kaydedilmesini sağlar.

Son dönemde geliştirilen **readr** paketi ile sunulan `read_csv()`, `read_csv2()` ve `read_delim()` fonksiyonları da Excel dosyalarının `read.csv()` fonksiyonuna göre daha hızlı yüklenmesi ve bu fonksiyonda karşılaşılan bazı sorunların yaşanmaması nedeniyle tercih edilmeye başlamıştır(Wickham, Hester, ve Francois 2017). Bu fonksiyonların özellikleri de `read.csv()` fonksiyonlarının özellikleri gibidir. **readr** paketi ayrı olarak yüklenebileceği gibi **tidyverse** paketi ile birlikte yüklenebilir.

```
library(tidyverse)
read_csv(file, col_names = TRUE, col_types = NULL,
```

```
  locale = default_locale(), na = c("", "NA"), quoted_na = TRUE,
  quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf,
  guess_max = min(1000, n_max), progress = show_progress())

read_delim(file, delim, quote = "\"", escape_backslash = FALSE,
  escape_double = TRUE, col_names = TRUE, col_types = NULL,
  locale = default_locale(), na = c("", "NA"), quoted_na = TRUE,
  comment = "", trim_ws = FALSE, skip = 0, n_max = Inf,
  guess_max = min(1000, n_max), progress = show_progress())
```

Excel verileri haricinde diğer programlarda hazırlanan veriler **foreign** ve **Hmisc** paketleri ile Kullanılabilir (R Core Team 2016; Harrell Jr 2016).

```
library(foreign)
library(Hmisc)
#SPSS Programı
data.spss <- read.spss("Dosya Adı")
# SAS Programı
data.sas <- sasxport.get("Dosya Adı")
# Stat Programı
data.stata <- read.stata("Dosya Adı")
# Systat programı
data.systat <- read.systat("Dosya Adı")
```

Bunlar haricinde son dönemde yaygınlaşan açık kaynak politikaları ile birlikte bir çok uluslararası kuruluş, şirket veya kamu kurumu tarafından sunulan verilere geliştirilen paketler ile de erişmek mümkündür. Mesela Dünya Bankası verilerine **WDI** paketi, Ekonomik İşbirliği ve Kalkınma Örgütü (OECD) verilerine **OECD** paketi ve Eurostat verilerine **eurostat** paketler ile erişmek mümkündür (Arel-Bundock 2016; Persson 2016; Leo vd. 2016). Örnek olarak Dünya Bankası WDI göstergelerinden 1990-2014 dönemi kişi başı gelir verilerini kullanılmak istendiğinde aşağıdaki kod kullanılabilir.

```
require(WDI)
# Dünya Bankası verilerinden içinde gdp geçen verilerin listesinin alınması
gdp.list <- WDIsearch(string = "gdp")
#Kullanılacak verinin kodunun seçilmesi
ind <- gdp.list[94,1]
#1990-2014 yılları arasında kişi başı gelir verisinin alınması
gdp_per_capita <- WDI(country = "all", indicator = ind ,
                      start = 1990,end = 2014)
```

Bu paketlerin çalışması için internet bağlantısının olması gereklidir. Veriler yüklendikten sonra ".Rdata" formatında veya başka bir formatta kaydedilerek daha sonra kullanılabilir. Ayrıca son dönemde geliştirilen paketler ile "web-scrapping" olarak adlandırılan ve internet sitelerinden veya API'lerden veri indirilerek kullanılabilir.

Tablo 1.6: Temel Veri Fonksiyonları

Fonksiyon	Amacı
head()	İlk sıradaki verileri gösterir
tail()	Son sıradaki verileri gösterir
names(),colnames()	Sütun adlarını gösterir
dim()	Verinin boyutlarını gösterir
srt()	Verinin yapısını gösterir
table(),xtabs()	Veri sıklığını ve çapraz tabloları gösterir
margin.table(), prop.table()	Tablonun sıklık ve marjinal değerlerini gösterir
ftable()	Üç değişkenli sıklık tablosunu gösterir

Hazırlanan veriler ".Rdata" formatında saklanabileceği gibi başka formatlarda da kaydedilebilir. Excel formatında verilerin kaydedilmesi için **WriteXLS** ve **XLConnect** paketleri kullanılır (Schwartz 2015). **WriteXLS** paketinde yer alan `WriteXLS()` fonksiyonu veri setlerinin Excel 2003 (.xls) veya Excel 2007 (.xlsx) formatlarında kaydedilmesini sağlar. Bu fonksiyondaki *ExcelFileName* dosya adı ve uzantısını, *SheetNames* Excel sayfası oluşturulacaksa adlarını *row.names* ve *col.names* satır ve sütun isimlerinin aktarılmasını, *na* eksik verilerin nasıl gösterileceğini ayarlamayı sağlar.

```
WriteXLS(VERI, ExcelFileName = "DOSYAADI.xls/xlsx",
           SheetNames = NULL,
           row.names = FALSE, col.names = TRUE,
           AdjWidth = FALSE, AutoFilter = FALSE,
           na = "")
```

Excel dışındaki formatlar için **foreign** paketi kullanılarak veriler kaydedilebilir.

```
# SPSS programı için
write.foreign(data, "c:/mydata.txt", "c:/mydata.sps",   package="SPSS")
# SAS Programı için
write.foreign(data, "c:/mydata.txt", "c:/mydata.sas",   package="SAS")
#Stata Programı için
write.dta(data, "c:/mydata.dta")
```

1.11 Verilerin İncelenmesi

Yüklenen verilerin incelenmesi için (1) RStudio'nun "Environment" bölmesini ve (2) İzleme komutları kullanılabilir. Eski versiyonlarında verilerin özelliklerini görmek ve verileri sıralamak için komutların kullanılması gerekirken son eklenen özelliklerle verilerin doğrudan sıralanması ve süzülmesi mümkün hale gelmiştir.

Verileri incelemek için genel olarak kullanılan fonksiyonlar Tablo 1.6 ile gösterilmektedir.

Bunlara ek olarak hazırlanan tablolarla ilgili yapilacak testler için **gmodels** paketinde yer alan `CrossTable()` fonksiyonu kullanılabilir (Warnes vd. 2015).

Bu komutların örnekleri **datasets** paketinde yer alan `mtcars` veri seti ile gösterilecektir. Mtcars 1973-74 yıllarında kullanılan araçların yakıt tüketimi (mpg), silindir sayısı (cyl), beygir gücü (hp), arka aks oranı (drat), pound cinsinden ağırlığı (wt), hızlanma süresi (qsec), vites türü (am), ileri vites sayısı (gear) ve karbüratör (carb) sayısı özelliklerini bulunduran 32x11'lik bir veri setidir.

```
veri <- mtcars
# Ilk beş satırı gösterir.
head(veri,n = 5L)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

```
# Son beş satırı gösterir.
tail(veri, n = 5L)
```

```
##                mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

```
#Sütun adlarını gösterir
colnames(veri)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```
#Satır sütun sayısı olarak verinin boyutlarını gösterir
dim(veri)
```

```
## [1] 32 11
```

```
# Verinin yapısını gösterir
str(veri)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
```

```
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
# Satırı vites (gear), sütunu karbüratör sayısı (carb) olan bir tablo
#oluşturur
with(veri,table(gear, carb))
```

```
##     carb
## gear 1 2 3 4 6 8
##    3 3 4 3 5 0 0
##    4 4 4 0 4 0 0
##    5 0 2 0 1 1 1
```

```
# Satırı vites (gear), sütunu karbüratör sayısı (carb) olan bir tablo
#oluşturur
xtabs( ~ gear + carb, data = veri)
```

```
##     carb
## gear 1 2 3 4 6 8
##    3 3 4 3 5 0 0
##    4 4 4 0 4 0 0
##    5 0 2 0 1 1 1
```

```
# Satırın marjinal değerlerini verir
tablo <- with(veri,table(gear, carb))
margin.table(tablo,1)
```

```
## gear
##  3  4  5
## 15 12  5
```

```
# Sütunun marjinal degerlerini verir
margin.table(tablo,2)
```

```
## carb
##  1  2  3  4  6  8
##  7 10  3 10  1  1
```

```
# Tablonun sıklık oranlarını verir
prop.table(tablo)
```

```
##     carb
## gear       1       2       3       4       6       8
##    3 0.09375 0.12500 0.09375 0.15625 0.00000 0.00000
```

```
##    4 0.12500 0.12500 0.00000 0.12500 0.00000 0.00000
##    5 0.00000 0.06250 0.00000 0.03125 0.03125 0.03125
```

```
# Satırın sıklık oranlarını verir
prop.table(tablo,1)
```

```
##     carb
## gear         1         2         3         4         6         8
##    3 0.2000000 0.2666667 0.2000000 0.3333333 0.0000000 0.0000000
##    4 0.3333333 0.3333333 0.0000000 0.3333333 0.0000000 0.0000000
##    5 0.0000000 0.4000000 0.0000000 0.2000000 0.2000000 0.2000000
```

```
# Sütunun sıklık oranlarını verir
prop.table(tablo,2)
```

```
##     carb
## gear         1         2         3         4         6         8
##    3 0.4285714 0.4000000 1.0000000 0.5000000 0.0000000 0.0000000
##    4 0.5714286 0.4000000 0.0000000 0.4000000 0.0000000 0.0000000
##    5 0.0000000 0.2000000 0.0000000 0.1000000 1.0000000 1.0000000
```

```
# İlk sırada yer alan silindir sayısına göre vites sayısı
# ve karbüratör sayısını verir.
ftable(xtabs( ~  cyl + gear + carb, data = veri))
```

```
##          carb 1 2 3 4 6 8
## cyl gear
## 4   3         1 0 0 0 0 0
##     4         4 4 0 0 0 0
##     5         0 2 0 0 0 0
## 6   3         2 0 0 0 0 0
##     4         0 0 0 4 0 0
##     5         0 0 0 0 1 0
## 8   3         0 4 3 5 0 0
##     4         0 0 0 0 0 0
##     5         0 0 0 1 0 1
```

Veri setlerinde bazı verileri seçmek için `c()` işleci yerine kullanılabilecek diğer bir fonksiyon ise **wrapr** paketinde yer alan `qc()` fonksiyonudur. Bu fonksiyon veri setlerinde yer alan verileri tırnak işareti içinde belirtmeye gerek olmadan seçmeye yarar. Aynı pakette yer alan `qe()` ve `qs()` fonksiyonları da benzer şekilde kullanılabilir. Aşağıda yer alan örnekte `mtcars` veri setinde yer alan *mpg* ve *hp* değişkenlerinin ilk altı değeri gösterilmektedir.

```
if (!require("wrapr")) install.packages("wrapr")
require(wrapr)
data(mtcars)
head(mtcars[,qc(mpg, hp)])
```

```
##                    mpg  hp
## Mazda RX4         21.0 110
## Mazda RX4 Wag     21.0 110
## Datsun 710        22.8  93
## Hornet 4 Drive    21.4 110
## Hornet Sportabout 18.7 175
## Valiant           18.1 105
```

1.12 Eksik Veri

Sayısal analizlerde en önemli sorunlardan biri eksik verilerdir. Eksik veriler ölçüm yapılamamsından kaynaklanabileceği gibi hatalı kayıtlr neticesinde de meydana gelebilir. Bazı çalışmalarda eksik veriler çeşitli yöntemler kullanarak tamamlanabilirken çoğu zaman da analizlerden çıkartılmaktadır. Bir veri setinde eksik veri olup olmadığını anlamak için ilk olarak `complete.cases()` fonksiyonu kullanılır. Eksik verileri veri setinden çıkarmak içinse `na.omit()` fonksiyonu kullanılır.

```
complete.cases(...)
na.omit(object)
```

```
data(mtcars)
complete.cases(mtcars)
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [29] TRUE TRUE TRUE TRUE
```

Bu temel fonksiyonlar başlangıç için faydalı olmakla birikte veri setinin incelenerek eksik verilerin yapısını görmek için başta **MICE** paketi (Multivariate Imputation by Chained Equations) olmak üzere **Amelia**, **Hmisc**, **missForest** paketleri kullanılmaktadır (Buuren vd. 2015; Honaker, King, ve Blackwell 2016; Stekhoven 2016).

MICE paketinde yer alan `md.pattern()` fonksiyonu eksik verilerin dağılımını göstermek için kullanılır. Aşağıda sunulan örnekte kayıp verilerin bulunduğu `nhanes` veri seri sunulmaktadır. Söz konusu veri setinin yapısına bakıldığında 5x5'lik bir tablo sunulmaktadır. Tabloda '1' eksik olmayan veriyi, '0' ise eksik veriyi temsil etmektedir. Tabloda en son satır eksik kaç veri olduğunu ve en son sütun ise kaç değişkende eksik veri olduğunu göstermektedir. İlk sütun ise hangi değişkende kaç satırda verilerin olduğunu göstermekte ve sırasıyla eksik veri olan değişkenlere göre satır sayısını göstermektedir. `nhanes` veri setinde 13 satırda tüm veriler eksiksiz iken bir satırda *bmi* değişkeninin eksik olduğu, üç satırda *chl* verisinin eksik olduğu, bir satırda hem *hyp* hem de *bmi* verisinin eksik olduğu ve 7 satırda ise sadece *age* verisinin oldugu görülmektedir. Toplamda ise *hyp* değişkeninde 8 veri, *bmi* değişkeninde 9 veri ve *chl* değişkeninde 10 veri ve toplam 27 verinin eksik olduğu görülmektedir.

```
require(mice)
veri <- nhanes
md.pattern(veri)
```

```
##    age hyp bmi chl
## 13   1   1   1   1  0
##  1   1   1   0   1  1
##  3   1   1   1   0  1
##  1   1   0   0   1  2
##  7   1   0   0   0  3
##      0   8   9  10 27
```

```
# Eksik verilerin bulunduğu satırlar çıkartıldı
veri.1 <- na.omit(veri)
md.pattern(veri.1)
```

```
##      age bmi hyp chl
## [1,]   1   1   1   1 0
## [2,]   0   0   0   0 0
```

1.13 Verilerin Hazırlanması

Verilerin yüklenmesi ve verilere ilişkin ilk değerlendirmeleirn yapılmasından sonraki aşama verilerin analize hazır hale getirilmesi ve kullanılmasıdır. Konuya devam etmeden önce verilerle uğraşan herkesin uyması gereken bazı kuralların hatırlatılması faydalı olacaktır. Her ne kadar basit olarak degerlendirilse de bu kurallara uyulması hem çalışırken yapılacak hataları asgari düzeye indirecek hem de çalışmanın güvenilirliğini arttıracaktır. Bu süreçte yapılan hatalara ilişkin son dönemde öne çıkan en önemli olay, Harvard Üniversitesi profesörlerinden Carmen Reinhart ve Kenneth Rogoff'un 2010 yılında yaptıkları bir çalışmada Excel tablolarında beş satırda verileri yanlışlıkla çıkartmaları ve bu hatanın başka bir üniversiteden akademisyenlerin söz konusu çalışmayı tekrar etmek istemeleri üzerine kamuoyuna yansıması ve tekrar edilen çalışma neticesinde katsayıların yanlış hesaplandığının ileri sürülmesidir[2]. Bu olay, verilerin hazırlanmasında yapılabilecek basit hataların en başarılı akademisyenleri bile zor duruma düşürebilecek nitelikte olduğunun güzel bir örneğidir.

Bu nedenle veri analizinde bir numaralı kural, ham verilerin asla ve asla değiştirilmemesidir. Ham veriler üzerinde yapılacak değişikliklerin takip edilmesi zor olduğundan hem çalışma sahibi hem de çalışmadan faydalanacaklar için ham verinin olduğu gibi kalması elzemdir. İkinci olarak, ham verilerde yapılan düzenlemelere ilişkin kodlar RScript veya başka bir şekilde tutulması ve düzenlenen veri seti karışıklık yaratmayacak şekilde kaydedilmelidir.

Veriler incelenirken veriden kaynaklanan hatalar, aykırı ve uç değerlere ve eksik verilere

[2] http://www.bloomberg.com/news/articles/2013-04-18/faq-reinhart-rogoff-and-the-excel-error-that-changed-history

dikkat edilmeli, yapılan düzenlemeler ve düzeltme yöntemleri açıklama veya dip not olarak belirtilmelidir.

Verilerin hazırlanması için çok sayıda paket mevcut olup **data.table** ve **dplyr**,**tidyr**, **reshape2** paketleri bunların başlıcalarıdır (Dowle vd. 2016; Wickham ve Francois 2016; Wickham ve Henry 2017). Son dönemin önemli paket geliştiricilerinden Hadley Wickham tarafından hazırlanan **tidyverse** paketi ile bu paketler ve veri analizi için kullanılabilecek çok sayıda paket ve fonksiyon yüklenebilir.

1.13.1 Değişken isimlerinin düzenlenmesi

Az değişkenin kullanıldığı çalışmalarda değişkenler kolaylıkla anlaşılabilir ve takip edilebilirken değişken sayısı arttıkça isimler karışıklığa yol açabilir. Bu nedenle değişken isimlerinin belirli bir formatta ve anlaşılabilir şekilde hazırlanması ve verilerdeki karakterlerin belirli bir formatta sunulması önem arz etmektedir.

Değişken isimlerini düzenlemek için temel `names()` fonksiyonu sunulmaktadır. Ancak son dönemde veri madenciliğinin gelişmesi ile hazırlanan paketler sayesinde farklı fonksiyonlar da kullanılabilmektedir.

Sunulan ilk örnekte "araba" isimli bir veri seti oluşturulup, değişken isimleri kontrol edilmiş, ikinci değişkenin ismi silindir olarak değiştirilmiştir. İkinci örnekte ise **dplyr** paketinde yer alan `rename()` fonksiyonu kullanılarak "mpg" olan değişken ismi "Yakıt" olarak değiştirilmiştir.

```
require(dplyr)
#Names fonksiyonu kullanılarak değişken isimlerinin düzenlenmesi
# Araba veri setini oluşturur
araba <- mtcars
# Araba veri setinde yer alan değişkenlerin isimlerini listeler.
names(araba)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```
# 2. değişkenin ismini görüntüler
names(araba)[2]
```

```
## [1] "cyl"
```

```
#2. değişkenin adını silindir olarak değiştirir
names(araba)[2] <- "silindir"
names(araba)
```

```
##  [1] "mpg"      "silindir" "disp"     "hp"       "drat"     "wt"
##  [7] "qsec"     "vs"       "am"       "gear"     "carb"
```

```
# Dplyr paketi kullanılarak değişken ismini değiştirir
```

```
araba.1 <- mtcars %>%
  rename(Yakit = mpg)
names(araba.1)
```

```
##  [1] "Yakit" "cyl"   "disp"  "hp"    "drat"  "wt"    "qsec"  "vs"
##  [9] "am"    "gear"  "carb"
```

İkinci örnekte "araba.1" seti oluşturulurken "mtcars" ifadesinin arkasına " %>% " isaretlerinin eklendiği dikkatinizi çekmiştir. Bu işaret **magrittr** paketi ile eklenen "forward-pipe (fpo)" işlecidir . Özellikle uzun kodlarda, yazımı sadeleştirmek ve anlaşılmasını kolaylaştırmak amacıyla 2014 yılında kullanıma sunulmuş ve çok kısa bir sürede popüler hale gelmiştir (Bache ve Wickham 2016). Buna ilişkin daha detaylı bilgi R ile Programlama kısmında verilecektir.

1.13.2 Yeni değişken oluşturulması

R'nin diğer istatistik programlarına göre en önemli avantajlarından birisi değişken oluşturmada sağladığı esnekliktir. En temel düzeyde değişken oluşturmak için veri seti ve veri setinde yer alan değişkenleri düzenlemek için `$` işleci kullanılır. Benzer bir şekilde **dplyr** paketinde yer alan `mutate()` ve `transmute()` fonksiyonları da yeni değişken oluşturmak için kullanılır. `mutate()` fonksiyonu kullanıldığında mevcut değişkenler korunurken `transmute()` fonksiyonu eski değişkenleri veri setinden kaldırır.

Nominal verilerden kategorik veriler elde etmek için `cut()` fonksiyonu da kullanılan başka bir fonksiyondur. Bu fonksiyonda *x* öğesi kullanılacak veri setini, *breaks* kategorileri ve *labels* kategori adlarını tanımlamak için kullanılır. Ancak sınırlar belirlenirken veri setindeki en düşük ve en yüksek veriye dikkat edilmeli ve bu değerler *breaks* içinde tanımlanmalıdır. Aksi takdirde veri tanımlanamamaktadır.

```
cut(x, breaks, labels = NULL,
    include.lowest = FALSE, right = TRUE, dig.lab = 3,
    ordered_result = FALSE, ...)
```

Bazen değişkenlerin standart hale getirilmesi yani ortalama ve standart sapmaya göre uzaklıklarının hesaplanması gerekebilir. Bu durumda `scale()` fonksiyonu kullanılarak standartlaştırma yapılabilir.

Aşağıda yer alan örnekte `mtcars` veri setinde silindir başına yakıt tüketimini (mpgc= mpg/cyl) gösteren bir değişken oluşturularak verimlilik sınıflarına göre kategorilere ayrılmıştır.

```
require(dplyr)
data(mtcars)
# mpg / cyl degerini hesaplayarak mpgc isimli bir degisken olusturur
mtcars$mpgc <-  mtcars$mpg / mtcars$cyl
summary(mtcars$mpgc)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   1.300   1.928   3.108   3.837   5.700   8.475
```

```
# Dplyr paketi kullanılarak yeni degisken olusturulmasi
mtcars.1 <- mtcars %>%
  mutate( mpgc = mpg /cyl)
names(mtcars.1)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb" "mpgc"
```

```
#Dplyr kullanılarak mpg'ye göre A-F arası verimlilik sınıfları ve
#normalleştirme yapılarak mpg_z isimli değişken oluşturulmuştur.
mtcars.2 <- mutate(mtcars,
                   verimlilik = cut(mpg,
                                    breaks=c(0,10,15,20,25,30,35),
                                    labels=c("F", "E","D","C","B","A")))%>%
  mutate(mpg_z=scale(mpg))
head(mtcars.2$mpg_z,n = 4L)
```

```
##           [,1]
## [1,] 0.1508848
## [2,] 0.1508848
## [3,] 0.4495434
## [4,] 0.2172534
```

1.13.3 Verilerin Sıralanması

Hazırlanan verilerin sıralanması için üç yöntem kullanılabilir. İlk olarak RStudio'ya eklenen özellikler ile verinin büyükten küçüğe veya tam tersi olarak sıralanması yapılabilir. Ancak bu özellik sadece bir değişken için kullanılabilmektedir.

İkinci yöntem `order()` fonksiyonu kullanılarak verilerin sıralanmasıdır. Fonksiyonda mevcut küçükten büyüğe doğru sıralama yapılırken, büyükten küçüğe sıralama için değişkenin önüne "-" işareti konulması veya "decreasing" özelliğinin "TRUE" olarak seçilmesi gereklidir.

```
order(..., na.last = TRUE, decreasing = FALSE,
      method = c("auto", "shell", "radix"))
```

Son olarak **dplyr** paketinde bulunan `arrange()` fonksiyonu ile değişkenler sıralanabilir. Eğer büyükten küçüğe sıralama yapılmak isteniyorsa değişkenin `desc()` fonksiyonu ile kullanılması gerekir.

```
require(dplyr)
attach(mtcars)
# Order fonksiyonu ile sıralama
mtcars_or <- mtcars[order(mpg,cyl),]
head(mtcars_or)
```

```
##                      mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
## Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
## Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
## Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
## Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
##                       mpgc
## Cadillac Fleetwood  1.3000
## Lincoln Continental 1.3000
## Camaro Z28          1.6625
## Duster 360          1.7875
## Chrysler Imperial   1.8375
## Maserati Bora       1.8750
```

```
#dplyr arrange fonksiyonu ile sıralama
head(arrange(mtcars, mpg, cyl))
```

```
##    mpg cyl disp  hp drat    wt  qsec vs am gear carb   mpgc
## 1 10.4   8  472 205 2.93 5.250 17.98  0  0    3    4 1.3000
## 2 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4 1.3000
## 3 13.3   8  350 245 3.73 3.840 15.41  0  0    3    4 1.6625
## 4 14.3   8  360 245 3.21 3.570 15.84  0  0    3    4 1.7875
## 5 14.7   8  440 230 3.23 5.345 17.42  0  0    3    4 1.8375
## 6 15.0   8  301 335 3.54 3.570 14.60  0  1    5    8 1.8750
```

1.13.4 Verilerin birleştirilmesi

Verilerin birleştirilmesi denildiğinde satır/sütunların birleştirilmesi veya iki ve daha çok veri setinin birleştirilmesi anlaşılabilir. Satır ve sütünların birleştirlmesi için `rbind()` ve `cbind()` fonksiyonları kullanılmaktadır. Farklı iki veri setinin birleştirilmesi için ise `merge()` fonksiyonu veya **dplyr** paketinde sunulan `join()` fonksiyonu kullanılabilir.

`merge()` fonksiyonunda *x,y* öğeleri birleştirilecek veri setleri veya veri nesnelerini, *by*, *by.x* ve *by.y* ise birleştirme için kullanıulacak sütunlara ait özellikleri tanımlamak için kullanılır.

```
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"),
      incomparables = NULL, ...)
```

Aşağıda sunulan örnekte data1 ve data2 veri setleri oluşturularak bu setler "id" değerleri kullanılarak birleştirilmişlerdir. Bu süreçte dikkat edilmesi gereken husus `merge()` fonksiyonu ile her iki veri setinde yer alan ortak değişkenler (bu örnekte id=3) birleştirilmiş ve diğerleri yeni veri setinin dışında tutulmuşlardır. Eğer ortak olmayan değişkenleri de eklemek istiyorsak fonksiyona "all=TRUE" seçeneğini de eklemek gerekmektedir.

```r
# Iki yeni veri seti olusturulması
data_1 <- data.frame(id = c(1,2,3), value = rnorm(3,1,0.2))
data_2 <- data.frame(id = c(3,4,5,6), value = rnorm(4,1,0.2))
print(data_1);print(data_2)
```

```
##   id     value
## 1  1 0.8237201
## 2  2 0.7567708
## 3  3 1.0252501
```

```
##   id     value
## 1  3 1.1416892
## 2  4 0.9333741
## 3  5 0.8541452
## 4  6 0.9962470
```

```r
#Merge fonksiyonu ile birlestirme
new_data<-merge(data_1, data_2, by = "id")
new_data_1<-merge(data_1, data_2, by = "id", all = TRUE)
new_data
```

```
##   id value.x  value.y
## 1  3 1.02525 1.141689
```

```r
new_data_1
```

```
##   id   value.x   value.y
## 1  1 0.8237201        NA
## 2  2 0.7567708        NA
## 3  3 1.0252501 1.1416892
## 4  4        NA 0.9333741
## 5  5        NA 0.8541452
## 6  6        NA 0.9962470
```

dplyr paketinde sunulan birleştirme seçenekleri de benzer özellikleri barındırmaktadır. `inner_join()` fonksiyonu birleştirilecek veri setlerinden eşleşenleri alırken, `left_join()` ve `right_join()` fonksiyonları sırasıyla birinci ve ikinci setteki tüm dğeişkenleri ve diğer sette yer alan bunlarla eşleşen değerleri alır ve sütunlara ilişkin eksik veri varsa "NA" değeri atayarak iki veri setinin sütunlarını birleştirir. Bunlara ek olarak `full_join()` fonksiyonu her iki veri setini tamamen birleştirirken `anti_join()` fonksiyonu ilk veri seti ve ikinci veri setindeki ortak satırları çıkartarak değerleri verir.

```r
inner_join(x, y, by = NULL)
left_join(x, y, by = NULL)
right_join(x, y, by = NULL)
full_join(x, y, by = NULL)
anti_join(x, y, by = NULL,)
```

`rbind()` ve `cbind()` fonksiyonları ise iki veri setini tamamen birleştirmek için kullanılır.

```
require(dplyr)
#Dplyr paketi fonksiyonlari ile veri setlerini birlestirme
inner_join(data_1,data_2,by="id")
```

```
##   id value.x  value.y
## 1  3 1.02525 1.141689
```

```
left_join(data_1,data_2,by="id")
```

```
##   id   value.x  value.y
## 1  1 0.8237201       NA
## 2  2 0.7567708       NA
## 3  3 1.0252501 1.141689
```

```
anti_join(data_1,data_2,by="id")
```

```
##   id     value
## 1  1 0.8237201
## 2  2 0.7567708
```

```
# Rbind fonksiyonu ile degiskenleri ekleme
rbind(data_1,data_2)
```

```
##   id     value
## 1  1 0.8237201
## 2  2 0.7567708
## 3  3 1.0252501
## 4  3 1.1416892
## 5  4 0.9333741
## 6  5 0.8541452
## 7  6 0.9962470
```

1.13.5 Verilerin seçilmesi

Verilerin seçilmesi için temel olarak iki özellik kullanılır. Bunlardan ilki veri setindeki belirlenen değişkenlerin seçilmesi, ikincisi ise verilerin belirli özelliklere göre filtrelenmesi yani süzülmesidir. Verilerin seçilmesi çok sayıda fonksiyon bulunmakta, bu nedenle, verilerin sınıfına göre kullanılabilecek fonksiyonlar değişmektedir.

Veri setindeki değişkenlerin seçilmesi için temel fonksiyonlardan ilki köşeli parantez `[]` veya iki köşeli parantez `[[]]` fonksiyonlarıdır. Bunlardan ilki liste olarak sonuçları verirken içeriğe ulaşmak için ikincisi kullanılır. Bu fonksiyonların daha detaylı anlaşılması için elimizde içinde paketlenmiş çikolataların olduğu bir çikolata kutusu olduğunu varsayalım. Köşeli parantez paketlenmiş her bir çikolatayı,iki köşeli parantez ise paketi çıkartılmış çikolatayı seçmemize imkan tanır. Aşağıda sunulan örnekte `mtcars` veri setinde yer alan değerlere köşeli parantez

kullanılarak nasıl ulaşılabileceği gösterilmiştir.

```
# Ikinci satırda yer alan değerleri verir
mtcars[2,]
# Ikinci sütunda yer alan değerleri verir
mtcars[,2]
# Ikinci satir, ikinci sütunda yer alan değerleri verir
mtcars[2,2]
# Ikinci sütundaki değerleri verir
mtcars[[2]]
```

Örnekte dikkat edilirse tek köşeli parantez içerisinde "," kullanılmıştır. Virgülden öncesi satır sonrası ise sütun kısmını ifade etmekte olup her ikisinin belirtilmesi halinde tek bir değer verecektir. Son komut ise veri setinin ikinci sütununun seçilmesini saglamaktadır, ancak bir öncekine göre farkı data.frame nesnesinin sütunlardan oluşan bir liste olması sebebiyle satır sayısını belirtmeye gerek kalmadan doğrudan ikinci sütunun seçilmesine imkan tanımasıdır. Köşeli parantez işleci özellikle az sayıda veri arken kullanışlı bir işleç olup bununla yapılabilecek veri seçme işlemleri Tablo 1.7 ile gösterilmektedir.

Tablo 1.7: Veri Seçme Fonksiyonları

Fonksiyon	Kullanımı
x[i]	Vektördeki i. elemanı gösterir
x[-i]	Vektördeki i hariç tüm elemanları gösterir
x[1:n]	Vektördeki ilk n elemanı gösterir
x[c(a,b,c)]	Vektördeki seçili a,b,c elemanlarını gösterir
x[x>k]	Vektörde k değerinden büyük elemanları gösterir
x[x>k | x < k]	Vektörde k değerinden büyük veya küçük elemanları gösterir

Verileri seçmeye yarayan baska bir işleçte $ işlecidir. Yeni değişken oluşturulmasında kısaca bahsedilen bu işleç, data.frame nesnesindeki değişkenlere isimleri ile erişime imkan tanır. Bu işleç köşeli parantez ile kullanılırsa o değişkenin n. sırasındaki elemanı verecektir.

```
# mpg degişkeninin değerlerini verir
head(mtcars$mpg)
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

```
# mpg degiskeninin besinci elemanini verir
mtcars$mpg[5]
```

```
## [1] 18.7
```

```
# mpg degiskeninin on ile onbes araligindaki elemanlari verir.
mtcars$mpg[10:15]
```

```
## [1] 19.2 17.8 16.4 17.3 15.2 10.4
```

Verilerin belirli özelliklere göre süzülmesi gerektiğinde üç yöntem kullanılabilir. Bunlardan ilki köşeli parantez kullanılarak süzme kriterlerinin belirlenmesi, ikincisi `subset()` fonksiyonun

kullanılması, üçüncüsü ise **dplyr** paketinde yer alan `filter()` ve `select()` fonksiyonlarının kullanılmasıdır.

`subset()` fonksiyonunda *x* öğesi seçilecek veri setini, *subset* seçme kriterlerini (mesela mpg > 40) ve *select* ise hangi değişkenlerin seçileceğini (birden fazla değişken seçilebilir) belirler. Burada kullanılabilecek kısayollardan birisi, subset fonksiyonunda seçilecek değişkenlerden ilki ile sonuncusu arasına ":" konularak aradaki tüm değişkenlerin seçilmesi sağlanabilir. `filter()` fonksiyonu da `subset()` fonksiyonu ile benzer şekilde kullanılabilir. `select()` fonksiyonunda ise `starts_with("X")` fonksiyonu X ile başlayanları, `ends_with("X")` fonksiyonu X ile bitenleri, `contains("X")` fonksiyonu X ifadesini içerenleri ve `matches("X")` fonksiyonu X paternine uyan değişkenleri seçmeye yarar.

```
subset(x, subset, select)
subset(airquality, Temp > 80, select = c(Ozone, Temp))
subset(airquality, Day == 1, select = -Temp)
subset(airquality, select = Ozone:Wind)

filter(data, ...)
filter(mtcars, cyl == 8)
filter(mtcars, cyl < 6)

select(data, ...)
```

Yukarıda belirtilen fonksiyonların örnekleri sırasıyla aşağıda gösterilmektedir. Örneklerden görüleceği üzere köşeli parantez ile süzme işlemleri bir noktadan sonra karmaşıklaşmaktadır. Ayrıca köşeli parantez ile süzme işleminde satır değerinden sonra, işareti konulmazsa satır yerine sütun değerlerini verecektir.

```
require(dplyr)
# mtcars veri setinde mpg > 20 olan satirlari süzer
# ve tüm sütunları seçer
mtcars.1<-mtcars[mtcars$mpg>20,]

# mtcars veri setinde mpg > 20 ve silindir sayısı 4 olan
# satırları seçer ve tüm sütunları gösterir.
mtcars.2<-mtcars[mtcars$mpg>20 & mtcars$cyl==4,]

# mtcars veri setinde mpg > 20 ve otomatik vitesli olan
# satırları seçer 3-5 sütunlarını verir.
mtcars.3<-mtcars[mtcars$mpg>20 & mtcars$cyl==4,c(3:5)]

# mtcars veri setinde mpg > 20 olan satırları süzer.
mtcars.4<-subset(mtcars, mpg>20)

# mtcars veri setinde mpg > 20 ve silindir sayısı 4 olan
# satırları süzer ve tüm sütunlari verir.
```

```
mtcars.5<-subset(mtcars, mpg >20 & cyl == 4 )

# mtcars veri setinde mpg > 20 veya silindir sayısı 4 olan
# satırları süzer ve tüm sütunları verir.
mtcars.6<-subset(mtcars, mpg >20 | cyl == 4 )

# mtcars veri setinde mpg > 20 ve silindir sayısı 4 olan
# satırları süzer ve mpg, cyl ve gear sütunlarını verir.
mtcars.7<-subset(mtcars, mpg >20 & cyl == 4, select =c(mpg, cyl,  gear))
# mtcars.7 veri setinin dplyr paketi ile hazırlanması
mtcars.8<-filter(mtcars,mpg >20 & cyl == 4) %>%
  select(., c(mpg, cyl, gear))
```

Verilein seçilmesi için `which` fonksiyonu da kullanılabilir. Bu fonksiyon verilerin belirli kriterleri taşıyıp taşımadığını göstermek için `logical` bir sonuç verebileceği gibi filtreleme için de kullanılır. Buna ilişkin aşağıda verilen ilk örnekte 1-20 arasında rastgele oluşturulan bir x değişkenin 10'dan küçük değerleri `which`fonksiyonu ile gösterilmiş, ikinci örnekte ise 1'den 102a kadar olan sayılardan mod(2)'ye göre olanlar seçilmiştir.

```
# which fonksiyonu örneği
set.seed(1234)
x <- sample(20)
which(x < 10)
```

```
## [1]  1  7  8  9 10 11 12 15 16
```

```
# ------------------

x1 <-  1:10 %% 2 == 0
which(x1)
```

```
## [1]  2  4  6  8 10
```

1.13.6 Verilerin özetlenmesi

R kullanılırken karşılaşılan en önemli sorunlardan birisi çok basit gereksinimler için bazı fonksiyonlara ulaşmanın zor olmadısıdır. Bu nedenle bazen uygulamada R'nin kolay işleri yapmak için çok karışık, karışık işleri yapmak içinse çok kolay olduğu değerlendirmesi yapılır. ÖZet verilere ulaşıp bunları tablo şeklinde sunmak da R için çok karmaşık işlerden biri haline gelmektedir. Mesela `mtcars` veri setinden üç değişkenin özet istatistiki değerlerni elde etmek için şöyle bir fonksiyon yazılması gerekir:

```
round(with(mtcars, t(sapply(mtcars[c("mpg", "disp",  "hp")],
                   function(x) c(n=length(x), avg=mean(x),
                   stdev = sd(x))))), 2)
```

Ancak geliştirilen paketlerle bu sorun ortadan kaldırılmıştır. Verilerin özetlenmesi için uygulanabilecek birkaç yöntem ve kullanılabilecek bir kaç paket bulunmaktadırbulunmaktadır. Bunlardan ilki `summary()` fonksiyonunun kullanılmasıdır. Bu fonksiyon ile veri setinde yer alan dğeişkenlerin temel açıklayıcı istatistikleri sunulmaktadır.

```
data(mtcars)
summary(mtcars[,1:4])
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
```

Diğer bir yöntem ise satır ve sütun değerlerinin toplamını ve ortalama değerlerini veren `colSums()` fonksiyon ailesidir. Bu fonksiyonlarda sütunların toplam ve ortalama değerleri için `colSums()` ve `colMeans()` fonksiyonları, satır toplamı ve ortalamaları için `rowSums()` ve `rowMeans()` fonksiyonları kullanılır. Söz konusu fonksiyonlarda x kullanılacak veri setini, *dims* özellikle dizilerde hangisinin satır (1) hangisinin sütun (2) olacağını belirtmek içn kullanılır

```
colSums (x, na.rm = FALSE, dims = 1)
rowSums (x, na.rm = FALSE, dims = 1)
colMeans(x, na.rm = FALSE, dims = 1)
rowMeans(x, na.rm = FALSE, dims = 1)
```

Mtcars veriseti kullanılarak her bir sütunun ortalama değerini hesaplamak için:

```
data(mtcars)
colMeans(mtcars)
```

```
##        mpg        cyl       disp         hp       drat         wt
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250
##       qsec         vs         am       gear       carb
##  17.848750   0.437500   0.406250   3.687500   2.812500
```

Ancak bu fonksiyonlar bir tablo için kullanılacaksa karışıklığa yol açabilir. Bu nedenle, **stargazer** paketi, elde edilecek sonuçları (hem özet hem de ileride gösterileceği şekilde modellerin sunulması) yayın kalitesinde (LATEX, html veya ASCII) olarak sunmaya imkan tanımaktadır (Hlavac 2015). `stargazer()` fonksiyonuna sadece veri seti yazıldığında özet değerleri hemen vermektedir. Bunun haricinde *type* ile yayın türü (text, html veya LATEX), *out* ile çıktının kaydedileceği dosya, *style* ile yayın için dergi stilleri ve daha bir çok özellik belirlenebilmektedir. Word dosyalarına bu tabloları eklemek için tablo html olarak kaydedilip söz konusu dosya Word ile açılarak kullanılabilir. Stargazer ile sağlanan diğer özellikler için stargazer[3] adresine veya paketin açıklama dosyasına bakılabilir.

[3]Stargazer Cheatsheet, http://jakeruss.com/cheatsheets/stargazer.html

```
stargazer(  ...,
            type = "latex", title = "", style = "default",
            summary = NULL, out = NULL, out.header = FALSE,
            column.labels = NULL, column.separate = NULL,
            covariate.labels = NULL, dep.var.caption = NULL,...)
```

```
require(stargazer)
stargazer(mtcars, median = TRUE, iqr = TRUE, type="text")
```

```
##
## ===============================================================================
## Statistic N   Mean   St. Dev.  Min   Pctl(25) Median  Pctl(75)   Max
## -------------------------------------------------------------------------------
## mpg       32 20.091   6.027   10.400  15.425  19.200   22.800  33.900
## cyl       32  6.188   1.786     4       4        6       8        8
## disp      32 230.722 123.939  71.100 120.825  196.300 326.000  472.000
## hp        32 146.688  68.563    52     96.5     123     180      335
## drat      32  3.597   0.535   2.760   3.080    3.695   3.920    4.930
## wt        32  3.217   0.978   1.513   2.581    3.325   3.610    5.424
## qsec      32 17.849   1.787   14.500  16.892  17.710   18.900  22.900
## vs        32  0.438   0.504     0       0        0       1        1
## am        32  0.406   0.499     0       0        0       1        1
## gear      32  3.688   0.738     3       3        4       4        5
## carb      32  2.812   1.615     1       2        2       4        8
## -------------------------------------------------------------------------------
```

Eğer tüm değişkenler değil de sadece bir kaç değişken seçilecekse bu durumda ufak bir düzeltmeye başvurulabilir. Aşağıda yer alan örnekte sadece hp, mpg ve disp değişkenlerine ait değerler `stargazer()` fonksiyonu ile gösterilmektedir.

```
require(wrapr)
require(stargazer)
stargazer(mtcars[qc(hp, mpg,disp)], median = TRUE, iqr = TRUE,
          type = "text")
```

```
##
## ===============================================================================
## Statistic N   Mean   St. Dev.  Min   Pctl(25) Median  Pctl(75)   Max
## -------------------------------------------------------------------------------
## hp        32 146.688  68.563    52     96.5     123     180      335
## mpg       32 20.091   6.027   10.400  15.425  19.200   22.800  33.900
## disp      32 230.722 123.939  71.100 120.825  196.300 326.000  472.000
## -------------------------------------------------------------------------------
```

Üçüncü yöntem `aggregate()` fonksiyonu kullanılarak değişkenlere belirli fonksiyonlar uygulanabilir. Burada x öğesi veri setini, *by* fonksiyonların uygulanacağı değişken listesi ve

FUN ile uygulanmak istenen fonksiyou tanımlamak için kullanır. Bu fonksiyonlar ortalama (mean), standart sapma (sd), medyan (median) vb. fonksiyonlar olabilir. Eğer eksik veriler özet dışında tutulacaksa *na.rm = TRUE* olarak seçilmelidir.

```
aggregate(x, by, FUN, ..., na.rm = FALSE, simplify = TRUE ...)
```

Yaygın kullanılan başka bir yöntem ise **dplyr** paketi ile sunulan `summarise()` ve `summarise_all()` fonksiyonlarının kullanılmasıdır. Bu fonksiyonlar ile verileri hem gruplamak hem de özet değişkenleri oluşturmak mümkündür. Ayrıca birden fazla özetleme fonksiyonu da bu fonksiyonlara eklenebilir. İki fonksiyon arasındaki fark ise, birincisinde bir değişkene ilişkin özet veriler alınırken ikincisi tüm değişkenlere bunu uygulamaktadır. `summarise()` fonksiyonunda birden fazla özetleme fonksiyonu kullanılacaksa bunların adlandırılması, `summarise_all()` fonksiyonunda birden fazla özetleme fonksiyonu kullanılacaksa bunların *funs* öğesinde ile tanımlanması gerekir.

```
summarise(.data, ...)
summarise_all(.tbl, .funs, ...)
```

```
require(dplyr)
# Temel aggregate fonksiyonu ile verilerin özetlenmesi.
# Bu örnekte mtcars veri seti slindir sayisi ve vites türüne göre
# gruplandırılarak her değişkenin ortalaması hesaplanmaktadır.

attach(mtcars)
aggdata <- aggregate(mtcars, by=list(cyl,am),
                     FUN = mean , na.rm = TRUE)
print(aggdata)
```

```
##   Group.1 Group.2      mpg cyl     disp        hp     drat       wt
## 1       4       0 22.90000   4 135.8667  84.66667 3.770000 2.935000
## 2       6       0 19.12500   6 204.5500 115.25000 3.420000 3.388750
## 3       8       0 15.05000   8 357.6167 194.16667 3.120833 4.104083
## 4       4       1 28.07500   4  93.6125  81.87500 4.183750 2.042250
## 5       6       1 20.56667   6 155.0000 131.66667 3.806667 2.755000
## 6       8       1 15.40000   8 326.0000 299.50000 3.880000 3.370000
##       qsec    vs am     gear     carb
## 1 20.97000 1.000  0 3.666667 1.666667
## 2 19.21500 1.000  0 3.500000 2.500000
## 3 17.14250 0.000  0 3.000000 3.083333
## 4 18.45000 0.875  1 4.250000 1.500000
## 5 16.32667 0.000  1 4.333333 4.666667
## 6 14.55000 0.000  1 5.000000 6.000000
```

```
detach(mtcars)

# Dplyr-summarize fonksiyon ile verilerin özetlenmesi.
# Bu örnekte mtcars veri seti slindir sayisi ve vites türüne göre
```

```
# gruplandırılarak  mpg değişkeninin oralaması ve standart
# sapması hesaplanmaktadır.

mtcars %>%
  group_by(.,cyl,am) %>%
  summarise(m_mpg = mean(mpg), sd_mpg = sd(mpg))
```

```
## # A tibble: 6 x 4
## # Groups:   cyl [?]
##     cyl    am m_mpg sd_mpg
##   <dbl> <dbl> <dbl>  <dbl>
## 1    4.    0.  22.9  1.45
## 2    4.    1.  28.1  4.48
## 3    6.    0.  19.1  1.63
## 4    6.    1.  20.6  0.751
## 5    8.    0.  15.0  2.77
## 6    8.    1.  15.4  0.566
```

```
# Dplyr-summarize fonksiyon ile verilerin özetlenmesi.
# Bu örnekte mtcars veri seti silindir sayısı ve vites türüne göre
# gruplandırılarak tüm değişkenlerin ortalaması ve standart sapması
# hesaplanmaktadır.
mtcars %>%
  group_by(.,cyl,am) %>%
  summarise_all(funs(mean,sd))
```

```
## # A tibble: 6 x 20
## # Groups:   cyl [?]
##     cyl    am mpg_mean disp_mean hp_mean drat_mean wt_mean qsec_mean
##   <dbl> <dbl>    <dbl>     <dbl>   <dbl>     <dbl>   <dbl>     <dbl>
## 1    4.    0.     22.9     136.     84.7      3.77    2.94      21.0
## 2    4.    1.     28.1      93.6    81.9      4.18    2.04      18.4
## 3    6.    0.     19.1     205.    115.       3.42    3.39      19.2
## 4    6.    1.     20.6     155.    132.       3.81    2.76      16.3
## 5    8.    0.     15.0     358.    194.       3.12    4.10      17.1
## 6    8.    1.     15.4     326.    300.       3.88    3.37      14.6
## # ... with 12 more variables: vs_mean <dbl>, gear_mean <dbl>,
## #   carb_mean <dbl>, mpg_sd <dbl>, disp_sd <dbl>, hp_sd <dbl>,
## #   drat_sd <dbl>, wt_sd <dbl>, qsec_sd <dbl>, vs_sd <dbl>, gear_sd <dbl>,
## #   carb_sd <dbl>
```

Son olarak **skimr** paketi de verilerin özetlenmesi için çeşitli fonksiyonlar sunmaktadır. Bu pakette yer alan `skim()` fonksiyonu da değişkenlerin temel özelliklerini ve histogramını sunmaktadır. Bu fonksiyon ile *skim_df* nesnesi elde edilir ve bu nesne `kable()` fonksiyonu ile tablo şeklinde hazırlanabilir.

```
require(skimr)
data(mtcars)
skim(mtcars)
```

1.13.7 Veri setinin dönüştürülmesi

Uygulamada veri setleri iki şekilde, 1)Uzun (Long) ve 2) Geniş (Wide) veri setleri olarak karşımıza çıkar. Uzun veri setleri sırasıyla her bir değişkenin her birim için değerleri alt alta barındırdığı veri setleridir. Geniş veri setleri ise her bir birim için değişkenlerin ayrı ayrı sıralandığı veri setleridir. Uygulamada sağladığı kolaylıklar açısından uzun veri setleri geniş veri setlerine tercih edilmektedir. R'de veri setlerinin geniş veya uzun hale getirilmesi için **tidyr** paketi kullanılmakta olup, fonksiyonlarının çeşitliliği nedeniyle de bu paketin kullanımı anlatılacaktır (Wickham ve Henry 2017).

tidyr paketinde verilerin dönüştürülmesi için iki temel fonksiyon sunulmaktadır: `gather()` ve `spread()` . `gather()` fonksiyonu çok sayıda sütunu tek veya birkaç sütun haline getirirken, `spread()` fonksiyonu tek sütundaki değerleri çoklu hale getirmek için kullanılır. Bu fonksiyonlarda *data* öğesi veri setini, *key* oluşturulacak veya dağıtılacak sütun için kullanılacak isimi, *value* oluşturulacak değer sütunu için verilecek değişken adını ve *convert* değerlerin sınıflarında yapılacak değişikliği (TRUE olarak seçilir) belirtmek için kullanılır.[4] Her değişken için aynı olan ve sıralamada esas teşkil edecek değişkenin önüne "-" işareti konularak fonksiyona eklenir.

```
gather(data, key, value, ..., na.rm = FALSE, convert = FALSE,
  factor_key = FALSE)

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE,
  sep = NULL)
```

Aşağıda yer alan örnekte A,B,C firmalarına ait hisse senetleri için günlük rassal olarak değerler oluşturularak elde edilen geniş veri seti uzun hale getirilmiştir.

```
# A,B,C olarak üç tane hisse oluşturulmuş ve daha sonra
# uzun veri seti haline getirilmiştir.
require(tidyr)
set.seed(23)
hisse <- data.frame(
   gun = as.Date('2017-01-01') + 0:9,
   A = as.integer(rnorm(10, 5, 1)),
   B = as.integer(rnorm(10, 10, 1)),
   C = as.integer(rnorm(10, 15, 1)))
head(hisse)
```

```
##          gun A  B  C
```

[4] Bazı durumlarda birden fazla sınıfın tek bir sütuna dönüştürülmesi gerekebilir-mesela date, character vs. Bu durumda "convert" özelliğinin kullanılması gerekir.

```
## 1 2017-01-01 5 10 15
## 2 2017-01-02 4  8 14
## 3 2017-01-03 5  9 15
## 4 2017-01-04 6 10 13
## 5 2017-01-05 5  8 14
## 6 2017-01-06 6 10 14
```

```
hisse_uzun <- gather(hisse,-gun,key = Hisse,value = Deger)
head(hisse_uzun)
```

```
##          gun Hisse Deger
## 1 2017-01-01     A     5
## 2 2017-01-02     A     4
## 3 2017-01-03     A     5
## 4 2017-01-04     A     6
## 5 2017-01-05     A     5
## 6 2017-01-06     A     6
```

```
# Uzun veri seti geniş hale getirilmiştir

hisse_genis <- spread(hisse_uzun,key = Hisse,value = Deger)
head(hisse_genis)
```

```
##          gun A  B  C
## 1 2017-01-01 5 10 15
## 2 2017-01-02 4  8 14
## 3 2017-01-03 5  9 15
## 4 2017-01-04 6 10 13
## 5 2017-01-05 5  8 14
## 6 2017-01-06 6 10 14
```

1.13.8 Karakterlerin düzenlenmesi

Karakterlerin düzenlenmesi için **stringr** paketi çeşitli fonksiyonlar sunmaktadır. Özellikle Türkçe karakterlerde yaşanabilecek sorunlar için bu fonksiyonlar oldukça kullanışlıdır (Wickham 2016).

Bu fonksiyonlardan ilki karakterlerin büyük veya küçük harfli olarak yazılmasını sağlayan `str_to_upper()`, `str_to_lower()` ve `str_to_title()` fonksiyonlarıdır. Burada *string* dönüştürülecek karakter grubunu, *locale* ise dil grubunu belirtmekiçin kullanılır. Eğer *locale* tanımlanmazsa karakter grubu İngilizce olarak seçilecektir. Türkçe karakterler için *locale="tr"* olarak tanımlanmalıdır.

```
str_to_upper(string, locale = "")
str_to_lower(string, locale = "")
str_to_title(string, locale = "")
```

```
str_to_upper("imtina", "tr")
str_to_upper("imtina", "tr")
```

Karakter sayısını göstermek için `str_count()` fonksiyonu kullanılır. Bu fonksiyonda *pattern* öğesi sayılacak harf veya harf grubunu belirlemeye yarar ve söz konusu karakter grubunda kaç tane olduğunu gösterir. `str_length()` fonksiyonu da karakter grubundaki karakter sayısını verir. Buna benzeyen bir fonksiyon da `str_detect()` fonksiyonudur, ancak bu sayı yerine bakılan değerin karakter grubunda olup olmadığını gösterir. Eğer aranan karakter grubu (pattern) söz konusu harf grubundan çıkartılmak isteniyorsa `str_extract()` fonksiyonu kullanılır. Gruplarda yer alan karakterlerin konumlarına bakılmak isteniyorsa `str_locate()` fonksiyonu kullanılır.

```
require(stringr)
meyve <- c("armut", "muz", "havuc", "portakal", "kivi", "ananas", "mango")
str_count(meyve)
```

```
## [1] 5 3 5 8 4 6 5
```

```
str_length(meyve)
```

```
## [1] 5 3 5 8 4 6 5
```

```
str_count(meyve, pattern="mu")
```

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

```
str_count(meyve, pattern="a")
```

```
## [1] 1 0 1 2 0 3 1
```

```
# İçinde a olan karakter gruplarını tespit eder.
str_detect(meyve, pattern="a")
```

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

```
# İçinde mu olan karakter gruplarını çıkartır.
str_extract(meyve,"mu")
```

```
## [1] "mu" "mu" NA   NA   NA   NA   NA
```

```
# İçinde mu olan karakter gruplarının konumlarını gösterir.
str_locate(meyve,"mu")
```

```
##      start end
## [1,]     3   4
## [2,]     1   2
## [3,]    NA  NA
## [4,]    NA  NA
## [5,]    NA  NA
```

```
## [6,]    NA  NA
## [7,]    NA  NA
```

Son olarak karakter gruplarında değişiklik yapılmak isteniyorsa bu birkaç fonksiyon ile yapılabilir. `str_replace()` fonksiyonu karakter grubunda yer alan ve *pattern* öğesi ile tanımlanan bir veya birkaç karakterin, *replacement* öğesiile tanımlanan başka bir grupla değiştirilmesini sağlar. `str_replace_na()` ise NA değeri olan elemanları "NA" şeklinde karaktere dönüştürür. `str_split()` fonksiyonu bir karakter grubunu belli bir yapıya göre bölerken `str_sub()` fonksiyonu da belirli aralıktaki değerlerin alınmasını sağlar. Bu fonksiyonda *start* öğesi kaçıncı karakterden itibaren başlanacağı ve *end* öğesi kaçıncı karaterden itibaren bitirileceğini gösterir. Eğer *start* ve *end* değerleri başta "-" olarak yazılırsa bu durumda sondan başa doğru sayarak ayırmayı yapar. `str_trim()` ise grubun başındaki (left) veya sonundaki (right) veya her iki taraftaki (both) boşlukları kaldırmaya yarar.

```
str_replace(string, pattern, replacement)
str_replace_na(string, replacement = "NA")

str_split(string, pattern, n = Inf)

str_sub(string, start = 1L, end = -1L)

str_trim(string, side = c("both", "left", "right"))
```

stringr paketinde bunlar haricinde kullanılabilecek çok sayıda fonksiyon olup, paketin yardım dosyasından fonksiyonların açıklamalarına bakılabilir.

1.14 R'de Programlama

R'nin temel özellikleri kullanışlı olsa da çalışmalar uzadıkça ve karmaşık hale geldikçe programlama kapasitesinden faydalanarak belirli fonksiyonları kulllanmak zaman ve kaynak tasarrufu sağlayacaktır. Programlama teknikleri bir kaç değişkenden oluşan veri setlerinden milyonlarca verinin kullanıldığı çok kapsamlı veri setlerine kadar karşılaşılabilecek her durum için kullanıcıya önemli kolaylıklar sağlamaktadır.

Bu bölümde çalışmalarda ihtiyaç duyulan ve programlamanın temeli olan bazı komutlar ve fonksiyonlar anlatılacak ve kısaca fonksiyon yazma ve programlama sürecinde karşılaşılabilecek hataların nasıl giderileceğine değinilecektir. Bu konularda daha detaylı bilgiler için Matloff (2011) ve Wickham (2014) kitaplarından faydalanılabilir. Ayrıca **pryr** paketi de programlama ile ilgili çok sayıda fonksiyon sunmaktadır (Wickham 2018).

R ile programlamanın temel konularına geçmeden önce **magrittr** paketi ile sunulan *fpo* (%>%) işleci hakkında biraz daha detaylı bilgi vermek yerinde olacaktır. Bu işleç kodların yazımını sadeleştirerek işlecin sol tarafında yer alan değerleri sağ tarafında yer alan fonksiyonlar ile kullanılmasını sağlar. Mesela `mtcars` veri setinde yer alan *gear* ve *mpg* değişkenlerinin

ortalaması hesaplanmak istendiğinde aşağıda yer alan kod yazılır. Burada görüldüğü üzere veri seti alınarak iki değişken seçilmiş ve bunların ortalaması hesaplanmıştır. Bu işlecin en önemli avantajlarından birisi bu değerler hesaplanırken veri setinde herhangi bir değişiklik yapılmamıştır. İşlecin eklenmesi için `Ctrl + Shift + M` kısayolu kullanılabilir.

```
require(tidyverse)
mtcars %>%
  select(gear,mpg) %>%
  apply(2,mean)
```

```
##     gear      mpg
##  3.68750 20.09062
```

Bu işlecin tek handikapı ise tek yönlü olarak çalışması yani işlecin tüm işlemleri bir sırada yapmasıdır. Mesela, örnekte değişkenlerin grafikle gösterilmesi istendiğinde `apply()` fonksiyonundan sonra `plot()` fonksiyonu eklenirse sadece *gear* ve *mpg*'nin ortalama değerleri gösterilecek ve iki değerin olduğu bir grafik verilecektir.

```
require(tidyverse)
mtcars %>% select(gear,mpg) %>% apply(2,mean) %>% plot()
```

Bu sorunu çözmek için `%T>%` işleci kullanılır. Dikkat edilirse bu işleç fonksiyonların uygulanacağı veriden sonra kullanılarak ve iki fonksiyonun ayrı ayrı uygulanması sağlamaktadır.

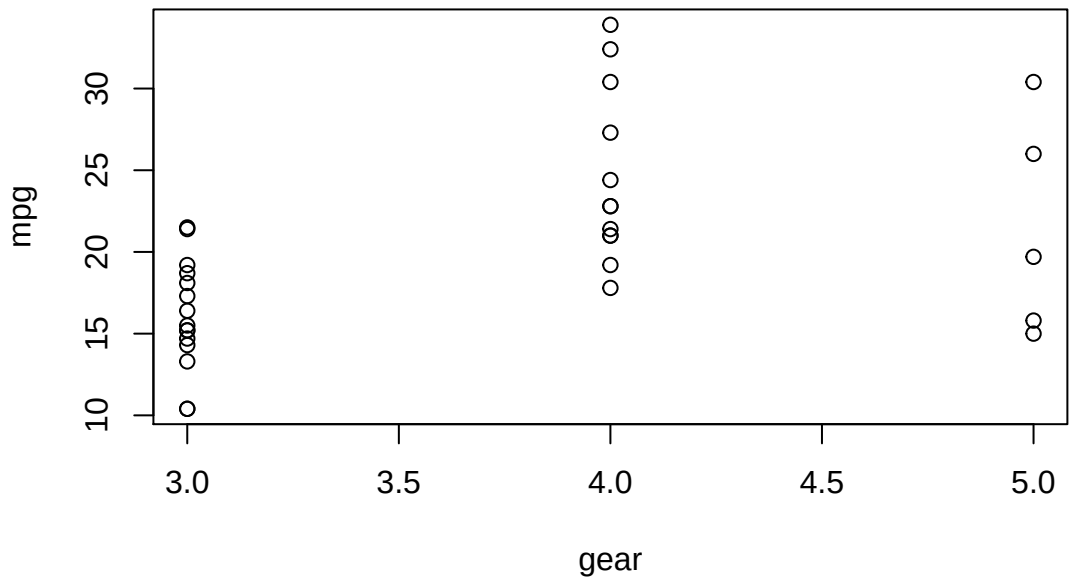
```
require(tidyverse)
data(mtcars)
mtcars %>%
  select(gear,mpg) %T>%
  plot() %>%
  apply(2,mean)
```

```
##     gear      mpg
##  3.68750 20.09062
```

Programlamada değinilmesi gereken önemli başka bir husus, fonksiyonların özellikleridir. R'de fonksiyonlar *başa yazılan* (prefix) ve *araya eklenen* (infix) olmak üzere iki kategoride değerlendirilebilir. Fonksiyonların önemli bir kısmı, fonksiyon yazma kısmında detaylı bir şekilde gösterileceği üzere, özelliklerinden önce gelir ve önce fonksiyon adı sonra da özellikleri şeklinde yazılır. Ancak `infix` fonksiyonlarda ise durum biraz farklıdır. Bu fonksiyonlar fonksiyon argümanlarının arasına eklenerek işlemlerin yapılmasını sağlar. Özellikle matematiksel işlemlerde bunların önemli bir kısmı kullanılmakta olup, Tablo 1.8 R'de tanımlı `infix` fonksiyonlarını göstermektedir.

Bu fonksiyonlar haricinde `+`, `-`, `*` gibi işleçlerde %% olmadan kullanılabilir. Tanımlı fonksiyonlar haricinde kullanıcılar tarafından da `infix` fonksiyonları tanımlanabilir.

```
# Infix fonksiyonu örneği
5 %+% 8
5  + 8
```



Şekil 1.1: T işleci grafik örneği

Tablo 1.8: Infix Fonksiyonları

Fonksiyon	Açıklaması
%%	Kalan değeri gösterir
%/%	Tam sayılı bölme
%*%	Matris çarpımı
%o%	Matris dış çarpımı
%x%	Matris Kronecker çarpımı
%in%	Eşleştirme fonksiyonu

```
## [1] 13
```

1.14.1 If ve ifelse komutları

R'de en yaygın kullanılan programlama komutu `if-else` ve `ifelse()` fonksiyonlarıdır. Bu fonksiyonlar belirli koşulların gerçekleşmesi veya gerçekleşmemesi halinde diğer fonksiyonların çalışmasını sağlamak için kullanılır.

`if-else` fonksiyonu ayrı olarak kullanılabileceği gibi `ifelse()` olarak bütünleşik de kullanılabilir. İlk durumda *if* fonksiyonunun ardından gerçekleşmesi gereken koşul ve uygulanacak fonksiyon belirtildikten sonra *else* fonksiyonu ve koşulun gerçekleşmemesi halinde uygulanacak fonksiyon belirtilir. `ifelse()` fonksiyonunda ise gerçekleşmesi gereken koşul *test* olarak

belirtildikten sonra uygulanacak fonksiyon *yes* ve gerçekleşmemesi durumunda uygulanacak fonksiyon *no* öğelerinde yazılır.

```
if(cond) cons.expr  else  alt.expr

ifelse(test, yes, no)
```

`If-else` fonksiyonuna verilecek en basit örnek, bir önceki başlıkta `cut()` fonksiyonu ile yapılan örneğin tekrar edilmesidir. Örnekte `mtcars` veri setinde mpg değişkenine göre altı verimlilik sınıfı oluşturulmuştur.

```
require(dplyr)
data(mtcars)
head(mtcars%>%
  mutate(.,verim = ifelse( mpg <= 10 ,"F",ifelse(
    mpg >11 & mpg <= 15,"E",ifelse(
     mpg > 15 & mpg <=20,"D",ifelse(
      mpg > 20 & mpg <=25,"C",ifelse(
       mpg > 25 & mpg <=30,"B","A")))))))
```

```
##    mpg cyl disp  hp drat    wt  qsec vs am gear carb verim
## 1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4     C
## 2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4     C
## 3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1     C
## 4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1     C
## 5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2     D
## 6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1     D
```

1.14.2 Döngüler

Her türlü bilgisayar programının ilk ayağını döngüler oluşturmaktadır. Döngüler programda tekrarlanması istenilen fiilerin belirli koşullar gerçekleşmesi halinde başlamasını ve belirli koşulların gerçekleşmesi halinde sona erdirilmesini sağlayarak maliyet ve işlem zamanından tasarruf edilmesini sağlar. Her döngüde üç temel unsur bulunmaktadır: (1) Başlangıç değeri, (2) Hareket değeri ve (3) Bitiş Değeri.

R'nin temel versiyonunda üç döngü sunulmakta olup, çeşitli paketlerle yeni döngüler eklenmiştir. Bu döngüler:

1. **For** döngüsü
2. **While** döngüsü
3. **Repeat** döngüsü

`For` döngüsü bir nesnenin başlangıç değerinden bitiş değerine kadar belirlenen değerleri almasını sağlamak için kullanılır. Bu döngüde *var* döngüde değişken olarak kullanılacak değeri, *seq* bu değerin başlangıç ve bitiş değerlerini, *expr* ise kullanılacak fonksiyonları

belirtmeyi sağlar. Eğer birden fazla öngü içiçe kullanılırsa iç içe döngüler (nested loops) oluşturulmuş olur.

```
for(var in seq) expr
```

Aşağıda yer alan örnekte `mtcars` veri setindeki değerlerin normalleştirilmiş değerlerinin (z-değerleri) hesaplanması için bir `for` döngüsü oluşturulmuş ve bu değerler "for.mtcars_normalized" isimli bir veri setine kaydedilmiştir. Döngünün başlangıç ve bitiş değerleri için veri setinin satır ve sütun boyutları(sat/sut değerleri) kullanılmış ve "data.frame" kullanılarak veri seti oluşturulmuştur. İlk döngü veri setinin ortalama ve standart sapma değerleri hesaplamak için kullanılmış, ikinci döngüde ise oluşturulan veri setinde j. satırındaki i. sütun değerinden o sütun için hesaplanan ortalama değer çıkartılarak stndart sapmaya bölünmüştür. Son olarak yeni veri setindeki değişken isimleri mtcars veri setinin isimleri ile düzenlenmiştir.

```
sat <- nrow(mtcars)
sut <- ncol(mtcars)
for.mtcars_normalized <- data.frame(1:sat)
for (i in 1:sut) {
  ort <- mean(mtcars[,i])
  sapma <- sd(mtcars[,i])
  for (j in 1:sat) {
    for.mtcars_normalized[j,i] <- (mtcars[j,i] - ort) / sapma
  }
}
names(for.mtcars_normalized) <- names(mtcars)
head(for.mtcars_normalized, n=5L)[1:5]
```

```
##          mpg        cyl       disp         hp       drat
## 1  0.1508848 -0.1049878 -0.5706198 -0.5350928  0.5675137
## 2  0.1508848 -0.1049878 -0.5706198 -0.5350928  0.5675137
## 3  0.4495434 -1.2248578 -0.9901821 -0.7830405  0.4739996
## 4  0.2172534 -0.1049878  0.2200937 -0.5350928 -0.9661175
## 5 -0.2307345  1.0148821  1.0430812  0.4129422 -0.8351978
```

`While` ve `Repeat` döngüleri de `for` döngüsü gibi oluşturulur. Ancak `for` döngüsünden farklı olarak bir koşulun gerçekleşmesine bağlı olarak döngüyü devam ettirirler. `Repeat` döngüsü genelde `if-else` fonksiyonu ile birlikte kullanılır ve döngüyü bitirmek için `break` komutunun eklenmesi gerekir. `next` index{next} komutu ise bir döngü tamamlandıysa veya tamamlanmadan kesildi ise bir sonraki aşamaya geçilmesi için kullanılır.

```
while(cond) expr
repeat expr
break
next
```

Ancak döngüler veri işlemleri için kullanılacaksa bilgisayar kaynaklarını yoğun bir şekilde

kullanmaları ve işlem süresini uzatmalarından dolayı pek tercih edilmez. Bunlar yerine bir sonraki başlıkta anlatılacak olan `apply` fonksiyonları veya **dplyr** paketinde sunulan fonksiyonlar tercih edilmektedir.

1.14.3 Apply fonksiyonları

Apply ailesinin ilk fonksiyonu `apply()` fonksiyonudur. Apply fonksiyonu seçilecek fonksiyonların bir dizi (array) ve matrise uygulanmasına imkan tanıyarak sonucun bir vektör olarak verilmesine imkan tanır. Bu fonksiyonda X fonksiyon uygulanacak dizi veya matrisi, *margin* fonksiyonun satırlara (1) veya sütunlara (2) veya her ikisine (c(1,2)) uygulanmasını, *FUN* ise uygulanacak fonksiyonu belirtmeyi sağlar.

```
apply(X, MARGIN, FUN, ...)
```

Aşağıda yer alan örnekte 5x6'lık matris oluşturularak satır ve sütunlarının toplamı `apply()` fonksiyonu ile gösterilmiştir.

```
# 1den 30'a kadar sayılar kullanılarak 5*6lik matris olustur.
matris.1<-matrix(1:30, nrow = 5)
# Matrisin satırlarının toplamını verir
apply(matris.1, 1, sum)
```

```
## [1]  81  87  93  99 105
```

```
# Matrisin sütunlarının toplamını verir.
apply(matris.1, 2, sum)
```

```
## [1]  15  40  65  90 115 140
```

Apply ailesinin ikinci fonksiyonu `lapply()` fonksiyonudur. Bu fonksiyon seçilen fonksiyonun vektör,liste ve data.frame sınıfı verilere uygulanmasını ve sonucun liste olarak verilmesini sağlar. Üçüncü fonksiyon ise `sapply()` fonksiyonudur. Bu da `lappy()` fonksiyonu ile aynı işlevleri görmekte fakat sonuç olarak liste olarak değil vektör olarak gösterilmektedir. Eğer sonuçlar `lapply()` gibi gösterilmek istenirse *simplified* = *FALSE* olarak belirtilmelidir.

```
lapply(X, FUN, ...)
sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

Aşağıda yer alan örneklerden birincisinde `mtcars` veri seti kullanılarak her bir değişkenin ortalama değeri hesaplanmış ve "list" sınıfı olarak değerler verilmiştir.

```
# Birinci örnek: Lapply fonksiyonu
mtcars_mean <- lapply(mtcars,mean)
head(mtcars_mean)[1:5]
```

```
## $mpg
## [1] 20.09062
##
```

```
## $cyl
## [1] 6.1875
##
## $disp
## [1] 230.7219
##
## $hp
## [1] 146.6875
##
## $drat
## [1] 3.596563
```

```
mtcars_scale <- data.frame(lapply(mtcars, scale))
head(mtcars_scale)[1:5]
```

```
##          mpg        cyl        disp         hp       drat
## 1  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## 2  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## 3  0.4495434 -1.2248578 -0.99018209 -0.7830405  0.4739996
## 4  0.2172534 -0.1049878  0.22009369 -0.5350928 -0.9661175
## 5 -0.2307345  1.0148821  1.04308123  0.4129422 -0.8351978
## 6 -0.3302874 -0.1049878 -0.04616698 -0.6080186 -1.5646078
```

```
mtcars_normalized <- data.frame(lapply(mtcars, function(x){(x-mean(x))/sd(x)}))
head(mtcars_normalized)[1:5]
```

```
##          mpg        cyl        disp         hp       drat
## 1  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## 2  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## 3  0.4495434 -1.2248578 -0.99018209 -0.7830405  0.4739996
## 4  0.2172534 -0.1049878  0.22009369 -0.5350928 -0.9661175
## 5 -0.2307345  1.0148821  1.04308123  0.4129422 -0.8351978
## 6 -0.3302874 -0.1049878 -0.04616698 -0.6080186 -1.5646078
```

İkinci örnekte ise z-değerleri hesaplanmış ve bunlar data.frame sınıfı olarak sunulmuştur.

```
# İkinci örnek: Sapply ve scale fonksiyonları
mtcars_s_mean <- sapply(mtcars,mean)
head(mtcars_s_mean, n=5L)
```

```
##        mpg        cyl       disp         hp       drat
##  20.090625   6.187500 230.721875 146.687500   3.596563
```

```
mtcars_s_scale <- data.frame(sapply(mtcars, scale))
head(mtcars_s_scale)[1:5]
```

```
##          mpg        cyl        disp         hp       drat
## 1  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
```

```
## 2  0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## 3  0.4495434 -1.2248578 -0.99018209 -0.7830405  0.4739996
## 4  0.2172534 -0.1049878  0.22009369 -0.5350928 -0.9661175
## 5 -0.2307345  1.0148821  1.04308123  0.4129422 -0.8351978
## 6 -0.3302874 -0.1049878 -0.04616698 -0.6080186 -1.5646078
```

Üçüncü örnekte ise ikinci örnekteki uygulama `scale()` fonksiyonu ile aynı işlevi gören bir fonksiyon marifetiyle yapılmıştır.

```
# Üçüncü örnek: Sapply ve kullanıcı tarafından
# yazılan bir fonksiyonun kullanılması
mtcars_s_normalized <- data.frame(sapply(mtcars,
                                         function(x){
                                           (x - mean(x))/sd(x)}))
head(mtcars_s_normalized, n = 5L)[1:5]
```

```
##          mpg        cyl       disp         hp       drat
## 1  0.1508848 -0.1049878 -0.5706198 -0.5350928  0.5675137
## 2  0.1508848 -0.1049878 -0.5706198 -0.5350928  0.5675137
## 3  0.4495434 -1.2248578 -0.9901821 -0.7830405  0.4739996
## 4  0.2172534 -0.1049878  0.2200937 -0.5350928 -0.9661175
## 5 -0.2307345  1.0148821  1.0430812  0.4129422 -0.8351978
```

Apply ailesinin dördüncü fonksiyonu `mapply()` fonksiyonudur. Bu ise bir fonksiyonun bir değer grubuna birden fazla uygulanmasını sağlar. *FUN* uygulanacak fonksiyonu, *X* uygulanacak değerleri, *times* tekrar sayısını belirtmek için kullanılır. Eğer tekrar sayısına birden fazla değer yazılırsa birinci elemandan başlayarak giderek artan (veya azalan) şekilde X değerlerini uygular. Eğer *X*'te birden fazla değer belirlendiyse oluşturulacak vektörün ilk elemanına ilk *X* değerini, ikinci elemanına ikinci *X* değerini uygulayarak devam eder.

```
mapply(FUN, times, X, MoreArgs = NULL, SIMPLIFY = TRUE,
       USE.NAMES = TRUE)
```

Aşağıda sunulan örneklerde `rep` fonksiyonunun `mapply()` ile farklı şekillerde uygulanması gösterilmektedir.

```
# 5 defa 5 rakamını tekrarlar ve 5*1'lik bir data.frame olusturur.
mapply(rep, times=5, x=5)
# 1-5 arası rakamları beş defa tekrarlar,
# 5*5 lik bir data.frame oluşturur.
mapply(rep, times = 5, x=1:5)
# 1-5 arası rakamları azalan bir şekilde beş defa tekrarlar.
#5*5 lik bir data.frame olusturur.
mapply(rep, times = 5, x=5:1)
# 5 rakamını 1'den baslayarak 5'e kadar artan bir şekilde tekrar eder.
mapply(rep, times = 1:5, x=5)
# 5 rakamını 5 ten baslayarak 1'e kadar azalan bir sekilde tekrar eder.
mapply(rep, times = 5:1, x=5)
```

```
# 1-5 arası rakamları 1'den baslayarak 5'e kadar
# artan bir sekilde tekrar eder.
mapply(rep, times = 1:5, x=1:5)
# 1-5 arası rakamları 5'ten baslayarak 1'e kadar
# azalan bir sekilde tekrar eder.
mapply(rep, times = 1:5, x=5:1)
```

Bunlar haricinde "apply" fonksiyonları ailesinde `tapply()`, `vapply()`, `rapply()`, `eapply()` fonksiyonları da bulunmaktadır. Bu fonksiyonlara ilişkin detay bilgilere R programlama kitaplarından bakılabilir.

Döngüler için kullanılabilecek başka bir araç da **foreach** paketidir (Calaway ve Weston 2015). Bu paket döngüler için sağladığı bazı kolaylıkların yanında çok işlemcili bilgisayarlarda paralel işlem yaparak işlemleri hızlandırdığından kısaca değinilmesi yerinde olacaktır. **foreach** paketinde yer alan `foreach()` fonksiyonu tanımlanan bir fonksyonu belirtilen sayıda tekrar ederek sonuçlarını liste olarak verir. ".combine" seçeneği ile söz konusu değerlerin vektör veya "data.frame" olarak da sunulması sağlanabilir.

Bu fonksiyon en temel haliyle `foreach(X, .combine='') %do% FUN` olarak yazılabilir. Burada *X* fonksiyona uygulanacak tekrar edilecek değişkenleri belirtirken *.combine* çıktıların nasıl birleştirileceğini (`c()`veya `cbind()` fonksiyonları veya `+` , `*` gibi aritmetik işlemler), `%do%` uygulama komutunu ve *FUN* ise değişkenlerin uygulanacağı fonksiyonu tanımlamak için kullanılır. *FUN* kısmında yer alacak fonksiyon, tanımlanmış fonksiyonlardan biri olabileceği gibi kullanıcı tarafından da tanımlanan fonksiyonda olabilir.

```
foreach(X, .combine, ...) %do% FUN
```

Aşağıda yer alan örneklerde **foreach** paketi ve fonksiyonu ile yapılan bazı uygulamalar gösterilmektedir. Kodlar kullanılırken `c()` ve `cbind()` fonksiyonlarının çıktılara etkisine ve son üç örnekte `foreach()` fonksiyonunda tanımlanmış değerlerin uygulama fonksiyonunda kullanılmadığına dikkat edilmesi gerekir.

```
require(foreach)
set.seed(12)
#1-5 arası Ab ve B değerlerinin çarpımı
foreach(a=1:5,b=1:5, .combine='c') %do% {a*b}
```

```
## [1]  1  4  9 16 25
```

```
#1-5 arası Ab ve B değerlerinin çarpımının cbind ile gösterilmesi
foreach(a=1:5,b=1:5, .combine='cbind') %do% {a*b}
```

```
##      result.1 result.2 result.3 result.4 result.5
## [1,]        1        4        9       16       25
```

```
# 1-100 arası değerlerin karelerinin alınırken toplam gösterimin
# 10 ile sınırlandırılması
foreach(a=seq(1,100, length=10), .combine='c') %do% {a^2}
```

```
##  [1]     1   144   529  1156  2025  3136  4489  6084  7921 10000
```

```
#Rassal olarak 5x5 lik sayı elde edilmesi
foreach(a=1:5, .combine='cbind') %do% {rnorm(5)}
```

```
##        result.1   result.2    result.3   result.4   result.5
## [1,] -1.4805676 -0.2722960 -0.77771958 -0.7034643  0.2236414
## [2,]  1.5771695 -0.3153487 -1.29388230  1.1888792  2.0072015
## [3,] -0.9567445 -0.6282552 -0.77956651  0.3405123  1.0119791
## [4,] -0.9200052 -0.1064639  0.01195176  0.5069682 -0.3024592
## [5,] -1.9976421  0.4280148 -0.15241624 -0.2933051 -1.0252448
```

```
#Rassal olarak 5x5 lik sayı elde edilmesi, c() ile gösterimi
foreach(a=1:5, .combine='c') %do% {rnorm(5)}
```

```
##  [1] -0.2673848 -0.1991057  0.1311226  0.1457999  0.3620647  0.6739812
##  [7]  2.0720358 -0.5410286 -1.0704922 -0.3724567 -0.4851414  0.2747842
## [13] -0.4795126  0.7981053 -1.0044512  0.1049842 -1.1559929  0.5781346
## [19] -1.5956257 -0.3085037  0.4494659 -0.9770533  0.1899979  0.7314534
## [25] -0.4925991
```

```
foreach(a=5, .combine='c') %do% {rnorm(5)}
```

```
## [1] -0.04268491 -0.11267058  0.45682725  2.02033484 -1.05089006
```

1.14.4 Fonksiyon yazılması

R'nin sağladığı en önemli avantajlardan birisi kullanıcı tarafından fonksiyonlar yazılarak analizlerin kolaylaştırılmasıdır. Hazırlanan tüm paketler aslında fonksiyonlar bütünü olduğundan, fonksiyon yazılması anlaşıldığı takdirde ileri düzey programlamanın da temelleri de kolaylıkla öğrenilir.

R'de tüm fonksiyonların üç temel unsuru vardır: Birincisi, fonksiyonların içindeki kodların oluşturduğu `body()` , ikincisi fonksiyonların nasıl çalışacağını belirleyen parametrelerin olduğu `formals()` ve üçüncüsü fonksiyonun değişkenlerinin bulunduğu `environment()` unsurudur. Eğer aksi belirtilmediyse üçüncü unsur R'nin genel çevresidir (global environment). Ancak bunun bazı istisnaları vardır. Mesela, temel fonksiyonlar (Primitive Functions) olarak adlandırılan `sum()` gibi fonksiyonlarda formals(), body() ve environment() değerleri "NULL" değerini verir.

Bir fonksiyon öncelikle `function(X){}` komutu ile başlar. Burada X fonksiyonda kullanılacak parametreleri belirlerken, çengelli parantez içine uygulanacak fonksiyonlar veya kodlar yazılır. Fonksiyonun sonunda yer alan `return` komutu ise istenilmesi halinde fonksiyon ile oluşturulan nesnenin gösterilmesini sağlar.

```
function(X){
  ...
```

```
  return(Y)
}
```

Aşağıda yer alan örnekte, veri parametresi kullanılarak ortalama, standart sapma, ortanca ve varyans değerlerini göstermeye yarayan **ozet** isimli bir fonksiyon oluşturulmuştur.

```
ozet<-function(veri){
  a<-mean(veri)
  b<-median(veri)
  c<-var(veri)
  d<-sd(veri)
  e<-c(a,b,c,d)
  return(e)
}
x<-1:10
ozet(veri = x)
```

```
## [1] 5.500000 5.500000 9.166667 3.027650
```

Ozet fonksiyonunun unsurlarına bakılmak istenildiğinde yukarıda belirtilenler haricinde `edit()` fonksiyonu ile fonksiyonun detaylarına kolaylıkla bakılabilir.

```
body(ozet)
formals(ozet)
environment(ozet)
edit(ozet)
```

Fonksiyonlar hazırlanırken kolaylık sağlayan iki foksiyon vardır. Bunlardan ilki değişkenlerin görüntülenmesini sağlayan `print()` fonksiyonu, ikincisi elde edilen değerlerin açıklamalı sunumunu sağlayan `cat()` fonksiyonudur.

```
print(x)
cat(... , file = "", sep = " ", fill = FALSE, labels = NULL,
    append = FALSE)
```

örnek olarak 1-100 arası sayıların toplamı şu şekilde yazılabilir:

```
x <- 1:10
print(sum(x))
```

```
## [1] 55
```

```
cat("1'den 10'a kadar sayıların toplamı", sum(x), "eder")
```

```
## 1'den 10'a kadar sayıların toplamı 55 eder
```

Eğer kullanıcılar tarafından `infix` fonksiyonu tanımlanacaksa bunun eğik tek tırnak (“) içerisinde yazılması gereklidir.Aşağıda yer alan örnekte + işareti ile fonksiyon yazılmıştır.

```
`%+%` <-  function(a,b) {
  paste0(a,b)
}

`%+%`("yeni ", "ev")
```

```
## [1] "yeni ev"
```

```
`%+%`(5,8)
```

```
## [1] "58"
```

1.14.5 Hata ayıklama ve sorun giderme

Kısa fonksiyonlara kullanıcılar tarafındna yapılan hatalar kolaylıkla fark edilebilirken, programların uzaması hata yapma olasılığını da arttırır. Bu durumda R'de sunulan bazı fonksiyonlardan faydalanılarak hatalar tespit edilebilir ve sorunlar giderilebilir.

Öncelikle RStudio kullanılıyorsa fonksiyondaki hatalar IDE tarafından otomatik olarak gösterilecektir.Bununla birlikte uzun fonksiyonlarda `Shift + F9` tuşu ile denetleme noktaları (breakpoint) konularak programın bu denetleme noktalarına kadar tutarlı bir şekilde çalışıp çalışmadığı kontrol edilebilir.

Bu özellikler haricinde `traceback()` ve `browser()` fonksiyonları ile hatanın nerede olduğu denetlenebilir. Hata ayıklama konusunda daha detaylı bilgi için Wickham (2014) kaynağına başvurulabilir.

1.15 RMarkdown ve Belge hazırlama

R'de son dönemdeki en önemli yenilik RMarkdown paketi ile dinamik bir yazım imkanının sağlanmasıdır. Akademik çalışmalardan işletmeler için raporlar hazırlanmasına kadar pek çok uygulamaya imkan sağlayan RMarkdown araştırmacıların hayatını önemli ölçüde kolaylaştırmaktadır.

RMarkdown, R'nin hem analiz hem de yazı programı olarak kullanılmasına ve MS Word, PDF, HTML, LATEX olarak çıktı alınmasına imkan tanıyan bir programdır. **rmarkdown** paketi ile bu özellik yüklenebileceği gibi RStudio ile kullanıma hazır halde sunulmaktadır. RStudio'da bu özellik için `New File -> RMarkdown` kısmında sırasıyla *Document*, *Presentation* ve *Shiny* kısmından yeni dosya seçilebileceği gibi *Template* kısmından daha önceden hazırlanmış formatlar da kullanılabilir. Ülkemizde MS Word yaygın olarak kullanıldığından bu kısımda Word uygulaması kısaca anlatılacaktır. Diğer formatlar ve Rmarkdown hakkında daha fazla konu için rmarkdown.rstudio.com adresine bakılabilir.

RMarkdown belgesi, temel özelliklerinin belirlendiği `YAML Header` , R kodlarının yazıldığı bölümler `Chunks` ve yazının içinde dinamik olarak eklenen kısım olmak üzere üç kısımdan oluşur. Belge hazırlandıktan sonra `Knitr` veya `Ctrl + Shift + K` tuşlarına basarak belgenin çıktısının alınması sağlanır. RMarkdown belgesi `.Rmd` olarak kaydedilir ve **knitr** ve **pandoc** kullanılarak çıktı haline getirilir. Çıktı alma sürecinin teknik detaylarına ilişkin daha fazla bilgi için yihui.name/knitr/ ve pandoc.org bağlantılarına bakılabilir.

`New File -> RMarkdown -> Document -> Word` seçildikten sonra `Title` kısmına başlık ve `Author` kısmına yazar adı yazılır ve `OK` denildikten sonra RStudio `Source` kısmında RMarkdown belgesi açılır. Açılan belgede başlık (`YAML Header`) "—" ile başlayıp "—" ile biten kısımdır. Burada "title" belgenin başlığını, "author" yazar adını, "date" tarihi ve "output" ise çıktı türünü belirtmek için kullanılır. Bu kısımlarda düzeltme yapılabileceği gibi silinerek de çıkartılabilir.

```
---
title: "Untitled"
author: "SMS"
date: "4 Haziran 2017"
output: word_document
---
```

İkinci kısım ise yazının şeklinin belirlenmesidir.RMarkdown ile eğik (italic), kalın (bold), üst simge (superscript) ve alt simge (subscript) aşağıda gösterildiği şekilde yazılabilir. Başlıklar için başlıktan önce sırasıyla birinci düzey için "#", ikinci düzey için "##" ve üçüncü düzey için "###" yazılarak başlıklar oluşturulabilir.

```
*italic* or _italic_
**bold** __bold__
superscript^2^ and subscript~2~
```

Sıralama ve liste oluşturmak için "*" veya numara ile listeler oluşturulur. Taslak metinler için bu özellik önemli kolaylık sağlasa da çok sayıda liste var ise Word belgelerinin çıktısı alındıktan sonra nihai halinin Word ortamında düzeltilmesi gerekebilir.

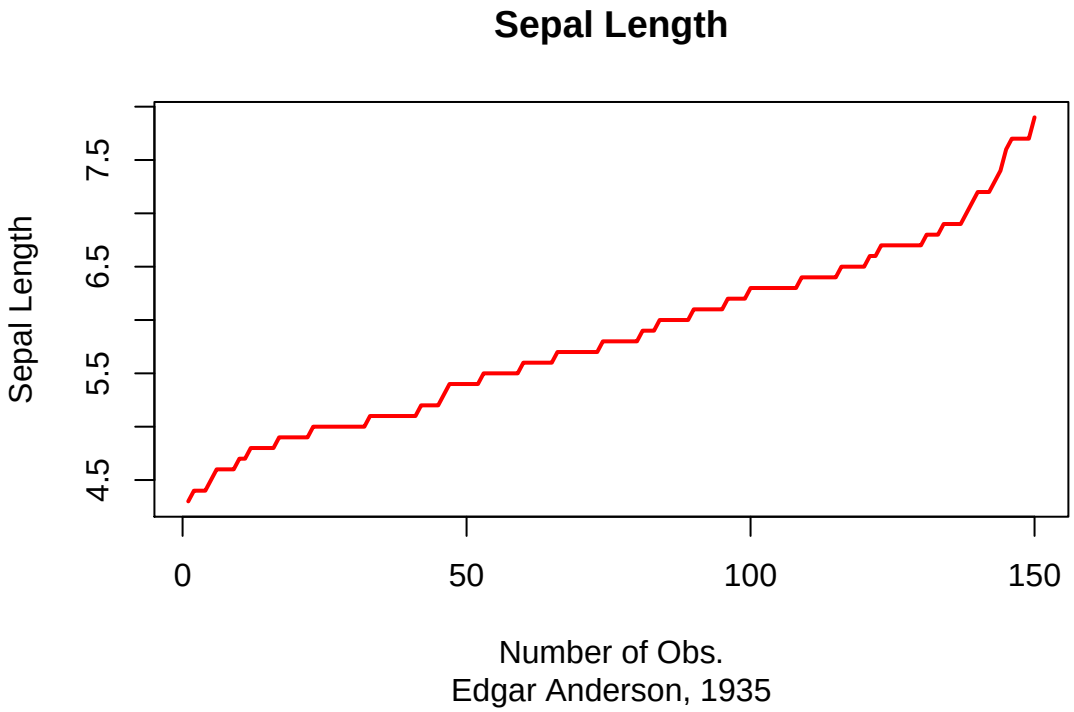
RMarkdown'ın en kullanışlı özelliği ise kodların yazıldığı `Chunk` yani parça kısmıdır. Parçalar `{r}` arasındaki kısımndan oluşur. Parça eklemek için ekrandan `Insert -> R` kullanılabileceği gibi `Ctrl + Alt + I` tuşları ile de ekleme yapılabilir. Bu parçalara yalnızca RScript değil aynı zamanda Python , Stan , SQL kodları da eklenebilir.

Aşağıda grafik kodlarını içeren bir parça gösterilmektedir. Bu örnek parça üzerinden bakıldığında `{r }` kısmının içerisinde şu ifadeler yer almaktadır:

```
baseplot, eval = FALSE, echo = TRUE, include = TRUE, message = FALSE,
warning = FALSE, error = FALSE, cache = FALSE, results = "asis",
fig.cap = "RMarkdown grafik örneği", fig.pos = "htb", fig.height = 4
```

```
# Temel grafik türlerinin gösterilmesi
require(ggplot2)
data(iris)
```

```
par(mfrow=c(1,1))
plot(sort(iris$Sepal.Length),type="l",col="red",lwd=2,
     main="Sepal Length", sub="Edgar Anderson, 1935",
     xlab="Number of Obs.", ylab="Sepal Length")
```



Şekil 1.2: RMarkdown grafik örneği

Bu ifadeler sırasıyla parçanın adı (baseplot) ve parçaya ait özellikler `eval` , `include` , `message` , `warning` , `error` , `results` olduğu görülmektedir. Her parça için ad belirtmeye gerek olmamakla birlikte grafik ve tablo hazırlamak için adın belirtilmesi atıf yapılması ve içerik oluşturulmasında kolaylık sağlar. Bu örnekte `eval` parçada yer alan kodun çalıştırılıp (TRUE) çalıştırılmayacağını (FALSE), `echo` R kodlarının gösterilip (TRUE) gösterilmeyeceğini (FALSE), `include` kodlar neticesinde elde edilen çıktının gösterilip (TRUE) gösterilmeyeceğini (FALSE), `message` ve `warning` uyarı mesajlarının gösterilip (TRUE) gösterilmeyeceğini (FALSE), `error` hata olması halinde parçanın derlenmesine devam edilip (TRUE) edilmeyeceğini (FALSE) belirtmek için kullanılır. `cache` ise derlemesi uzun süren kodlarda baştan derleme yapıldığında sıfırdan derleme yapılması yerine bir önceki sonucun bir dosyaya kaydedilerek (TRUE) zamandan tasarruf edilmesini sağlar. `results` kısmı ise elde edilen sonucun olduğu gibi "asis" veya gösterilmemesini "hide" sağlar. Ayrıca `fig.cap` , `fig.pos` ve `fig.height/fig.width` grafiğin sırasıyla adını, konumunu ve boy/en özelliklerini ayarlamak için kullanılır. Belirtilen bu özellikler neticesinde elde edilen grafik Şekil **??** ile gösterilmektedir.

Analiz sonucu elde edilen değerlerin gösterilmesi için `r` kullanılır. Burada bir analiz sonucu

elde edilen değerler gösterilebileceği gibi formüllerde yazılarak sonuçlar gösterilebilir. Bu kodun sunduğu en önemli avantaj analizlerde değerlerin değişmesi halinde yazıda sunulan değerinde güncellenmesi ve elle düzeltme ihtiyacının ortadan kalkmasıdır.

Referans ve kaynakça eklemek için ise *.bib* dosyaları kullanılarak aşağıda gösterildiği şekilde `bibliography` ekleyerek kaynakça eklenebilir. Kaynakçanın bulunduğu *.bib* dosyaları ise Mendeley veya Zotero gibi ücretsiz olarak sunulan referans yönetim programları ile hazırlanabilir. Kaynakça otomatik olarak dosyanın en sonuna RMarkdown tarafından yerleştirilecektir.

```
---
title: "Untitled"
author: "Murat"
date: "4 Haziran 2017"
output: word_document
bibliography: rkitap.bib
---
```

Yazının içinde gerekli atıflar ise `@` işaretinden sonra ilgili çalışmanın *.bib* dosyasındaki atıf adı ile yapılır. Referans göstermek için ise köşeli parantez ile kullanılması gereklidir.

Rmarkdown haricinde kullanılan başka bir uygulama ise **ReporteRs** paketidir. Bu paket ile Office dosyaları RMarkdown'a göre daha detaylı bir şekilde hazırlanabilir. Ayrıca **prettydoc** paketi de RMarkdown için farklı seçenekler sunmaktadır.

1.15.1 R Markdown ile LATEX

MS Word programı en çok kullanılan ofis programlarının başınd agelse de özellikle akademik çalışmalarda ihtiyaç duyulan fonksiyonları yeterince sağlayamamaktadır. Özellikle tez veya çok uzun raporlar hazırlanıyorsa belgenin tanzim edilmesi neredeyse belgenin yazımı kadar vakit almaktadır. Ayrıca bazı karakterlerin yazılmasıofis programlarında çok zor olmaktadır. Bu sorunla uğraşmamak için 1980'lerde geliştirilen LATEX (LaTeX) programı ile kullanıcılar tarafından belirlenen formatlarda belgelerin hazırlanması sağlanmakta ve kullanıcılara önemli vakit tasarrufu sağlamaktadır.

RMarkdown'da PDF dosyası seçildiğinde LATEX fonksiyonları kulllanılır hale gelmektedir. Ancak bunun için **Miktex** veya **Protex** programlarının yüklenmesi gereklidir. Bu programlar https://miktex.org/ veya https://www.tug.org/protext/ internet sayfalarından indirilebilir. Bu programlar indirildikten sonra RStudio otomatik olarak bu prgramları tanımlayacak ve kullanılır hale getirecektir.

LATEX başlı başına bir program olduğundan burada detayı bir şekilde anlatılmayacaktır. Ancak word dosyasında olduğu gibi YAML başlığında ilişkin özellikler belirtilebilir. Önek olarak, bu kitabın YAML başlığı şu şekilde düzenlenmiştir:

```
output:
  bookdown::pdf_book:
     highlight: tango
```

```
    number_sections: TRUE
    keep_tex: TRUE
    includes:
     header:
         - \usepackage{babel}
         - \usepackage{inputenc}
         - \usepackage{fontenc}
         - \usepackage{makeidx}
         - \usepackage{mathtools}
         - \usepackage{booktabs}
         - \usepackage{geometry}
         - \usepackage{graphicx}
         - \makeindex

bibliography: rkitap.bib
biblio-style: "apalike"
link-citations: TRUE
documentclass: book
fontsize: 12pt
lang: tr
toc: TRUE
lof: TRUE
lot: TRUE
toc_depth: 3
urlcolor: "blue"
linkcolor: "blue"
```

Burada sırasıyla çıktı olarak elde edilecek dökümanın tipi (pdf_book), bölümlerin numaralandırılıp numaralandırılmayacağı (number_sections) , elde edilen dökümanın .tex dosyasının saklanıp saklanmayacağı (keep_tex:TRUE), Miktex programının kullanacağı paketler (inludes: header) bibliyografya dosyası ve tipi (bibliography, biblio-style), yazı karakter büyükülüğü (fontsize), çalışmanın dili (lang), içindekiler (toc), şekil listesi (lof) ve tablo listesi (lot) olup olmayacağı tanımlanmıştır.

LATEX döküman ayarları yukarıda belirtildiği şekilde yapılabileceği gibi ayrıca oluşturulan şablonlar (templates) `template` başlığı ile de tanımlanabilir. Bu şablonlara pandoc-templates GitHub sayfasından erişilebilir.

```
documentclass: book
output:
  bookdown::pdf_book:
    template: xxxxx
```

RMarkdown .Rmd dosyalarının LATEX dosyalarına çevrilmesi için Pandoc programını kullandığından, Pandoc'ta belirtilen özellikler YAML başlığında tanımlanabilir. Pandocta tanımlanan LATEX özellikleri pandoc.org/MANUAL.html#variables-for-latex sayfasında

sıralanmakta olup, detaylı bilgi için ilgili sayfalara başvurulabilir.

LATEX'de kullanılan sembollere ve diğer fonksiyonlara ilişkin belgelere The LATEX Project ve Latex-Tutorial.com internet sayfalarından ulaşılabilir. Ayrıca Ek-1'de Winston Chang tarafından hazırlanan LATEX kılavuzuna bakılabilir. Söz konusu kılavuzun diğer versiyonları wch.github.io/latexsheet/ sayfasından indirilebilir.

1.16 Kod Yazılırken Dikkat Edilmesi Gereken Hususlar

Kod yazımında karşılaşılan en önemli sorunlardan birisi de herkesin farklı tarzı olması sebebiyle uzun kodların, özellikle de fonksiyonların, takibinin zorlaşmasıdır. Bu nedenle bazı usüllerin belirlenmesi kod yazımını ve paylaşımını kolaylaştıracak ve bazı süreçleri kısaltacaktır.

Bazı kurumların farklı tarzları olsa da genele hitap ettiğinden Google tarafından R kodları yazılırken uyulacak usulleri kısaca şu şekilde özetleyebiliriz. Bu usuller için Google's R Style Guide bağlantısına bakılabilir.

- Değişken isimleri küçük harfle yazılmalı, iki veya daha çok kelimeden oluşan değişken isimlerinde altçizgi (_) ve çizgi (-) kullanılmamalı, nokta (.) ile (ortalama.sure, average.time) ile yazılmalıdır.
- Fonksiyon isimleri ise büyük harfle başlamalı, birden fazla kelimeden oluşuyorsa aralarında boşluk veya başka bir karakter kullanılmadan kelimenin ilk harfi büyük harfle, (SimpsonKuralı gibi) yazılmalıdır.
- Bir satırdaki karakter sayısı 80 karakteri geçmemeli, bu sayıdan uzun kodlarda alt satıra geçilmelidir.
- İşleçlerin önünde ve arkasında mutlaka bir boşluk bırakılmalıdır. Fonksiyon içerisinde ise virgülden sonra bir boşluk bırakılmalıdır.
- Kıvrık parantezden sonra alt satıra geçilmeli, döngü komutlarında ise *else* iki kıvrık paratezin arasında olmalıdır.

```
if (condition) {
  --------
} else {
  --------
}
```

- Değer atamaları için <- işleci kullanılmalı, fonksiyonların içinde ise = işleci kullanılmalıdır.
  - Fonksiyonlarda parantezin hemen arkasından boşluk bırakılmamalıdır. Ayrıca fonksiyonlarda alt özellikler belirlenirken virgülden sonra alt satıra geçilmesi okumayı kolaylaştıracaktır.

Bölüm 2

R ile Grafik Hazırlama

Günümüzde ver miktarının hızla artması, verilere erişimin kolaylaşması ve bilgisayar programlarının gelişmesiyle verilerin sunuma hazır hale getirilmesi neredeyse ayrı bir uzmanlık alanı olarak ortaya çıkmıştır. Son dönemde geliştirilen paketler ile R gerek statik gerekse dinamik (web sayfaları veya başka uygulamalar için) en kullanışlı programlardan biri haline gelmiştir.

R'nin grafik hazırlama özelliklerinden bahsetmeden önce grafiklerin hazırlanmasına ilişkin bazı ilkelerden bahsedilmesi uygun olacaktır. Her ne kadar grafik hazırlamak kolay gözükse de her verinin görsel hale getirilmeye çalışılması karmaşık ve anlaşılmaz grafiklere yol açmaktadır. Bu nedenle, Edward Tufte'nin öncülüğünü yaptığı bu ilkelerin akılda tutulması güzel ve anlaşılabilir görseller hazırlanmasında faydalı olacaktır.

1. Başlangıç noktasının sıfır olması: Bazı durumlar haricinde baslangıç noktasının sıfırdan farklı seçilmesi grafiğin anlaşılırlığını zorlaştırmakta ve veriler arasındaki farkın olduğundan daha fazla gözükmesine neden olmaktadır.

2. En/Boy (E/B) oranının korunması: Bazen istenilen formatı sağlamak için grafiğin en/boy oranı ile oynamalar yapılabilir, ama bazı grafiklerde bu oranın değişmesi görselliği de bozacağından verilerin anlaşılmasında karmaşaya yol açabilir.

3. Grafiğin sade tutulması: Bazen grafiğin çekiciligini artırmak için başlık ve eksenlerde farklı stiller konulabilir. Ama grafik ne kadar sade olursa anlaşılması o kadar kolay olacaktır.

4. 3-Boyutlu grafikler: Çekici gözükseler de 3-boyutlu grafikler anlaşılmayı zorlaştırmaktadır. İki değişkenden fazla değişken iki boyutta gösterilmek istenildiğinde 3-boyutlu grafikler yerine renk, şekil ve boyut değişkenleri kullanılarak grafiğin hazırlanması verinin sunumunu kolaylaştıracaktır.

5. Birimler ve aralıklar: Grafikler hazırlanırken kullanılan birimler ve bu birimlere göre aralıkların seçilmesi grafiğin anlaşılmasını kolaylaştıracaktır. Aritmetik veya logaritmik birimlerin seçilmesi verilerin yorumunu da değiştireceğinden logaritmik birimlerin genelde büyüme ve zaman serisi gibi değişkenlerde kullanılması uygun olacaktır.

2.1 R'de Grafiklerin Hazırlanması

R'de grafikler temel olarak durağan ve dinamik grafik olmak üzere ikiye ayrılabilir. Durağan grafiklerde **base** , **ggplot** ve **lattice** olarak üç temel grafik hazırlama paketi bulunmaktadır. Dinamik grafiklerde ise Shiny , htmlwidget vb. çok sayıda paket hazırlanmış olup bu alan sürekli gelişmeye devam etmektedir.

Öğrenmenin kolay olması, grafik seçiminin basitliği ve esnekliği ve çok çeşitli girdilere imkan vermesi nedeniyle **ggplot** paketi en çok kullanılan paketleri arasndadır. Ayrıca son dönemde yapılan eklemelerle haritaların da eklenmesiyle ggplot son derece geniş bir kulanım alanı elde etmiştir.

Bu bölümde öncelikle temel grafikler anlatılacak daha sonra **ggplot** paketine geçilecektir. Lattice grafiği data teknik olduğundan ve daha az tercih edildiğinden, bu paket hakkında detaylı bilgi için Sarkar (2016) çalışmasına bakılabilir.

2.2 Temel (Base) Grafik Unsurları

Temel grafikler eldeki verilerin hemen incelenmesi için kullanılan basit özellikli grafiklerdir. Genelde verilerin yapısına bakmak için kullanılır ve yayın/sunum aşamasında grafikler **ggplot2** paketi ile hazırlanır. Tablo 2.1 temel grafiklerin türleri ve bunlara ilişkin fonksiyonları göstermektedir.

Tablo 2.1: Temel Grafik Türleri

Grafik Türü	R Kodu
Çizgi	plot()
Çubuk	barplot()
Histogram	hist()
Pasta	pie()
Nokta	dotchart()

Her bir fonksiyonda veri, grafik türü, eksen ve grafik başlıkları, renk ve boyutlar için öğeler mevcut olup, x ve y öğeleri grafikte kullanılacak verileri (bazı türlerde sadece x tanımlıdır), *type* grafik türünü, *main* grafik başlığını, *sub* grafik alt başlığını, *xlab* x-eksenini, *ylab* y-eksenini ve *asp* en/boy oranlarını belirlemek için kullanılır. Eğer doğrudan fonksiyon kullanılmayacaksa grafik türleri olarak "p" (points) noktalar, "l" (lines) çizgiler, "h" (histogram) çizgi-grafikleri, "s" (stairs) basamak grafiklerini belirtmek için kullanılırken, "n" ise grafik çizilmemesini sağlar. Bunun haricinde *col* renkleri, *lwd* boyutları, *pch* ise şekilleri seçmeye yarar.

```
plot(x, y, type="", main="", sub="",
     xlab="", ylab="", asp="", col="",
     lwd="", pch="")
```

Temel grafik fonksiyonları ile birlikte yaygın kullanılan `par()` fonksiyonu grafik parametrelerinin ayarlanmasını sağlar. Söz konusu fonksiyonla birlikte otuzdan fazla parametre belirlenebilmekte ve kullanılabilmektedir. En yaygın kullanılanlar grafik alanının bölünerek birden fazla grafiğin bir arada gösterilmesini sağlayan *mfrow* ve *mfcol*, yazıların büyütülmesini sağlayan *cex* , renklere ilişkin *col*, yazı tipine ilişkin *family* ve *font* parametreleridir.

Grafik örnekleri için **datasets** paketinde yer alan Edgar Anderson'un 1935 yılındaki "The irises of the Gaspe Peninsula" çalışmasında yer alan açelya (`iris`) veri seti kullanılacaktır.

```
# iris veri setinin yüklenmesi
attach(iris)
head(iris, n=5L)
```

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

Açelya veri setinde 3 tür açelyadan 50 çiçeğe ait yaprak ve çanak genişlikleri ve uzunluklarına ilişkin veriler bulunmaktadır. Bu veri setine ilişkin grafikler sırasıyla Şekil 2.1, 2.2 ve 2.3 ile sunulmaktadır. Kodlara dikkat edilirse `par(mfrow =c(1,2))` komutu kullanılarak grafiklerin hazırlandığı görülmektedir. Bu temel grafik çizim alanının 1 satır ve 2 sütuna bölünmesini sağlayan komut olup, bu değerler değiştirilerek birden fazla grafiğin gösterilmesi sağlanabilir.

İlk grafik örneğinde nokta ve histogram grafikleri gösterilmektedir.

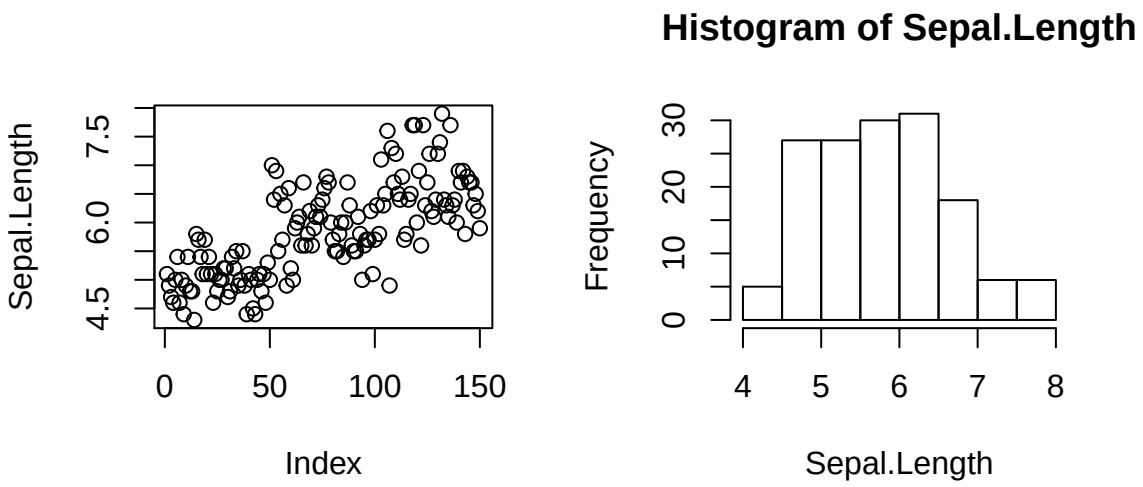
```
par(mfrow=c(1,2))
plot(Sepal.Length)
hist(Sepal.Length)
```

İkinci örnekte ise çanak uzunluğu ve genişliğine göre nokta grafiği oluşturularak ve çanak uzunlukları sıralanarak çizgi halinde gösterilmektedir.

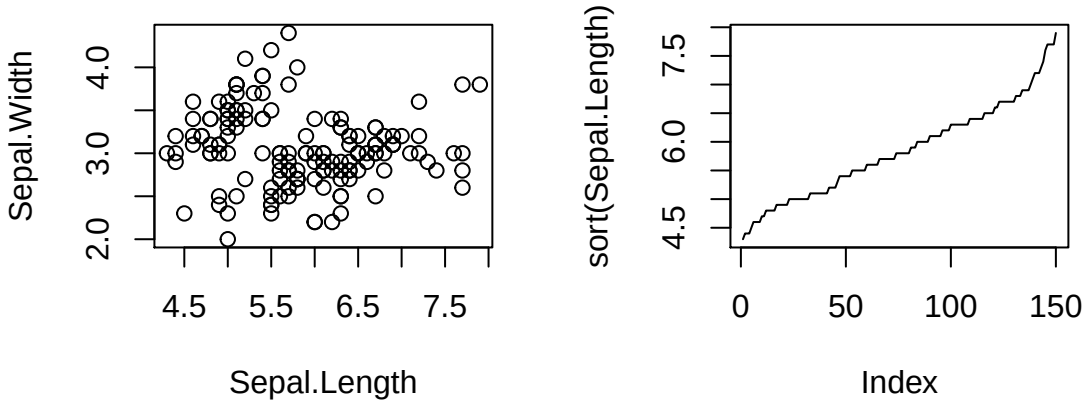
```
# Temel grafik türlerinin gösterilmesi-2
par(mfrow=c(1,2))
plot(Sepal.Length,Sepal.Width, type="p")
plot(sort(Sepal.Length),type="l")
```

Üçüncü örnekte ise çanak uzunluğu sıralı bir şekilde grafik eksen ve başlık değerleri olmadan ve bu değerler ile gösterilmektedir.

```
# Temel grafik türlerinin gösterilmesi-3
par(mfrow = c(1,2))
plot(sort(Sepal.Length), type = "l", col = "red", lwd = 2)
plot(sort(Sepal.Length),type = "l", col = "red", lwd = 2,
```



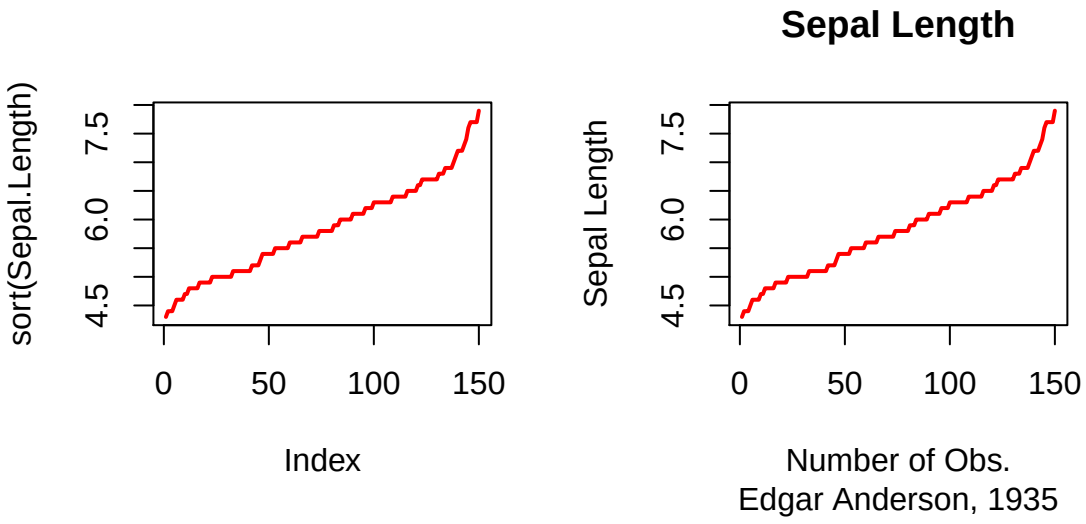
Şekil 2.1: Temel grafik örnekleri-1



Şekil 2.2: Temel grafik örnekleri-2

```
     main = "Sepal Length", sub = "Edgar Anderson, 1935",
     xlab = "Number of Obs.", ylab = "Sepal Length")
```

Temel grafik türleri hakkında daha detaylı bilgi için Frank McCown tarafından hazırlanan harding.edu/fmccown/r/ adresine bakılabilir. Ayrıca **plotrix** paketi ile sunulan fonksiyonlar ile temel grafik nesneleri önemli ölçüde geliştirilmiştir. Ancak **ggplot** paketi ve daha sonra geliştirilen diğer paketler nedeniyle bu fonksiyonlar son dönemdeki uygulamalarda pek tercih edilmemektedir.



Şekil 2.3: Temel grafik örnekleri-3

2.3 Fonksiyon Grafiklerinin Hazırlanması

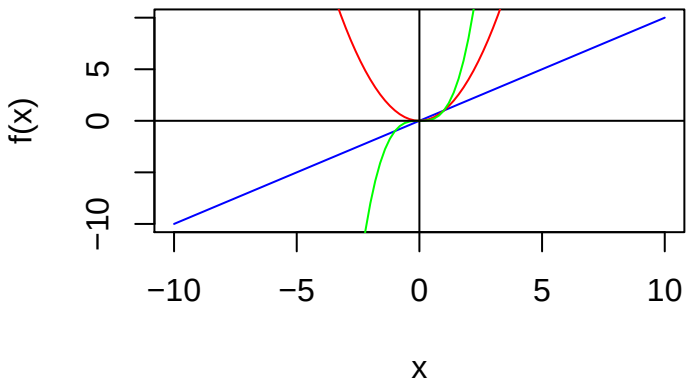
Fonksiyon grafiklerinin hazırlanması için `curve()` fonksiyonu kullanılır. Bu fonksiyonda *expr* ile kullanılacak fonksiyon, *from* ve *to* ile grafiğin başlangıç ve bitiş noktaları, *add* ile mevcut bir grafiğe ekleneceği (TRUE) veya eklenmeyeceği (FALSE), *type* ile grafik türü, *xname* ile X-eksenin adı ve diğer özellikler belirtilir.

```
# curve fonksiyonu
curve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", xname = "x", xlab = xname, ylab = NULL,
      log = NULL, xlim = NULL, ...)
```

Şekil 2.4 ile sırasıyla X, X^2 ve X^3 fonksiyonlarının `curve()` fonksiyonu ile elde edilmiş grafikleri gösterilmektedir. Burada dikkat edilecek önemli bir husus, ilk örnekte olduğu gibi fonksiyon olarak sadece "x" yazılacaksa bu parantez içinde yazılmalıdır, aksi takdirde değişken veya fonksiyon olduğu anlaşılmadığından komut hata verecektir.

```
# curve fonksiyonu ile kullanıcı tanımlı fonksiyon
curve((x),from = -10, to = 10, type = "l", col = "blue",ylab = "f(x)", add = FALSE)
curve(x^2,from = -10, to = 10, type = "l", col = "red", add = TRUE)
curve(x^3,from = -10, to = 10, type = "l", col = "green", add = TRUE)
abline(h = 0,v = 0)
```

Kullanıcı tarafından tanımlanan fonksiyonlara ek olarak kulllanılan fonksiyonların grafikleri de hazırlanabilir. Şekil 2.5 ile -5π ve 5π aralığında sinüs fonksiyonunun aldığı değerler gösterilmektedir.



Şekil 2.4: Curve fonksiyonu grafiği

```
curve(sin,-5*pi,5*pi, col = "red", ylab = "Sin(x)")
abline(h = 0,v = 0)
```

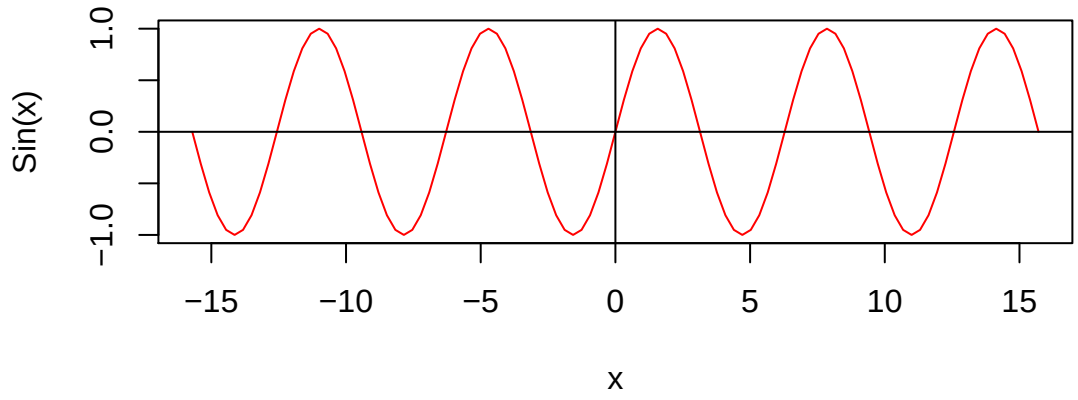
Fonksiyonların grafikle birlikte gösterilmesi için `text()` ve `mtext()` fonksiyonları kullanılır. İlk fonksiyonda matematiksel notasyon grafiğin içine eklenirken, ikincisinde eksenlerden birisine eklenir. Bu fonksiyonlarda yazılacak fonksiyon `expression()` ile tanıtılır. Burada LATEX komutları kullanılabilir, ancak `=` için `==` kullanılması gereklidir. `text()` fonksiyonunda x ve y öğeleri yazının koordinatlarını belirtmek için kullanılırken, `mtext()` fonksiyonunda *side* öğesi ile eksen (1=Aşağı, 2=Sol, 3=Yukarı, 4=Sağ) ve *at* ile eksen noktası, *las* ile yazının yönü (1:eksene paralel, 2: eksene dik) ve *line* ile (sıfırdan başlamak üzere) kaçıncı elemana ekleneceği belirtilir.

```
# text ve mtext fonksiyonları
text(x, y = NULL, labels = seq_along(x$x), adj = NULL,
     pos = NULL, offset = 0.5, vfont = NULL,
     cex = 1, col = NULL, font = NULL, ...)

mtext(text, side = 3, line = 0, outer = FALSE, at = NA,
      adj = NA, padj = NA, cex = NA, col = NA, font = NA, ...)
```

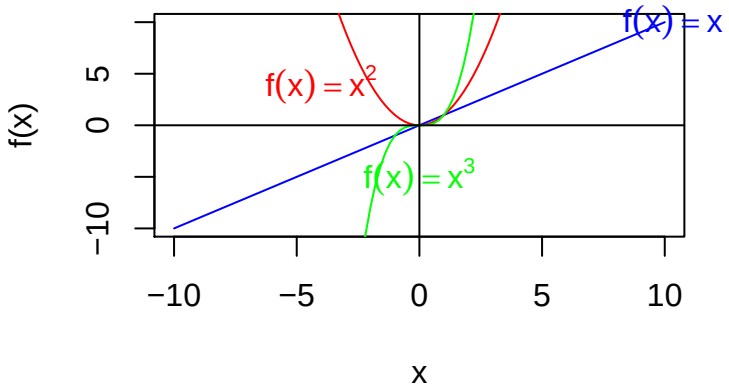
Şekil 2.6 ile sırasıyla X, X^2 ve X^3 fonksiyonları `text()` ve `mtext()` fonksiyonları ile birlikte kullanılmıştır.

```
# curve fonksiyonu ile kullanıcı tanımlı fonksiyon
curve((x),from = -10, to = 10, type = "l", col = "blue",
      ylab = "f(x)", add = FALSE)
mtext(text = expression(f(x) == x), side = 4, at = 10, line = 1,
```



Şekil 2.5: Sin fonksiyonu grafiği

```
      las = 2, adj = 1, col = "blue")
curve(x^2,from = -10, to = 10, type = "l", col = "red", add = TRUE)
text(-4, 4 ,expression(f(x) == x^2), col = "red")
curve(x^3,from = -10, to = 10, type = "l", col = "green", add = TRUE)
text(0, -5 ,expression(f(x) == x^3), col = "green")
abline(h = 0,v = 0)
```



Şekil 2.6: Fonksiyonların grafikle gösterimi

2.4 GGPLOT Paketi ile Grafiklerin Hazırlanması

Söz konusu örneklerde görüldüğü gibi temel grafik fonksiyonları veriler hakkında bilgi sağlamak için kullanışlı olsa da yayın kalitesinde grafikler hazırlamak için yeterli değildir. Bu nedenle Hadley Wickham tarafından hazırlanan **ggplot2** (Grammar of Graphics) paketi R kullanıcıları için bu sorunu çözmekle kalmamış, R'yi grafik hazırlamak için en popüler programlardan biri haline getirmiştir. Bu kısımda paketin temel özellikleri anlatılacak olup, daha kapsamlı bilgilere Wickham, Chang, ve RStudio (2016) paket referans dosyasından, "ggplot2: Elegant Graphics for Data Analysis" isimli kitaptan veya ggplot paketinin docs.ggplot2.org adresinde bulunan ulaşılabilir.

GGPLOT grafikleri birkaç unsurun bir araya getirilerek yapıldığı bir bloklar bütünü olarak düşünülebilir. Her bir blok grafiğin bir unsurunu tamamlamakta ve blokların tamamı derlendiğinde ortaya bir grafik çıkmaktadır. Her grafikte en az yedi unsur bulunmaktadır:

1. Veri (Data)
2. Grafik tipi (Geoms)
3. Kullanılacak verinin şekillendirilmesi (Stats)
4. Ölçeklendirme (Scales)
5. Bölümlendirme (Faceting)
6. Tema (Theme)
7. Koordinatlar (Coordinate)

Grafikler bir fonksiyon olarak düşünülürse grafiğin kodu blokların birbirine eklenmesi gibi yani `ggplot() + geom_X() + scale_XX() + ...` olarak yazılabilir.

Her grafikte oldugu ggplot grafiklerinde de ilk önce veri setinin tanımlanması gerekmektedir. Kullanılacak verilerin *data.frame* olarak değişken sınıflarının ve türlerin doğru kaydedilmesi gereklidir. Diğer bir çok programın aksine ggplot birden fazla *data.frame* nesnesinden seçilecek değişkenlere veri setleri fonksiyonda tanıtılması koşuluyla izin vermektedir.

Kullanılacak veri setleri `ggplot()` kısmında yazılabileceği gibi `geom_x()` kısmında da tanımlanabilir. Birden fazla veri seti (dikkat edin değişken değil) kullanılacaksa ilk veri setinin `ggplot()` kısmında tanımlanması işleri kolaylaştıracaktır.

Veri seti seçildikten sonra değişkenler ve grafiğe ilişkin temel görsel özelliklerin tanımlanması gerekmektedir. Bu *aesthetics* (estetik) veya kısaca `aes()` fonksiyonu ile sağlanır. Temel aes fonksiyonları:

- **x**: X eksenini belirler.
- **y**: Y eksenini belirler.
- **color**: Kullanılacak renkleri belirler, "red", "green" vb.
- **fill**: Kullanılacak renkleri belirler. Colordan farklı olarak grafikte alanlar kullanılıyorsa fill ile renkler seçilir.
- **alpha**: Renklerin saydamlığını ayarlar, özellikle çok sayıda değişken varsa yoğun bölgelerin koyu seyrek bölgelerin açık renk gözükmesi için kullanılır. 0 değeri en açık, 1

değeri en koyu değeri verir.
- **linetype**: Eğer çizgi kullanılacaksa tipini belirtkmek için kullanılır.
- **shape**: Scatterplotta kullanılacak şekilleri belirler.
- **size**: Grafik nesnelerinin büyüklüğünü belirler.

Bu fonksiyonlardan bazılarına sayısal değerler verilebileceği gibi kategorik değişkenlerde tanımlanarak değerler alması sağlanabilir. Mesela ülkelerin kıyaslanmasında nüfusa göre grafik nesnesinin büyüklüğü ayarlanmak istenirse *size* değerine nüfus büyüklüğünü gösteren değişken yazılır.

Belirtilmesi gereken önemli bir nokta, `aes()` fonksiyonu sadece `ggplot()` içerisinde değil diğer fonksiyonların içerisinde de kullanılabilir. Ancak `ggplot()` içerisinde tanımlanan aes unsurları tüm grafik için bağlayıcı olurken, diğer fonksiyonlarda tanımlananlar sadece o fonksiyon için bağlayıcı olurlar.

Aşağıda gösterilen örnek komutta `mtcars` verisetinde yer alan mpg (x ekseni) ve hp (y ekseni) değişkenleri grafik unsuru olarak atanmış ve renk olarak da sabit bir renk olan kırmızı veya veri setindeki başka bir değişken olan vites sayısına göre (gear) renklendirilmiştir.

```
# ggplot data, aes ve color örneği
require(ggplot2)
data(mtcars)
ggplot(data = mtcars, aes(x = mpg,y = hp,color = "red"))
ggplot(data = mtcars, aes(x = mpg,y = hp,color = as.factor(gear)))
```

Veri seti ve temel grafik özellikleri tanımlandıktan sonra kullanılacak grafik türünün seçilmesi gerekmektedir. Grafiğe eklenmek istenilen öğelerin türünü ve özelliklerini eklemek için `geom_x()` fonksiyonu kullanılmaktadır. `ggplot()`fonksiyonun sağladığı en önemli avantajlardan biri çok kolay bir şekilde birden fazla öğenin grafiğe eklenebilmesi ve özelliklerinin ayarlanabilmesidir. İkinci fonksiyonun ve diğer fonksiyonların eklenmesi için "+" işareti kullanılmaktadır. Yaygın olarak kullanılan `geom_x()` fonksiyonları şunlardır:

- *geom_bar()*: Çubuk grafiği
- *geom_boxplot()*: Kutu grafiği
- *geom_errorbar()*: T seklinde hata çubukları
- *geom_histogram()*: Histogram grafiği
- *geom_line()*: Çizgi grafikleri
- *geom_point()*: Scatterplot gibi nokta grafikleri
- *geom_text()*: Yazı grafikleri
- *geom_smooth()*: Lineer olmayan çizgiler ve lineer regresyon eğrisinin eklenmesini sağlar. Lineer regresyon eğrisi için `method = "lm"` komutu fonksiyona eklenir.

Bunlar haricinde **ggplot2** paketiyle kırktan fazla `geom_x()` fonksiyonu tanımlamak mümkündür. Ancak bazı fonksiyonlara sadece bir değişkenin tanımlaması gerekirken bazılarında iki değişkenin de tanımlanması gerekmektedir.

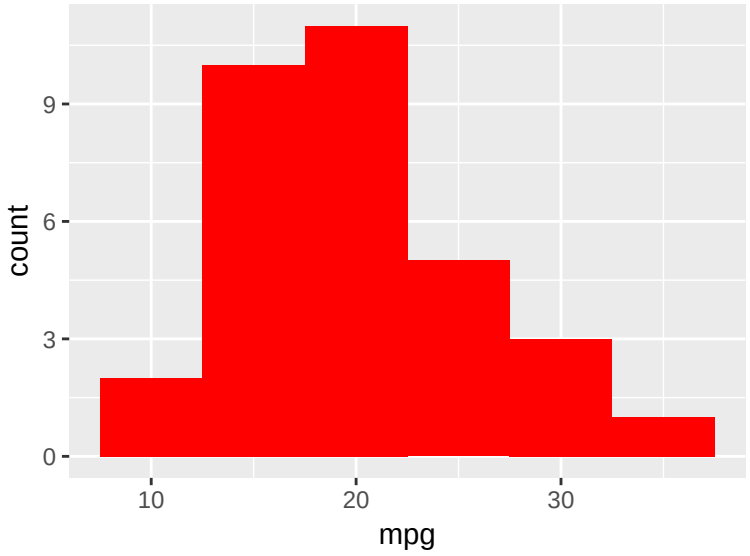
Tek değişkenli grafikler:

Tek değişkenli grafiklerde aşağıda gösterildigi üzere sadece "x" değişkeninin tanımlanması

yeterlidir. Bu grafikler özellikle dağılımın gösterilmesini sağlayan histogram, bar chart veya density gibi grafiklerdir. örnek olarak Şekil 2.7 de `mtcars` veri seti mpg değişkeni histogram olarak gösterilmektedir.

```
require(ggplot2)
require(ggthemes)
ggplot(mtcars, aes(x=mpg)) + geom_histogram()
ggplot(mtcars, aes(x=mpg)) + geom_density()
ggplot(mtcars, aes(x=mpg)) + geom_bar()
```

```
require(ggplot2)
require(ggthemes)
ggplot(mtcars, aes(x = mpg)) + geom_histogram(fill = "red",
                                              binwidth = 5)
```



Şekil 2.7: Tek değişkenli grafik örneği

İki ve daha çok değişkenli grafikler:

İki veya daha çok değişkenli grafikler için hem eksenler hem de üçüncü ve daha sonraki boyutları temsil eden unsurların tanımlanması gerekmektedir. "The Visual Display of Quantitative Information" isimli eseriyle tanınan ünlü istatistikçi Edward Tufte ve diger önde gelen grafik tasarımcıları anlaşılmasının zorluğu ve verilerin sunumu için yetersiz kalmaları nedeniyle 3 boyutlu grafikleri tavsiye etmemektedirler. Bu nedenle üçüncü ve sonraki boyutların iki boyuta renk, şekil veya ebat olarak konulması gerekmektedir.

Grafiğin daha iyi anlaşılması için bazen ortalama değerlerin gösterilmesi veya yatay, dikey veya verilerden gelen bir eğrinin eklenmesi gerekebilir. Bunun için `geom_hline()`,`geom_vline()` ve `geom_abline()` fonksiyonları sırasıyla yatay, dikey ve eğimli çizgilerin eklenmesi için kullanılabilir. Bu fonksiyonların özellikleri de yukarıda belirtilen geom fonksiyonları gibidir. Ancak x ve y eksenlerini kesen değerlerin tanımlanması ge-

rekir ki bunun için `geom_hline(yintercept="")`,`geom_vline(xintercept="")`, ve `geom_abline(intercept="", slope="")` kullanılır.

Aşağıda sunulan örneklerde sırasıyla `mtcars` veri setinde x ekseninde *mpg* ve y ekseninde *hp* değişkenleri olmak üzere dağılım grafiği (scatter plot) ve çizgi grafikleri ile renk, şekil ve boyut değişkenlerinin silindir sayısına göre hazırlanmış hali gösterilmektedir. Son iki komutta ise kutu grafiği ve doğrusal olmayan bir çizgi halinde grafikler sunulmaktadır.

```
# Çok değişkenli grafik örnekleri
require(ggplot2)
ggplot(mtcars, aes(x=mpg, y=hp)) + geom_point()
ggplot(mtcars, aes(x=mpg, y=hp)) + geom_line()
ggplot(mtcars, aes(x=mpg, y=hp, color= as.factor(cyl))) + geom_point()
ggplot(mtcars, aes(x=mpg, y=hp, shape= as.factor(cyl))) + geom_point()
ggplot(mtcars, aes(x=mpg, y=hp, size=  gear)) + geom_point()
ggplot(mtcars, aes(x=as.factor(gear), y=mpg)) + geom_boxplot()
ggplot(mtcars, aes(x=mpg, y=hp)) + geom_smooth()
```

Dikkat edilirse bazı grafiklerde bazı değişkenler `as.factor()` fonksiyonu içerisinde gösterilmiştir. Bunun nedeni, söz konusu değişkenleri "numeric" sınıfından çıkartarak kategori hale getirip ekseni kategori bazında düzenlemektir. Özellikle kesikli (discrete) değişkenler olması halinde bu değişkenlerin kategorik olarak ayrılması bazı grafikler için kolaylık sağlayacaktır. Şekil 2.8 bu örneklerden renk unsurunu göstermektedir.

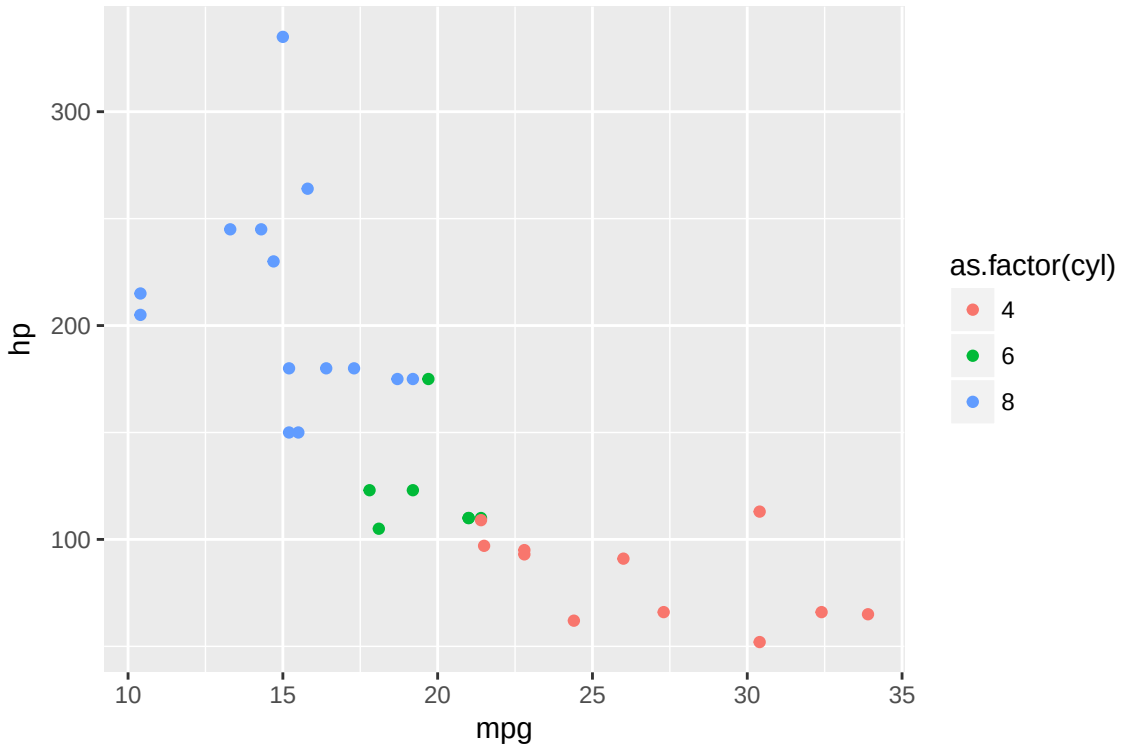
```
require(ggplot2)
ggplot(mtcars, aes(x=mpg, y=hp, color= as.factor(cyl))) + geom_point()
```

Grafik özellikleri belirlendikten sonra **Stat** özelliklerinin belirlenmesi gerekmektedir. Stat fonksiyonları verinin istatistiki dönüşümlerinde kolaylık sağlayan ve grafiklerin kolaylıkla hazırlanmasını sağlayan fonksiyonlardır.

Tablo 2.2: Ggplot Stat Fonksiyonları

Fonksiyon	Açıklaması
stat_ecdf()	Kümülatif dağılım fonksiyonunu çizer
stat_ellipse()	Yoğunlukların sınırını belirleyen ovaller çizer
stat_unique()	Mükerrer verileri temizler
stat_qq()	QQ grafiğini hazırlar
stat_function()	Bir fonksiyonun grafiğini ekler
stat_summary(),stat_summary_2d()	İki değişkenin özetini sağlar

Tablo 2.2 temel stat foksiyonlarını göstermektedir. Eğer bunlardan biri seçilmez ise ggplot otomatik olarak verileri oldugu gibi alan `stat_identity()` fonksiyonunu seçecektir. Eğer `stat_summary()` fonksiyonu kullanılacaksa ve başka bir fonksiyon belirtilmediyse ortalama ve bir standart hata değerini hesaplanmaktadır. Fonksiyon belirtmek için x veya y eksenindeki değişkenlere göre `fun.x="mean_cl_boot, mean_cl_normal, mean_dsl, median_hilow"` fonksiyonlarından biri seçilebilir.



Şekil 2.8: Çok değişkenli grafik örneği

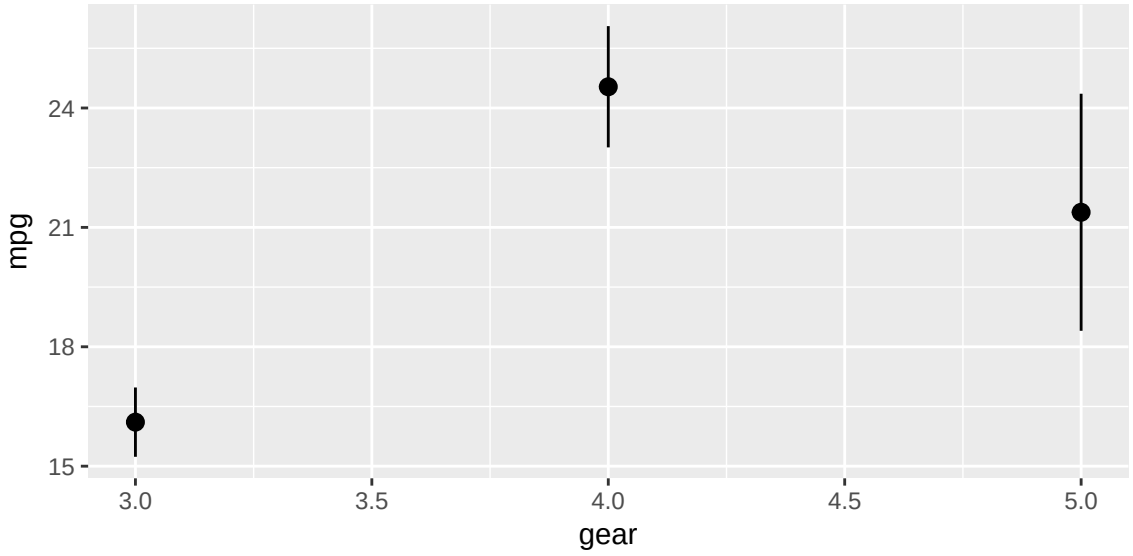
Aşağıda yer alan örneklerde sırasıyla `mtcars` veri setinde mpg değişkeninin vites sayısına göre kümülatif dağılım fonksiyonu örneği ve Şekil 2.9 ile vites sayısı ve mpg değişkenlerine göre özet verilerine (ortalama ve bir standart hata) ilişkin örnek gösterilmektedir.

```
# stat özelliklerine ilişkin örnek
require(ggplot2)
ggplot(mtcars, aes(mpg, colour = as.factor(gear))) +
  stat_ecdf(geom="line")
```

```
# stat özelliklerine ilişkin örnek
require(ggplot2)
ggplot(mtcars, aes(x=gear, y=mpg)) + stat_summary()
```

Grafiğin dördüncü unsuru **Scales** yani ölçeklerdir. Bu `aes()` fonksiyonu ile belirlenen değerlerin grafikte gösterilmesini sağlar. Temel olarak grafiğin görsel sunumu ile alakalı tüm unsurlar ölçekler tarafından ayarlanır. Ölçekler kendi içerisinde 3 gruba ayrılır: (1) Lejant (Legends) ayarları, (2) Eksen ayarları, (3) Grafik unsurlarına ilişkin ayarlar.

Lejant ayarları ,adından da anlasıldığı gibi grafikte bulunan öğelerin ne anlama geldiğini gösteren lejanta ilişkin ayarların yapılmasını sağlar. Temel olarak `aes()`fonksiyonunda eksenler dışında belirtilen özellikler (renk, şekil, boyut) otomatik olarak lejantı oluşturur. Ancak lejant başlığı, simgelerin sırası ve rengi gibi unsurların ayarlanması için bu ayarlar kullanılır. Lejant ayarları `guides()` , `guide_legend()` ve `guide_colourbar()` fonksiyonları ile yapılır.



Şekil 2.9: Stat özelliğine ilişkin örnek

`Guides()` fonksiyonu şekil, renk, başlık gibi temel özellikleri ayarlamaya yarayan genel bir fonksiyondur. Mesela renk ölçeği kullanıldığında bunun lejantta olması istenmiyorsa `guides(colour = "none")` komutu bir unsur olarak grafik koduna eklenir. Eğer renk, şekil gibi ölçeklerin lejantta gösterimi isteniyorsa `guide_legend()` fonksiyonu kullanılır. Bu fonksiyonda başlık (title), başlığın konumu (title.position-top,left,right,bottom), ölçek etiketlerinin gösterilip gösterilmeyeceği (label), etiketin konumu (label.position- top, left, right, bottom), lejant anahtarının genişliği, yüksekliği ve konumu (keywidth, keyheight,direction), lejantta olması istenilen satır ve sütun sayısı (nrow, ncol) ve diğer detay unsurlar ayarlanabilir.

```
guide_legend(title = waiver(), title.position = NULL, title.theme = NULL,
  title.hjust = NULL, title.vjust = NULL, label = TRUE,
  label.position = NULL, label.theme = NULL, label.hjust = NULL,
  label.vjust = NULL, keywidth = NULL, keyheight = NULL,
  direction = NULL, default.unit = "line", override.aes = list(),
  nrow = NULL, ncol = NULL, byrow = FALSE, reverse = FALSE, order = 0,
  ...)
```

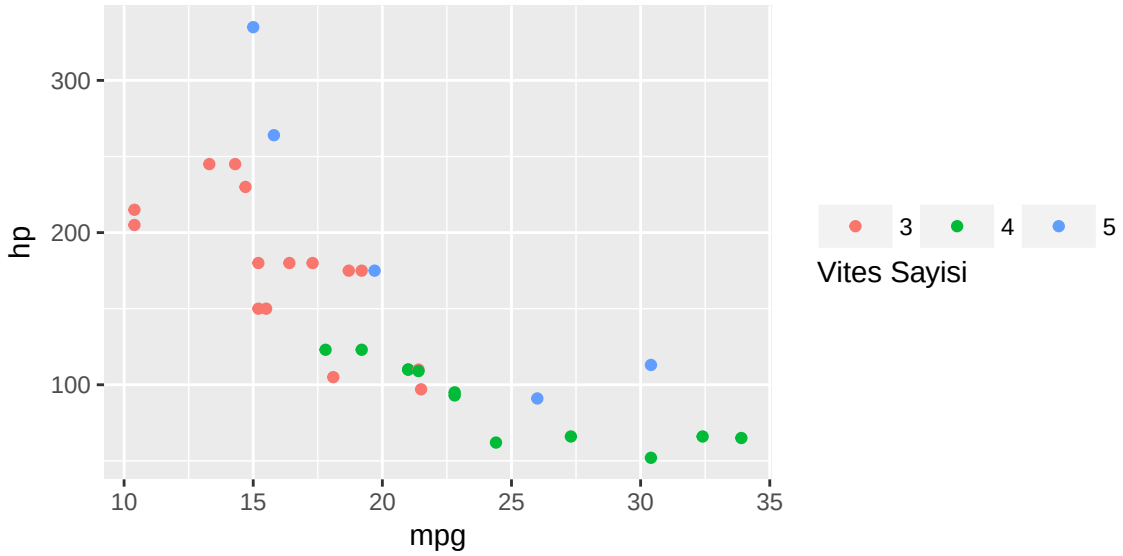
`guide_colourbar()` ise aynı özelliklerin sürekli değişkenlerin lejantta olması halinde kullanılmasını sağlar. Ölçek olarak sürekli değişkenler kullanıldığı zaman her bir değerin kategorik olarak kullanılması bir anlam ifade etmeyeceğinden belirli bir aralıkta sürekliliği gösteren ölçek kullanılması daha uygun olacaktır. `guide_legend()` fonksiyonundan farklı olarak genişlik ve yükseklik için *barwidth* ve *barheight*, renklendirme ölçeği için *nbin*, yatay veya dikey yönlendirme için *direction:horizontal,vertical* komutları kullanılır.

```
guide_colourbar(title = waiver(), title.position = NULL,
  title.theme = NULL, title.hjust = NULL, title.vjust = NULL,
  label = TRUE, label.position = NULL, label.theme = NULL,
  label.hjust = NULL, label.vjust = NULL, barwidth = NULL,
```

```
  barheight = NULL, nbin = 20, raster = TRUE, ticks = TRUE,
  draw.ulim = TRUE, draw.llim = TRUE, direction = NULL,
  default.unit = "line", reverse = FALSE, order = 0, ...)
```

Yukarıdaki örneklerde `mtcars` veri setinde mpg ve hp değerleri vites sayısı renk ölçeğinde olmak üzere `ggplot(mtcars, aes(x=mpg, y=hp,colour=as.factor(gear))) + geom_point()` kullanılarak grafik hazırlanmıştı. Şekil 2.10 lejantı, başlık "Vites Sayısı", başlığın konumu "Aşağı", satır sayısı "1" ve ölçek genişliği "2" olarak düzenlenmiş şekilde göstermektedir. İkinci örnekte ise gear değişkeni sürekli değişken olarak alınmış ve `guide_colourbar()` fonksiyonu ile grafik hazırlanmıştır.

```
# lejant örneği
require(ggplot2)
ggplot(mtcars, aes(x = mpg, y = hp, colour = as.factor(gear))) +
  geom_point() +
  guides(colour = guide_legend(title = "Vites Sayısı",
                          title.position = "bottom",
                          keywidth = 2, nrow = 1))
```



Şekil 2.10: Lejant Örneği

```
#Lejant ikinci örneği
ggplot(mtcars, aes(x = mpg, y = hp, colour = gear)) +
  geom_point() +
  guides(colour = guide_colourbar(title = "Vites Sayısı",
                                  title.position = "bottom",
                                  direction = "horizontal"))
```

Eksen ayarları ise grafik eksenlerinin ve başlığının düzenlenmesini sağlayan ayarlardır. Eksenlerin baslangıç ve bitiş değerlerini ayarlamak için `expand_limits()` ve `lims()` , eksenleri ve

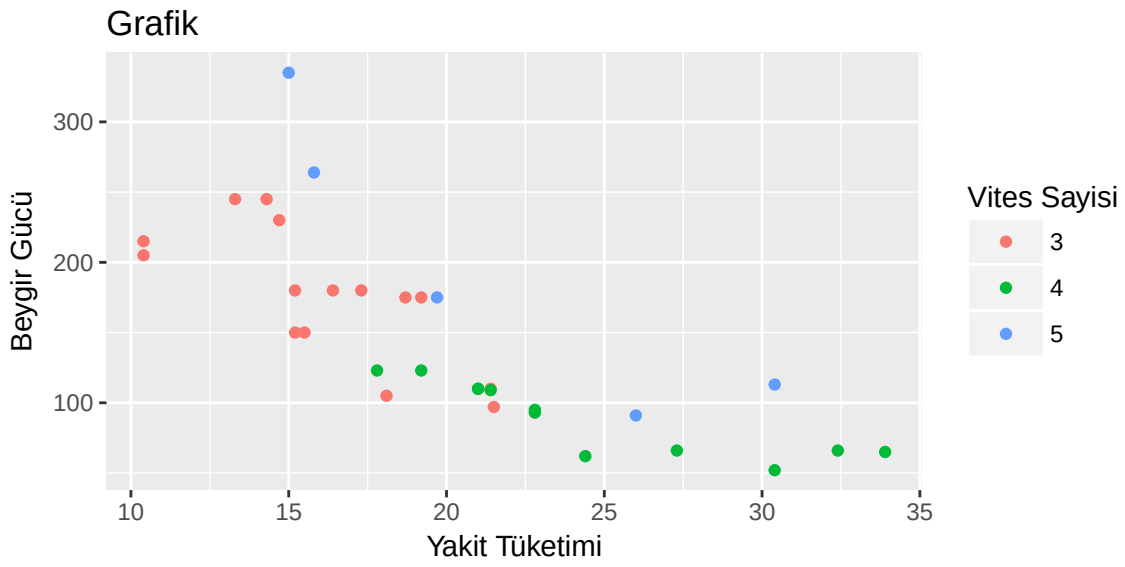
grafiği isimlendirmek için `labs()` fonksiyonları kullanılır. Mesela x ve y eksenlerinin alt ve üst sınırlarını belirlemek için kullanılacak fonksiyon şu şekilde yazılır:

```
#a,b,c,d değerleri alt ve üst sınırlar için girilecek değerledir
expand_limits(x = c(a,b), y = c(d,e))
```

Mevcut olarak ggplot değişkendeki en küçük değere göre eksen büyüklüklerini ayarlayarak verilerin daha detaylı gösterilmesini sağlar, ancak bazı durumlarda bu grafiğin anlaşılmasını zorlaştırabilir. Bu durumda grafiğin sınırlarının belirlenmesi daha iyi olacaktır. Eğer renk ölçeğinin sınırları belirlenmek istenirse `expand_limits(colour = seq(a, b, by = c))` fonksiyonu kullanılabilir. `lims(x = c(),y = c())` veya `xlim(a,b)`, `ylim(c,d)` fonksiyonları da kolaylıkla kullanılabilen ve aynı işlevi gören diğer fonksiyonlardır. `labs(title="", x="",y="")` fonksiyonu ise grafiğin ve eksenlerin isimlendirilmesi için kullanılır. Benzer bir şekilde `xlab()` ve `ylab()` fonksiyonları da kullanılabilir.

Şekil 2.11 bir önceki grafiğin eksen isimlerinin degiştirilmiş halini göstermektedir. Dikkat edilirse `labs` fonksiyonu `guides` içinde gösterilmemekte, ayrı bir blok olarak grafiğe eklenmektedir.

```
# Eksen isimleri örneği
require(ggplot2)
ggplot(mtcars, aes(x=mpg, y=hp,colour=as.factor(gear))) + geom_point() +
  guides(colour=guide_legend(title="Vites Sayısı",
                             title.position = "top",
                             keywidth = 2,nrow = 3),
         expand_limits(x=c(0,35),y=c(0,350))) +
  labs(title="Grafik", x="Yakıt Tüketimi", y="Beygir Gücü")
```



Şekil 2.11: Eksen örneği

Grafik unsurlarına ilişkin diğer ayarlar ise başta renk ayarları olmak üzere saydamlık ve

görsellikle alakalı ayarlardır. Bunlar sırasıyla:

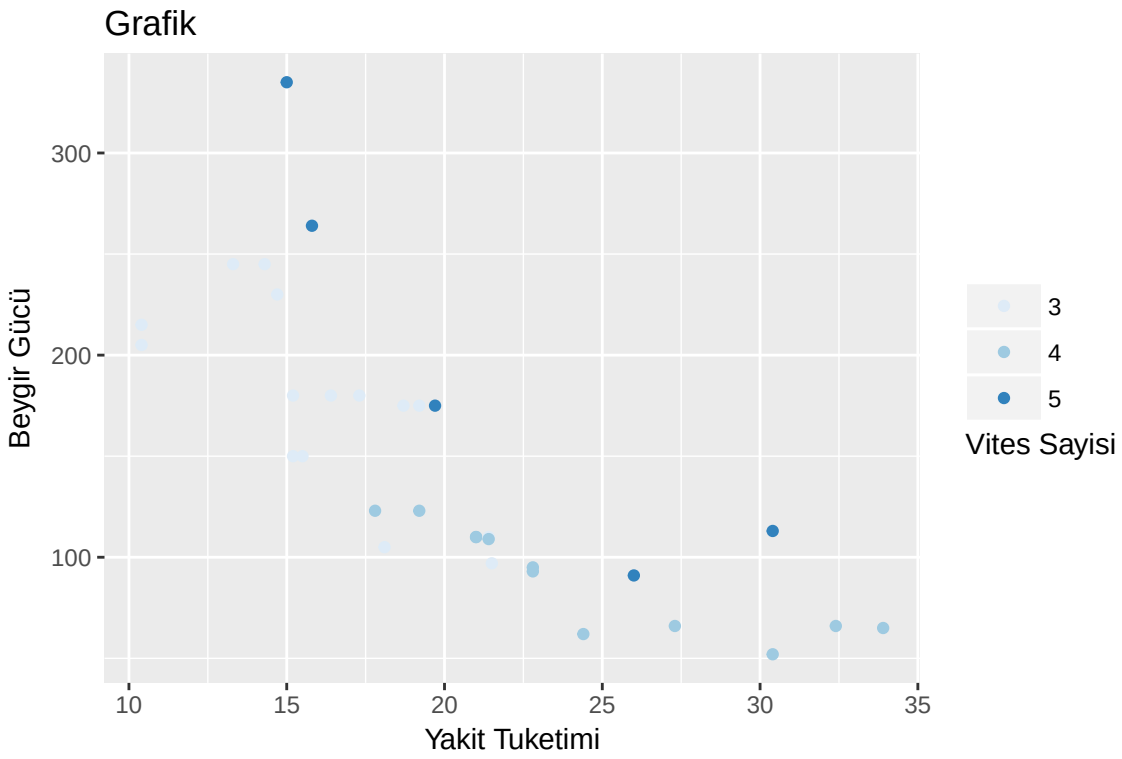
1. Saydamlık ayarları: `scale_alpha()`,
2. Renk ayarları: `scale_colour_brewer()`, `scale_colour_gradient()`, `scale_colour_grey()`, `scale_colour_hue()`,
3. Grafik unsurlarının ayarları: `scale_linetype()`, `scale_shape()`, `scale_size()`

fonksiyonları ile kullanılmaktadır. Her ne kadar karışık gibi dursa da aslında hepsinde hakim olan mantık aynıdır. Eger ölçek olarak bir değişken kullanacaksa bu değişkene göre belirlenecek özellikler bu fonksiyonlar ile yerine getirilir. Renk unsurları için genellikle `scale_colour_hue()` veya `scale_colour_brewer()` fonksiyonları kullanılmakta olup ilki farklı renk paleti sunarken, ikincisi aynı rengi farklı tonlarda kullanarak değişkenlerin gösterilmesini sağlar. Her iki fonksiyonda renklerin özelliklerini ayarlamak mümkün olup bunun için ggplot paketinin dökümentasyon sayfasından faydalanılabilir. `colour` fonksiyonuna benzer bir şekilde ölçek olarak `fill` kullanılmışsa `scale_fill_brewer()` fonksiyonu ile ayarları aynı şekilde yapmak mümkündür. Şekil 2.12 bir önceki örneği `scale_colour_brewer()` olarak göstermektedir.

```
# Renk ayarları örneği
require(ggplot2)
ggplot(mtcars, aes(x = mpg, y = hp,colour = as.factor(gear))) +
  geom_point() +
  guides(colour = guide_legend(title = "Vites Sayısı",
                               title.position = "bottom",
                               keywidth = 2, nrow = 3),
         expand_limits(x = c(0,35), y = c(0,350))) +
  labs(title = "Grafik", x = "Yakıt Tuketimi", y = "Beygir Gücü") +
  scale_colour_brewer()
```

Ölçek (scales) ayarlarından sonraki unsur bölümlendirme (faceting) unsurudur. Bazen belirli değişkenlere göre birden fazla grafik oluşturulması grafiğin anlaşılmasını önemli derecede kolaylaştırmaktadır. **ggplot** paketinde bölümlendirme için temel iki fonksiyon bulunmaktadır: `facet_grid()` ve `facet_wrap()`. Şekil 2.13 bir önceki örneğin karbüratör sayısına göre bölümlendirilmiş halini göstermektedir.

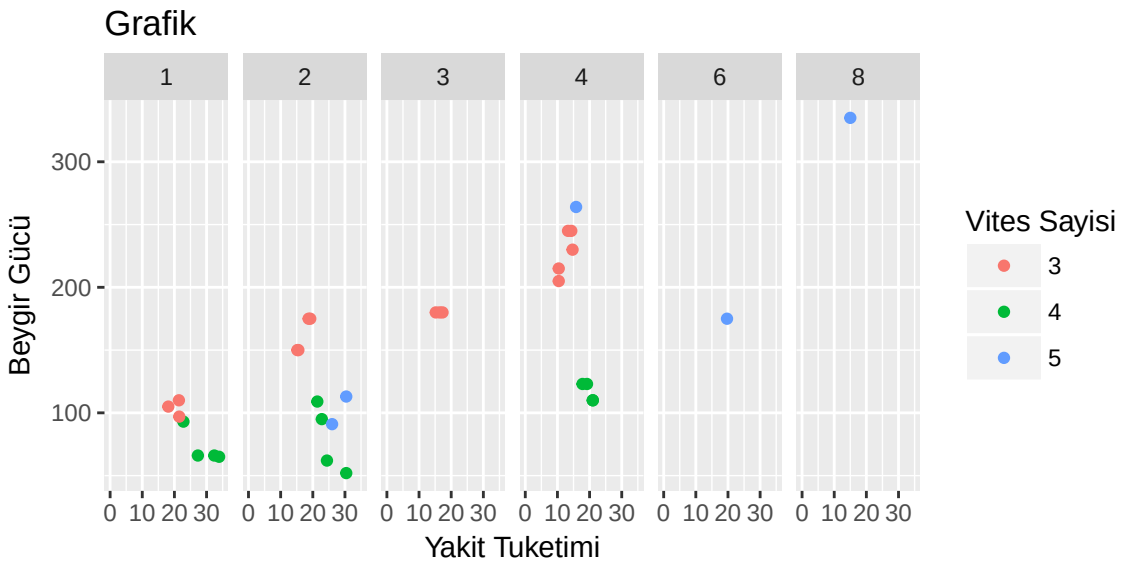
```
# Bölümlendirme örneği
require(ggplot2)
ggplot(mtcars, aes(x = mpg, y = hp, colour = as.factor(gear))) +
  geom_point() +
  guides(colour =guide_legend(title = "Vites Sayısı",
                               title.position = "top",
                               keywidth = 2,nrow = 3),
         expand_limits(y = c(0,350))) + xlim(0,35) +
  labs(title="Grafik", x = "Yakıt Tuketimi", y = "Beygir Gücü") +
  scale_colour_hue() +
  facet_grid(.~carb)
```



Şekil 2.12: Renk ayarları örneği

Grafiğin altıncı unsuru tema (Themes) öğeleridir. Bunlar grafiğin görünümünü ayarlayan eksen isimleri, eksen aralıkları, yazı türleri, grafik için renk seçimi vb. otomatik olarak ayarlayan unsurlardır. Gerek **ggplot2** paketi gerekse buna eklentiler yapan **GGaly** ve **ggthemes** paketleri söz konusu temaları ayarlamak için çok güzel seçenekler sunmaktadır (Arnold 2016). Burada mevcut tema `theme_grey()` olup, bunu değiştirmek için grafiğin sonuna `theme_bw()` vb. tema komutları yazılabilir. Farklı temaları kullanmak için söz konusu paketlerin yardım dosyalarından faydalanılabilir. Burada `theme_()` fonksiyonları içerisinde de görsellere ilişkin temel düzenlemeler yapılabilir.

```
# Tema örneği
require(ggplot2)
ggplot(mtcars, aes(x = mpg, y = hp, colour = as.factor(gear))) +
  geom_point() +
  guides(colour =guide_legend(title = "Vites Sayısı",
                              title.position = "top",
                              keywidth = 2,nrow = 3),
         expand_limits(x = c(0,35),y = c(0,350))) +
  labs(title="Grafik", x = "Yakıt Tuketimi", y = "Beygir Gücü") +
  scale_colour_hue() +
  facet_grid(.~carb) +
  theme_bw()
```



Şekil 2.13: Bölümlendirme örneği

Grafiğin son öğesi ise koordinat sistemidir (coordinates) . Mevcut olarak ggplot kartezyen koordinat sistemini kullanmakta olup farklı ihtiyaçlara göre bu da değiştirilebilmektedir. Ancak bu konu detay bir konu oldugundan paketin dökümantasyon sitesine veya kitaba başvurulabilir.

Hazırlanan grafiğin kaydedilmesi için RStudio Plots bölümünden faydalanılabileceği gibi `ggsave()` foksiyonu ile çalışma klasörüne *.tex (pictex), .pdf, .jpeg, .tiff, .png, .bmp, .svg* ve *.wmf* formatlarında kaydedilebilir. Söz konusu fonsiyonda *filename* "Grafik Adı.formatı", *path* klasörü, *plot* hangi grafik nesnesinin kaydedileceğini (mevcut olarak en son grafiktir) belirtir. Yükseklik (height) ve genişlik (width) için *units* ile ölçüleri inch, cm veya mm olarak belirtilir. "dpi" ise grafigin görüntü kalitesini belirler.

```
ggsave(filename, plot = last_plot(), device = NULL, path = NULL,
  scale = 1, width = NA, height = NA, units = c("in", "cm", "mm"),
  dpi = 300, limitsize = TRUE, ...)
```

Ggplot'a ilişkin eklenebilecek son bir husus, bu paket grafik kapasitesini sürekli geliştirmekte ve bu nedenle sürekli yeni uygulamaların ortaya çıkmasına imkan tanımaktadır. Özellikle yeni uygulamalar ile hareketli grafiklerin oluşturulmasına imkan tanımakta ve bu görsel metodların gücünü daha da arttırmaktadır. **Buna ek olarak,** Shiny gibi **web** uygulamaları giderek daha yaygın hale gelmekte ve bu uygulamaların temellerini bilenler için önemli fırsatlar sağlamaktadır. **Grafikler için bakılabilecek önemli bir site** R-graph-gallery.com adresidir. Bu adreste kullanılabilecek çok sayıda grafik türünün kodları bulunmaktadır. Edward Tufte'nin grafiklerinin uygulamaları için motioninso-cial.com/tufte/ sitesinden **de** faydalanılabilir.

2.4.1 Elektrik fiyatları için Heatmap örneği

Bu örnekte son dönemde kullanımı epey yaygınlaşan ve uzun bir serinin anlaşılır bir şekilde sunulmasını sağlayan "heatmap" grafiği oluşturulacaktır. Bunun için Enerji Piyasaları A.Ş. tarafından sunulan 2015-2016 yılları için sistem marjinal fiyatı kullanılacaktır[1]. Bu veriler saatlik olup, öncelikle saatlik verilerden günlük ortalama değerler elde edilecek ve daha sonra bu veri kullanılarak "heatmap" oluşturulacaktır. Veri EPİAŞ'ın sayfasından indirilebileceği gibi kitabın GitHub sayfasından da indirilebilir. Buna ilişkin kod aşağıda sunulmaktadır.

```
#Github sayfasından sistem marjinal fiyat verisinin yüklenmesi
library(readr)
smf <- read_delim("https://raw.githubusercontent.com/RiUA/riua/master/smf.csv",
    ";", escape_double = FALSE, col_types = cols(Saat = col_number(),
        Tarih = col_date(format = "%d/%m/%Y")),
    trim_ws = TRUE)
```

Veri saatlik olduğundan öncelikle her gün için ortalama fiyatların hesaplanması gerekmektedir. Bunun için **dplyr** paketi kullanılarak günlük ortalama sistem marjinal fiyatları (TL/MWh) hesaplanmış olup bu değerler Şekil 2.14 ile gösterilmektedir.

```
# Verilerin hazırlanması ve ilk grafik unsurları
require(dplyr)
require(ggplot2)
smf.m <-  smf %>%
  group_by(.,Tarih) %>%
  summarise(smfm = mean(SMF))
ggplot(smf.m) + geom_line(aes(x = Tarih, y = smfm),colour = "red") +
  xlab("Tarih") + ylab("TL/MWh") + theme_bw()
```

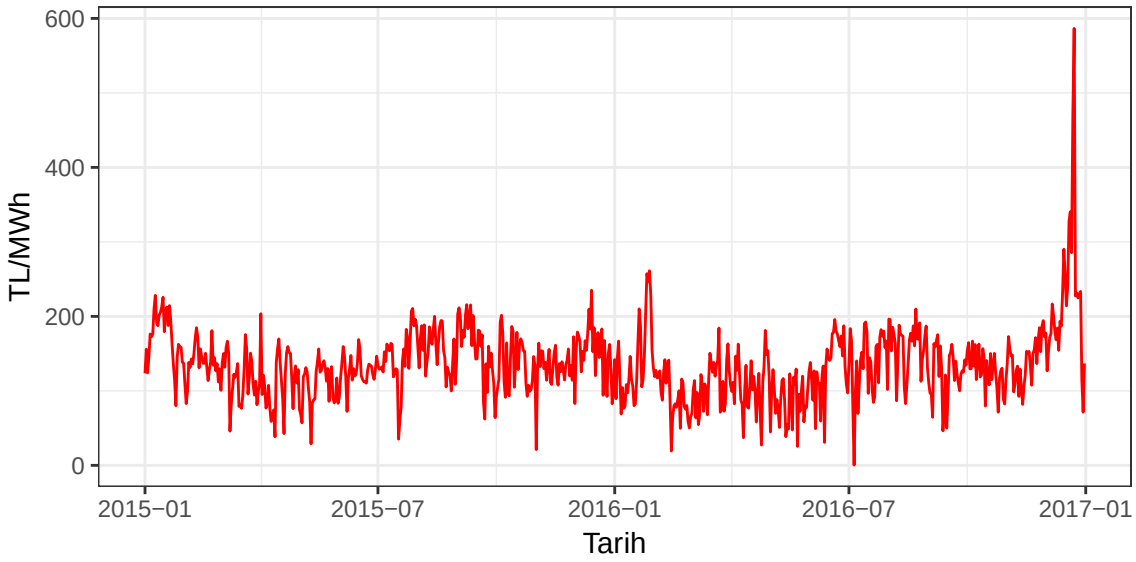
İkinci aşamada verinin günlük, haftalık, aylık ve yıllık olarak yeniden düzenlenmesi gerekmektedir. Bunun için kodlar şu şekildedir:

```
# Gün, hafta ve ay dğerlerinin değişken olarak düzenlenmesi
smf.m$Gun <-  factor(weekdays(smf.m$Tarih, T))
smf.m$gun <-  as.numeric(format(smf.m$Tarih, "%u"))
smf.m$Hafta <- as.numeric(format(smf.m$Tarih, "%W"))
smf.m$Ay  <- as.numeric(format(smf.m$Tarih, "%m"))
smf.m$Yil <- as.numeric(format(smf.m$Tarih, "%Y"))
```

Son aşamada **ggplot2**, **viridis** ve **ggthemes** paketleri kullanılarak oluşturulan grafik Şekil 2.15 ile gösterilmekte olup Y ekseni haftanın günlerini, X ekseni ise ayları ve haftaları göstermektedir. Şekilde de görüldüğü üzere 2016 yılının Aralık ayının son iki haftasında günlük ortalama fiyatlar önemli ölçüde yükselmiştir.

```
# Grafiğin oluşturulması
library(dplyr)
```

[1] EPİAŞ Şeffaflık Platformu-Piyasalar, https://seffaflik.epias.com.tr/transparency/



Şekil 2.14: Ortalama sistem marjinal fiyatı

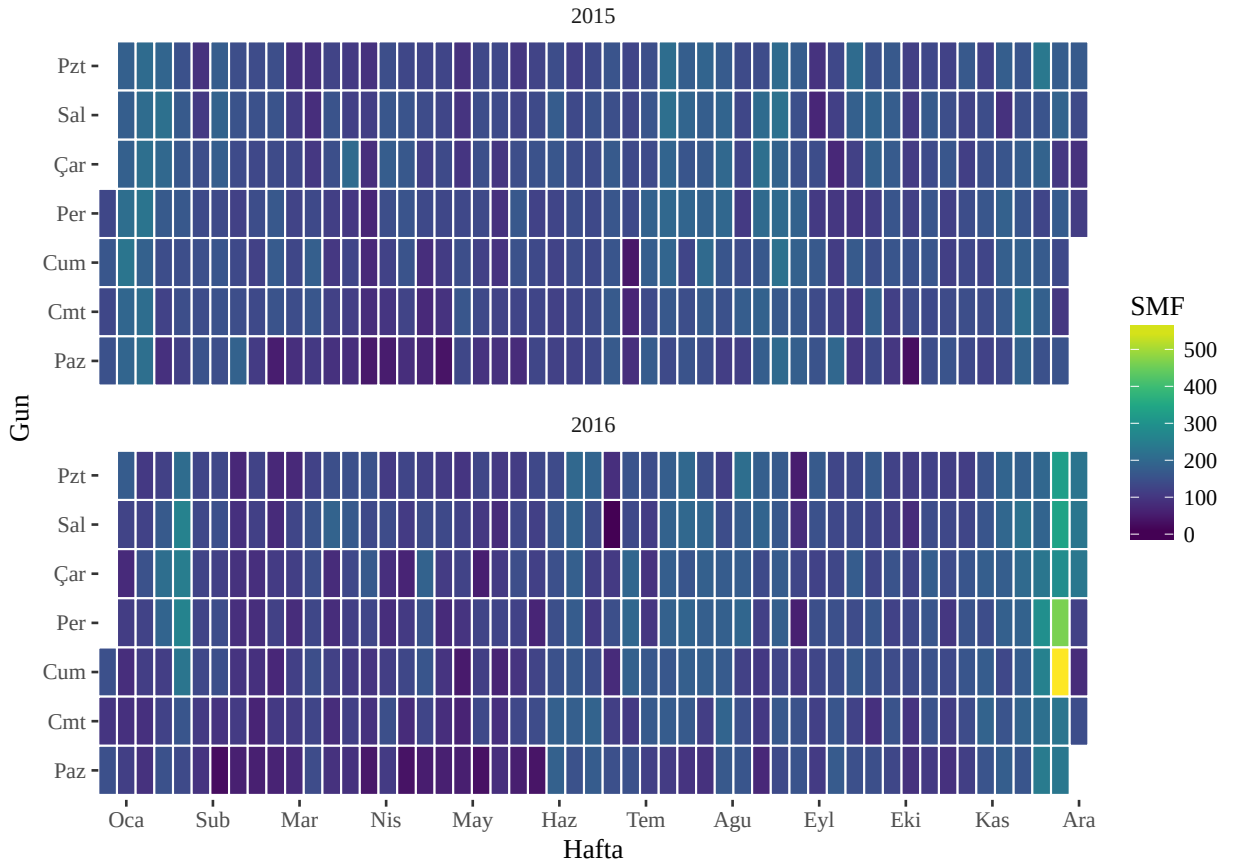
```
library(ggplot2)
library(viridis)
library(ggthemes)
ggplot(smf.m, aes(x = Hafta, y = reorder(Gun,-gun), fill = smfm)) +
  scale_fill_viridis(name="SMF",option = "D", limits = c(0, max(smf.m$smfm)))+
  geom_tile(color = "white", size = 0.4) + facet_wrap("Yil", ncol = 1) +
  scale_x_continuous(expand = c(0, 0), breaks = seq(1, 52, length = 12),
        labels = c("Oca", "Sub", "Mar", "Nis", "May", "Haz", "Tem", "Agu",
                   "Eyl", "Eki", "Kas", "Ara"))+
  ylab("Gun") + theme_tufte()
```

2.4.2 Economist dergisi örneği

Grafikler için güzel başka bir örnek The Economist dergisinin hazırladığı meşhur grafiklerden birini hazırlayarak verilecektir. Söz konusu örnekte ekonomik kalkınma ve yolsuzluk arasındaki ilişkiyi gösteren 2011 yılına ait bir makalede sunulan grafik güncel veriler kullanılarak hazırlanacaktır. Söz konusu makaleye The Economist-Corrosion and Development internet sitesinden ulaşılabilir[2]. Veri için 2015 yılı Transparency International "Corruption Perceptions Index-CPI" ve 2014 yılı Birleşmiş Milletler "Human Development Index-HDI" veri setleri kullanılmış olup söz konusu veri seti ayrıca Github sayfasına "hdi.csv" olarak yüklenmiştir. Veri setindeki her iki değişkende de daha yüksek değer daha iyi olduğu anlamına gelmektedir. Küsuratlı değerlerin karışıklık yaratmaması için HDI endeksi 100 ile çarpılmıştır.

Öncelikle verinin özelliklerine bakıldığında 164 ülkeye ait gözlem olup HDI için en düşük 35,

[2]The Economist, Corrosion and Development, 02/12/2011, http://www.economist.com/node/21541178



Şekil 2.15: Yıllar itibariyle günlük ortalama SMF

en yüksek 94 ve ortalama 69, CPI için en düşük 11 en yüksek 90 ve ortalama 43'tür.

```
#Veri setinin yüklenmesi ve özet değerlerine bakılması
library(readr)
hdi <- read_delim("https://raw.githubusercontent.com/RiUA/riua/master/hdi.csv",
    ";", escape_double = FALSE, col_types = cols(HDI = col_number()),
    trim_ws = TRUE)
summary(hdi)
```

```
##    Country               HDI             CPI
##  Length:163         Min.   :35.0   Min.   :11.00
##  Class :character   1st Qu.:55.0   1st Qu.:29.00
##  Mode  :character   Median :73.0   Median :38.00
##                     Mean   :69.6   Mean   :43.36
##                     3rd Qu.:83.0   3rd Qu.:56.50
##                     Max.   :94.0   Max.   :90.00
```

Grafiğin hazırlanması için ilk aşamada temel unsurlar ve şekiller tanımlanır. Burada grafiğe `geom_smooth()` fonksiyonu ile bir regresyon çizgisi eklenmiş ayrıca ortası boş olan nokta

grafiği için üç farklı boyutta ortası boş nokta, `geom_point()`, eklenmiştir.

```
# Temel grafik öğelerin tanımlanması
require(ggplot2)
eg1 <-  ggplot(hdi,aes(x = CPI, y = HDI))  +
  geom_smooth(method = "lm",
                formula = y ~ log(x),
                se = FALSE,
                color = "red") +
  geom_point(color = "lightblue4", size = 2.5, shape = 1) +
  geom_point(color = "lightblue3", size = 1.5, shape = 1) +
  geom_point(color = "lightblue4", size = 2, shape = 1)
```

İkinci aşamada gösterilmek istenilen bazı ülkeler seçilerek bunlar grafiğe eklenmiştir. Ancak normal `geom_text()` fonksiyonu kullanıldığında yazılar ve noktalar ile üst üste geleceğinden **ggrepel** paketinde bulunan `geom_text_repel()` yazıların üst üste gelmesi engellenmiştir.

```
# Grafiğe eklenecek ülke isimlerinin belirtilmesi
require(ggrepel)
pointsToLabel <- c("Russia", "Venezuela", "Iraq", "Myanmar", "Sudan",
                   "Afghanistan","Turkey", "Greece","Argentina","Brazil",
                   "India", "Italy", "China", "South Africa", "Spain",
                    "France", "USA", "Germany", "UK", "Norway", "Japan")

eg2 <- eg1 +  geom_text_repel(aes(label = Country),
              color = "gray20",
              data = subset(hdi, Country %in% pointsToLabel),
              force = 10)
```

Üçüncü aşamada ise eksenler ve başlıklar tanımlanmaktadır.

```
#eksenlerin ve başlıkların tanımlanması
eg3 <- eg2 +
  scale_x_continuous(name = "Corruption Perceptions Index (Higher Better)",
                                breaks = c(seq(0,100,10))) +
  scale_y_continuous(name = "Human Development Index (Higher Better)",
                       breaks = c(seq(0,100,20))) +
  expand_limits(y=0,x=0) +
  ggtitle("Corruption and Human Development",
          subtitle = "Sources: The Economist,
          Transparency International, UN")
```

Son olarak temada bazı ayarlamalar yapılarak grafiğin son hali Şekil 2.16 ile gösterilmektedir. Esas grafikten farklı olarak ülke grupları eklenmediğinden ülke gruplarına göre renklendirme yapılmamıştır.

```
# Temada bazı ayarların yapılması
require(ggthemes)
```

Şekil 2.16: Economist grafiği örneği

```
eg4 <-  eg3 + theme_tufte() +
    theme(text = element_text(color = "gray30"),
          axis.text = element_text(face = "italic"),
          axis.title.x = element_text(vjust = -1),
          axis.title.y = element_text(vjust = 2),
          axis.ticks.y = element_blank(),
          axis.line = element_line(color = "gray40", size = 0.5),
          axis.line.y = element_blank(),
          panel.grid.major = element_line(color = "gray50", size = 0.5),
          panel.grid.major.x = element_blank()
          )
```

2.4.3 Liman Yük Trafik Örneği

Ggplot için verilecek son örnek içinde güzel uygulamaları barındıran ve Sharp Sight Labs tarafından hazırlanan dünyanın en işlek limanlarına ilişkin grafik örneğidir. Bu çalışma için veriler Wikipedia'dan R yardımıyla alınmış olup, buna ilişkin kodlar sharpsightlabs.com/blog/ sitesinden indirilebilir. Ayrıca bu örneğe esas teşkil eden çalışma da yine aynı siteden bakılabilir.

Analize verilerin yüklenmesi ile başlanılacaktır. Bunun için **dplyr** paketi de kullanılarak veriler sharpsightlabs.com/wp-content/datasets adresinden indirilebilir. Ayrıca ham verilere "Wikipedia list of busiest container ports" araması ile de ulaşılabilir. Veriler 2004-2014 arası dünyanın en işlek konteynır limanlarının (İstanbul Ambarlı Limanı 39. sırada) bin TEU cinsinden yıllık konteynır trafiğini gösteren 9 değişken ve toplam 550 gözlemden oluşmaktadır. Ancak grafiklerde ilk 25 liman gösterilecektir.

```
# Liman verisinin yüklenmesi
require(dplyr)
url.world_ports <-
  url("http://sharpsightlabs.com/wp-content/datasets/world_ports.RData")
load(url.world_ports)
glimpse(df.world_ports)
```

```
## Observations: 550
## Variables: 9
## $ rank       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...
## $ year       <fct> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 201...
## $ continent  <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asi...
## $ economy    <fct> China, Singapore, China, Hong Kong, China, South Ko...
## $ port       <fct> Shanghai, Singapore, Shenzhen, Hong Kong, Ningbo-Zh...
## $ port_label <fct> Shanghai, Singapore, Shenzhen, Hong Kong, Ningbo/Z-...
## $ lon        <dbl> 121.473701, 103.819836, 114.057865, 114.109497, 121...
## $ lat        <dbl> 31.230416, 1.352083, 22.543096, 22.396428, 29.90162...
## $ volume     <dbl> 35268, 33869, 23798, 22374, 19450, 18423, 16624, 16...
```

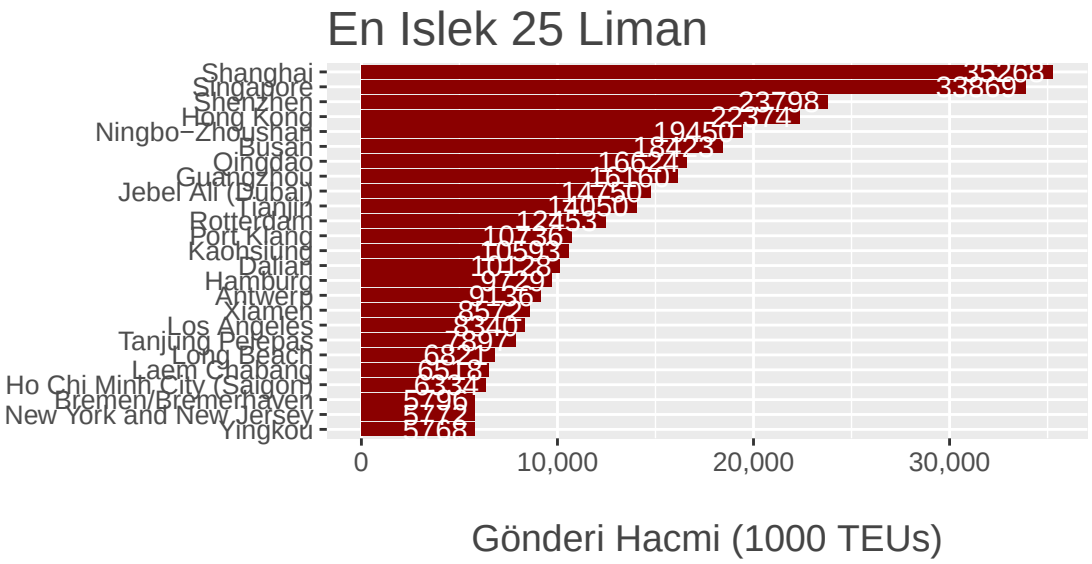
İkinci aşamada grafiklerin daha iyi gösterilmesi için kullanıcı tarafından bir tema (theme) hazırlanmış ve verilerin gösteriminde bu tema kullanılmıştır. Orjinal çalışmada dört tema oluşturulmakla birlikte burada sadece iki tanesi kullanılacaktır.

```
# Tema oluşturulması
require(ggplot2)
theme.porttheme <-
  theme(text = element_text(color = "#444444")) +
  theme(plot.title = element_text(size = 18)) +
  theme(plot.subtitle = element_text(size = 16)) +
  theme(axis.title = element_text(size = 14)) +
  theme(axis.title.y = element_text(angle = 0, vjust = .5,
                                    margin = margin(r = 15))) +
```

```
  theme(axis.text = element_text(size = 10)) +
  theme(axis.title.x = element_text(margin = margin(t = 20))) +
  theme(legend.title = element_blank())
```

Hazırlanan tema ile birlikte oluşturulan ilk grafik Şekil 2.17 ile gösterilmektedir. Söz konusu grafikte 2014 yılı itibariyle hacim bazında en işlek 25 liman gösterilmektedir.

```
# Hacim bazında en işlek 25 liman grafiği
require(ggplot2)
df.world_ports %>%
  filter(year == 2014, rank <= 25) %>%
  ggplot(aes(x = reorder(port, volume), y = volume)) +
    geom_bar(stat = "identity", fill = "dark red") +
    geom_text(aes(label = volume), hjust = 1.1, color = "#FFFFFF") +
    scale_y_continuous(labels = scales::comma_format()) +
    coord_flip() +
    labs(title = "En İşlek 25 Liman") +
    labs(x="", y = "Gönderi Hacmi (1000 TEUs)") +
    theme.porttheme
```



Şekil 2.17: 2014 yılı liman trafiği

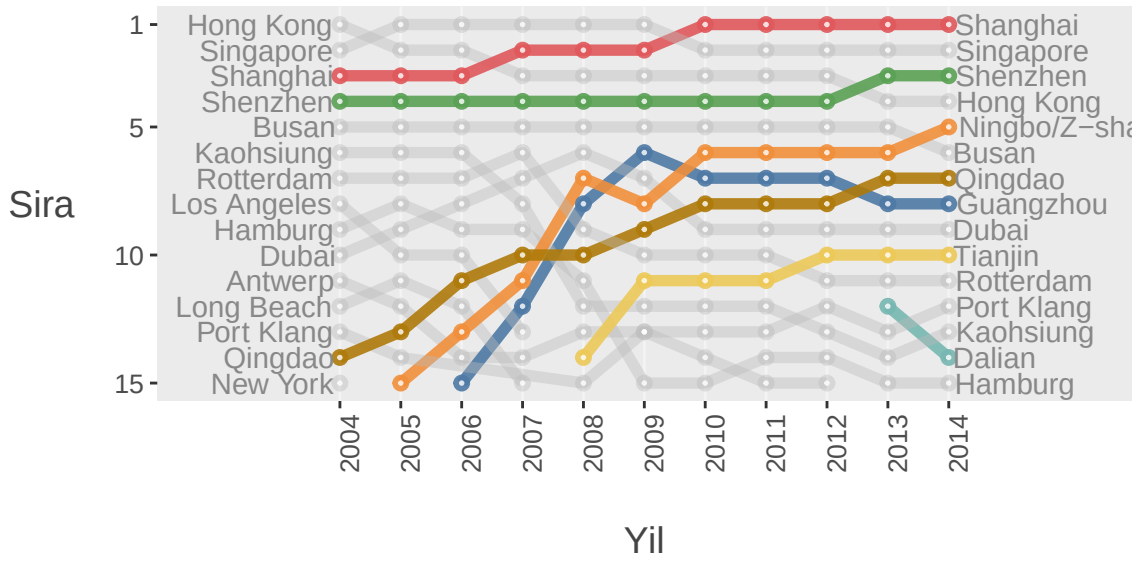
Şekil 2.18 ise Çin'de bulunan limanların (renkli) 2004-2014 yılları arasındaki sıralamalarındaki değişmeyi gösteren ve "Bump Chart" olarak adlandırılan grafiği göstermektedir.

```
# Bump-Chart kodları
require(ggplot2)
require(dplyr)
param.rank_n = 15
df.world_ports %>%
```

```
  filter(rank <= param.rank_n) %>%
  mutate(china_flag = ifelse(economy == "China", T,F)) %>%
  mutate(china_labels = ifelse(china_flag == T, port,"other")) %>%
  ggplot(aes(x = year, y = rank, group = port_label)) +
  geom_line(aes(color = china_labels, alpha = china_flag), size = 2) +
  geom_point(aes(color = china_labels, alpha = china_flag), size = 2.3) +
  geom_point(color = "#FFFFFF", alpha = .8, size = .3) +
  geom_text(data = df.world_ports %>%
            filter(year == "2014", rank <= param.rank_n),
            aes(label = port_label, x = '2014') , hjust = -.05,
            color = "#888888",
            size = 4) +
  geom_text(data = df.world_ports %>%
            filter(year == "2004", rank <= param.rank_n),
            aes(label = port_label, x = '2004') , hjust = 1.05,
            color = "#888888",
            size = 4) +
  scale_x_discrete(expand = c(.3, .3)) +
  scale_y_reverse(breaks = c(1,5,10,15)) +
  scale_alpha_discrete(range = c(.4,.9)) +
  labs(x = "Yıl", y = "Sıra") +
  theme.porttheme +
  theme(panel.grid.major.x = element_line(color = "#F3F3F3")) +
  theme(panel.grid.major.y = element_blank()) +
  theme(panel.grid.minor = element_blank()) +
  theme(legend.position = "none") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_color_manual(values = c("#4e79a5","#f18f3b","#af7a0a","#e0585b",
                                "#5aa155","#edc958","#77b7b2","#BBBBBB"))
```

Son olarak en işlek limanların haritada gösterilmesi için öncelikle bir harita teması hazırlanmış ve daha sonra 2014 yılı itibariyle limanlar gönderi hacmine göre Şekil 2.19 ile gösterilmiştir.

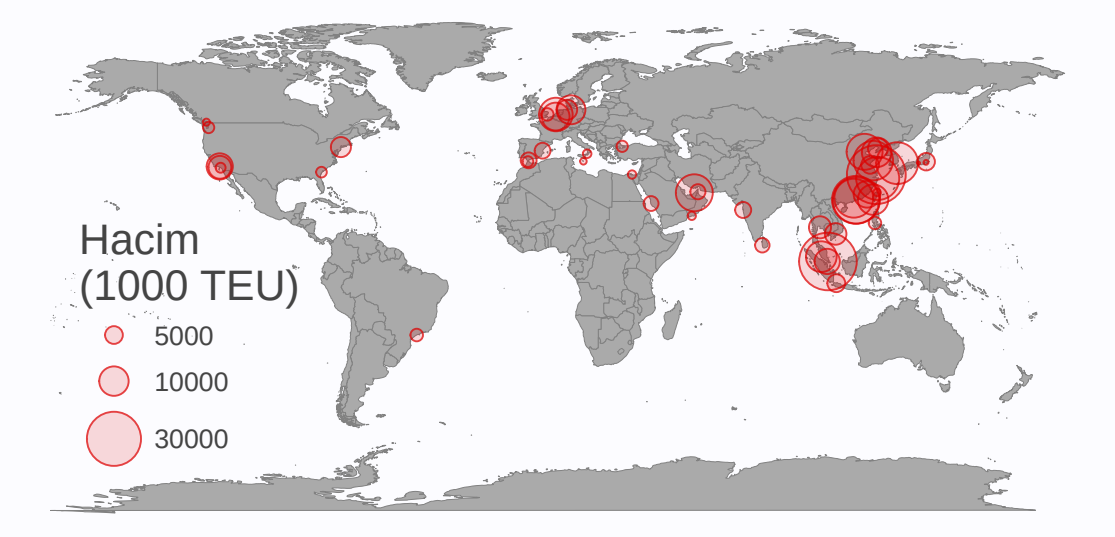
```
# Harita temasının hazırlanması
require(ggplot2)
theme.maptheeme <-
    theme(text = element_text(color = "#444444")) +
    theme(plot.title = element_text(size = 30)) +
    theme(plot.subtitle = element_text(size = 18)) +
    theme(panel.background = element_rect(fill = "#FCFCFF")) +
    theme(panel.grid = element_blank()) +
    theme(axis.text = element_blank()) +
    theme(axis.ticks = element_blank()) +
    theme(axis.title = element_blank()) +
    theme(legend.position = c(.17,.35)) +
```



Şekil 2.18: 2004-2014 liman sıralaması grafiği

```
    theme(legend.background = element_blank()) +
    theme(legend.key = element_blank()) +
    theme(legend.title = element_text(size = 16)) +
    theme(legend.text = element_text(size = 10))
```

```
# Harita ile limanların gösterilmesi için grafiğin hazırlanması
require(ggplot2)
require(dplyr)
require(tidyverse)
map.world_polygon <- map_data("world")
df.world_ports %>%
  filter(year == "2014") %>%
  ggplot(aes(x = lon, y = lat)) +
    geom_polygon(data = map.world_polygon, aes(x = long, y = lat,
                                               group = group),
                 fill = "#AAAAAA",colour = "#818181", size = .15) +
    geom_point(aes(size = volume), color = "#DD0000", alpha = .15) +
    geom_point(aes(size = volume), color = "#DD0000", alpha = .7,
               shape = 1) +
    scale_size_continuous(range = c(.2,10), breaks = c(5000, 10000, 30000),
                          name = "Hacim\n(1000 TEU)") +
    theme.maptheeme
```



Şekil 2.19: 2014 Yılı en işlek limanları

Kısım II

R ile Veri Analizine Giriş

Bölüm 3

İstatistikte Temel Kavramlar ve Yöntemler

Bu bölümde sosyal bilimlerde lisans ve lisansüstü programlarda İstatistiğe Giriş derslerinde anlatılan konuların R ile nasıl uygulandığı gösterilecektir. Burada konuların teorik temelleri yerine uygulaması anlatılmakta olup, daha detaylı bilgilere çeşitli istatistik kitaplarından erişilebilir.

Konunun iyi anlaşılabilmesi için bazı tanımların hatırlatılması yerinde olacaktır. *İstatistik birimi* sayılabilir veya ölçülebilir özellikleri içeren nesne, olgu veya olaylardır. *Anakütle* (istatiksel yığın - population) hakkında bilgi edinilmek istenen istatistik birimlerinin tamamını ifade eder. Belirli yöntemler kullanılarak seçilen ve anakütle ile aynı özellikleri taşıyan birimlerin oluşturduğu topluluğa ise *örneklem* (sample) denir. Anakütleyi ölçmek her zaman mümkün olmadığından analizlerin önemli bir kısmı örneklemlere dayanarak yapılmaktadır.

Örneklem istatistiği seçilen örneklemden elde edilen istatistiktir. Örneklem, büyüklüğüne ve seçilme yöntemine göre anakütleyi temsil edebileceği gibi anakütleden önemli ölçüde de sapabilir. *İstatistiki çıkarım* veya *çıkarımsal analiz* (statistical inference) örneklem istatistikleri kullanılarak bilinmeyen anakütle parametreleri hakkında olasılık kuramı kullanılarak sonuç çıkarmayı amaçlamaktadır.

İstatistik birimlerinin sahip oldukları özelliklere *değişken* denir. Değişkenlerin aldığı değerler ise *gözlem* veya *ölçüm değeri* olarak adlandırılır. Değişkenler aldıkları değerlerin sayısal veya sözel olmasına göre iki grupta değerlendirilebilir. *Sayısal değişkenler* (nümerik-metrik veri) sayma veya ölçme yoluyla elde edilen ve elde etme yöntemine göre *sürekli* (continuous) veya *ayrışık* (kesikli-discrete) olarak ikiye ayrılan değişkenlerdir. Sürekli değişkenler hava sıcaklığı gibi ölçülebilen değişkenler iken, ayrışık değişkenler çocuk sayısı gibi sayılabilen dğeişkenlerdir. Sayısal değişkenler aynı zamanda nicel değişken olarak da adlandırılmaktadır.

Nitel değişkenler ise renk, cinsiyet gibi sözel olarak tanımlanan değişkenlerdir. *Kategorik değişkenler* (metrik olmayan değişken) ise belirli kategorileri belirtmek için kullanılan ve verilerin sırası önemli olan (ordinal) veya önemsiz olan (nominal) değişkenlerdir. Anketlerde sorulan gelir veya yaş grubu soruları ordinal, şehir, cinsiyet vb. gruplar ise nominal kategorik

değişkene örnek teşkil etmektedir. Bunlar haricinde *oransal değişkenler* (ratio) sayısal verilerin birbirleri ile bölünmesi ile edilen veriler olup bu veri türü de sayısal veya kategorik olarak kullanılabilir. Sayısal değişkenler için parametrik testler uygulanabilirken, kategorik değişkenler için parametrik olmayan testler uygulanır.

Tahminci veya *tahmin edici* (estimator) örneklemden elde edilen ve anakütlenin bir parametresini (estimand) tahmin etmekte kullanılan bir istatistiktir. *Tahmin* (estimate) ise belirli bir örneklemde tahmin edicilerin aldığı değerlerdir ve bu nedenle örnekleme göre farklılık gösterir. Bir tahmin edicinin *örneklem dağılımı* ise rassal olarak seçilen bir örneklemde istatistiğin alabileceği değerlerin olasılık dağılımıdır (probability distribution). *Örnekleme hatası* (sampling error) anakütle parametrelerinin gerçek değeri θ ve tahmin edici $\hat{\theta}$ arasındaki farktır. Mesela, örneklem ortalaması ($\bar{X}$) anakütle ortalamasının (μ) yansız bir tahmin edicisidir ve aradaki fark örnekleme hatasıdır.

Örnekseme ilkesi (analogy principle) anakütledeki bir özelliğin örneklemdeki aynı özelliği kullanarak tahmin edilmesi gerektiğini söylemektedir. Yani anakütle ortalaması örneklem ortalaması, anakütle varyansı örneklem varyansı kullanılarak tahmin edilmelidir. *Yanlılık/sapma* (bias) tahmin edicinin beklenen değeri ile parametrenin gerçek arasındaki farktır. Eğer bir tahmin edici yansız/sapmasız ise beklenen değeri gerçek değere eşittir: $E(\hat{\theta}) = \theta$. *Etkinlik* (efficiency) ise tahmin edicinin örneklem dağılımının varyansını ifade eder. Bir tahmin edicinin başka bir tahmin ediciden daha etkin ise örneklem dağılımının varyansı daha düşüktür. *Tutarlı tahmin edici* (consistent estimator) örneklem büyüklüğü arttıkça tahmin edilen parametreye yakınlaşan tahmin edicidir. Diğer bir deyişle bir tahmin edici tahmin edilen bir parametreye *olasılıkta yakınsıyorsa* (convergence in probability) tutarlı bir tahmin edicidir. Bu, sonlu bir beklenen değere sahip birbirinden bağımsız ve eşit dağılmış rassal değişkenler örneklemi alındığında, örneklemin ortalaması anakütlenin beklenen değerine yakınsayacağını gösteren *Büyük Sayılar Yasası*'nın bir sonucudur. *Dağılımda yakınsama* (convergence in distribution) ise örneklemin dağılımının örneklemin büyüklüğü arttıkça anakütle dağılımına yakınsayacağını ifade eder. Bu özellikler en çok kullanılan teoremlerden Merkezi Limit Teoreminin temelini oluşturur.

3.1 Örneklem Yöntemleri

Örneklemler belirli varsayımlar altında *olasılıklı örneklem* (probability sampling) oluşturulabileceği gibi dağılıma ilişkin varsayımlar alınmadan *olasılıklı-olmayan örneklem* (non-probability-sampling) şeklinde oluşturulabilir.

Olasılıklı örneklem altı grupta incelenebilir. *Basit rassal örneklem* eldeki bir N elemanlı bir kümeden rassal olarak elemanlar seçilerek rassal bir örneklem oluşturulmasıyla elde edilir.. Böyle bir durumda kümedeki her elemanın seçilme olasılığı aynı olup $1/N$'dir. *Sistematik rassal örneklemde* ise N elemanlı bir kümeden belirli bir sistematik içerisinde elemanlar seçilmektedir. Mesela öğrenci listesiden beş ve beşin katlarında bulunan öğrencilerin seçilmesi gibi. *Katmanlı örneklemde* (stratified sampling) ise belirli kriterlere (yaş, cinsiyet, eğitim durumu vb.) göre

katmanlar (strata) oluşturularak bu katmanlardan rassal olarak elemanlar seçilir. Eğer ana kütle birbirine benzer alt kümelere bölünebiliyorsa bu yöntem tercih edilir.Eğer bu bölümlendirme coğrafik olarak yapılabiliyorsa bu durumda *küme örneklem* yöntemi uygulanır. Olasılıklı-olmayan örneklemler ise anakütlenin büyüklüğü, işlem maliyetleri veya belirli amaçlar göz önünde tutularak yapılan örneklemlerdir. Bunlara örnek olarak *kota* (quota) örneklemesi , *yargısal örnekleme* (judgemental sampling) , *kartopu örneklemesi* (snowball sampling) veya kullanımı pek tavsiye edilmeyen *kolay örneklem* (convenience sampling) sayılabilir.

Olasılıklı örneklem temel olarak `sample()` fonksiyonu ile yapılır. Bunun yanında **dplyr** ve **sampling** paketleri ile de uygulanabilir. `sample()` fonksiyonunda x bir veya birkaç elemandan oluşan vektörü, *replace* örneklemin yerine konularak seçilip seçilmeyeceğini, n seçilecek eleman sayısını *prob* olasılık ağırlıklarını belirtmek için kullanılır.

```
# Sample fonksiyonu
sample(x, size, replace = FALSE, prob = NULL)
```

Sampling paketi özellikle anket sonuçlarını değerlendirmek için çok kullanışlı fonksiyonlar içermektedir (Tillé 2016). Katmanlı veya küme örneklemi oluşturmak için `strata()` ve `cluster()` fonksiyonları kullanılılabilir. `strata()` fonksiyonunda *data* verileri içeren veri setini (data.frame veya matrix sınıfı olabilir), *stratanames* katman adlarını, *size* katmanın büyüklüğünü , *method* ise örneklemin hangi yöntemle seçileceğini ("srswor": yerine koymaksızın basit rassal, "srswr": yerine koyarak basit rassal, "systematic": sistematik ve "poisson": poisson örneklemi) ve eşit olmayan olasılıkla seçim için olasılıkları *pik* özelliği ile belirlenebilir. `cluster()` fonksiyonu da benzer şekilde tanımlanarak kullanılır.

```
#Strata ve cluster fonksiyonları
require(sampling)
strata(data, stratanames=NULL, size,
       method=c("srswor","srswr","poisson","systematic"),
       pik, description=FALSE)

cluster(data, clustername, size,
        method=c("srswor","srswr","poisson","systematic"),
        pik, description=FALSE)
```

3.2 Dağılım Ölçütleri

Dağılım ölçütleri elde edilen verilerin temel özelliklerinin birkaç değer ile gösterilmesini sağlayan ve veri seti hakkında genel bilgi edinilmesine imkan veren ölçütlerdir. Numerik ölçütler Verilerin yoğunlaştığı yerleri belirlemek için kullanılan *merkezi eğilim ölçütleri* , verideki değişikliği ve verinin yayılımını gösteren *yaygınlık/dağılım ölçütleri* ve *çarpıklık ve basıklık* ölçütleri olmak üzere üç grupta değerlendirilir.

3.2.1 Merkezi Eğilim Ölçütleri

Tablo 3.1: Merkezi Eğilim Ölçütleri

Ölçüt	Denklem	R Kodu
Aritmetik Ortalama	$\frac{\sum_{i=1}^{N} X_i}{N}$	mean()
Ortanca	-	median()
Tepe Değer (Mod)	-	-
Orta sınır değeri	$\frac{X_{min}+X_{max}}{2}$	range(), iqr(), fivenum()
Geometrik Ortalama	$\sqrt[n]{x_1 x_2 x_3 ...}$	geometric.mean()

Merkezi eğilim ölçütleri verinin dağılımının orta (tipik) noktalarını belirlemek için kullanılan ölçütledir. Bu ölçütler ve R fonksiyonları Tablo 3.1 ile gösterilmektedir.

Merkezi eğilim ölçülerinin fonksiyonları için `mtcars` veri seti kullanılarak örnekler verilecektir. İlk olarak **stargazer** paketi kullanılarak verinin temel özellikleri gösterilmiş, daha sonra **psych** paketi de kullanılarak ortalama, standart sapma, ortanca, geometrik ortalama ve aralık değerleri gösteren bir vektör oluşturulmuş ve son olarak orta-aralık (midrange) değeri hesaplanmıştır.

```
# Stargazer paketi ile verinin temel özelliklerinin gösterilmesi
require(stargazer)
data(mtcars)
stargazer(mtcars,type = "text",median = TRUE,iqr = TRUE)
```

```
##
## ===============================================================================
## Statistic N   Mean   St. Dev.  Min   Pctl(25) Median  Pctl(75)   Max
## -------------------------------------------------------------------------------
## mpg       32 20.091   6.027   10.400  15.425  19.200   22.800  33.900
## cyl       32  6.188   1.786     4       4       6        8       8
## disp      32 230.722 123.939  71.100 120.825  196.300 326.000  472.000
## hp        32 146.688  68.563    52     96.5     123     180      335
## drat      32  3.597   0.535   2.760   3.080    3.695   3.920    4.930
## wt        32  3.217   0.978   1.513   2.581    3.325   3.610    5.424
## qsec      32 17.849   1.787   14.500  16.892  17.710   18.900  22.900
## vs        32  0.438   0.504     0       0       0       1        1
## am        32  0.406   0.499     0       0       0       1        1
## gear      32  3.688   0.738     3       3       4       4        5
## carb      32  2.812   1.615     1       2       2       4        8
## -------------------------------------------------------------------------------
```

```
# Psych paketi ile merkezi eğilim ölçütlerine bakılması
require(psych)
with(mtcars, c(mean(mpg),sd(mpg),median(mpg),
               geometric.mean(mpg),range(mpg)))
```

```
## [1] 20.090625  6.026948 19.200000 19.250064 10.400000 33.900000
```

```
with(mtcars, c(IQR(mpg),quantile(mpg)))
```

```
##             0%    25%    50%    75%   100%
##  7.375 10.400 15.425 19.200 22.800 33.900
```

```
mid.range <- with(mtcars, 0.5*(min(mpg) + max(mpg)))
mid.range
```

```
## [1] 22.15
```

Bu paketlere ek olarak **DescTools** paketi ile sunulan `Mode()` fonksiyonu da mod değerinin hesaplanması için kullanılabilir.

3.2.2 Dağılım Ölçütleri

Tablo 3.2: Dağılım Ölçütleri

Ölçüt	Denklem	R Kodu
Değişim Aralığı	$X_{max} - X_{min}$	range()
Varyans (s^2)	$\frac{\sum_{i=1}^{N}(X_i-\bar{X})^2}{N-1}$	var()
Standart Sapma (s)	$\sqrt{\frac{\sum_{i=1}^{N}(X_i-\bar{X})^2}{N-1}}$	sd()
Değişim Katsayısı	$100\frac{s}{\bar{X}}$	Fonksiyonu bulunmamaktadır
Ortalama/Ortanca Mutlak Sapma	$\frac{\sum_{i=1}^{N}\lvert X_i-\bar{X}\rvert}{N}$	mad()

Dağılım ölçütleri verinin merkezine göre genel olarak nasıl dağıldığını ve merkeze göre uzaklıklarını göstermek için kullanılır. Bu ölçütler de Tablo **??** ile gösterilmektedir.

`mad()` fonksiyonu ile hem ortalama hem de ortanca mutlak sapma değerleri hesaplanabilir. Ortalama mutlak sapma için *center* öğesinde bunun *mean* veya *median* olarak belirtilmesi gerekir.

```
mad(x, center = median(x), constant = 1.4826, na.rm = FALSE,
    low = FALSE, high = FALSE)
```

Aşağıda yer alan örnekte söz konusu değerler `mtcars` veri seti ile hesaplanmaktadır.

```
# mad fonksiyonu ile dağılım ölçütlerinin hesaplanması
with(mtcars, c(var(mpg),sd(mpg),(100*sd(mpg)/mean(mpg))))
```

```
## [1] 36.324103  6.026948 29.998808
```

```
#center öğesi değeri mean olarak belirtilmiştir.
meanAD <- mad(mtcars$mpg, center = mean(mtcars$mpg))
medianAD <- mad(mtcars$mpg)
meanAD
```

```
## [1] 6.37518
```

```
medianAD
```

```
## [1] 5.41149
```

Değişkenler arasındaki ilişkiyi incelemek için ilk başvurulan istatistik korelasyon katsayısıdır . Başta Pearson yöntemi olmak üzere Kendall ve Spearman yöntemi ile de hesaplanan korelasyon katsayısı değişkenler arasında aynı veya ters yönlü ilişki olup olmadığını ve bu ilişkinin kuvvetini gösterir. R'de `cor()` fonksiyonu ile korelasyon katsayısı hesaplanmakta olup, bu fonksiyonda *x,y* öğeleri değişkenleri, *method* ise kullanılacak yöntemi belirtmek için kullanılır.

```
cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))
```

Korelasyon katsayısı ile birlikte bu değerin sıfırdan farklı olup olmadığını anlamak için `cor.test()` fonksiyonu ile hipotez testi uygulanır. Bu testte sıfır hipotezi değişkenler arasındaki korelasyonun sıfır olduğu hipotezi olup, p-değerinin belirli bir eşiğin altında olması halinde sıfır hipotezi reddedilir. Bu fonksiyonda `cor()` fonksiyonuna ek olarak *alternative* ile testin yönü ("two.sided": İki yönlü; "less": Küçük; "greater": Büyük), *conf.level* ile de güven aralığı belirtilir.

```
cor.test(x, y,
         alternative = c("two.sided", "less", "greater"),
         method = c("pearson", "kendall", "spearman"),
         exact = NULL, conf.level = 0.95, continuity = FALSE, ...)
```

Bu fonksiyona örnek olarak `mtcars` veri setinde yakıt tüketimi ve beygir gücü arasındaki korelasyona bakıldığında, korelasyonun -0,77 olduğu ve p-değerinin çok küçük olması sebebiyle iki değişken arasında istatistiki olarak anlamlı bir ilişki olduğu görülmektedir.

```
data(mtcars)
cor.test(mtcars$mpg, mtcars$hp)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mtcars$mpg and mtcars$hp
## t = -6.7424, df = 30, p-value = 1.788e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8852686 -0.5860994
## sample estimates:
##       cor
## -0.7761684
```

Eğer bir veya birden fazla değişkeni de dikkate alarak korelasyon hesaplanacaksa bu durumda *kısmi korelasyon* (partial correlation) değerine bakılır. Örnek olarak x,y ve z değişkenleri

varsa ve bu üçü birbiriyle sıfırdan farklı korelasyona sahipse, z'nin etkisi çıkartılarak x ve y arasındaki ilişki daha doğru değerlendirilir. Eğer z 'nin sadece y veya sadece x üzerindeki etkisi çıkartılırsa bu durumda *yarı-kısmi korelasyon* (part / semipartial correlation) hesaplanmış olur. Bu test sonucuna da iki değişken arasıdaki kısmi korelasyon (estimate) ve bunun sıfırdan farklı olup olmadığına ilişkin p-değeri (p.value) elde edilir.

Kısmi koreasyonu hesaplamak için **ppcor** paketi ile sunulan `pcor.test()` fonksiyonu kullanılabilir. Bu testte *x,y,z* öğeleri sırasıyla ilişkisi incelenecek ve veri alınacak değişkenleri, *method* ise kullanılacak yöntemi belirtmek için kullanılır. Bir önceki örnekten devamı olarak ağırlık veri alındığında yakıt tüketimi ve beygir gücü arasındaki ilişkinin istatistiki olarak anlamlı olmakla birlikte -0.77'den -0.54'e düştüğü görülmektedir.

```
require(ppcor)
attach(mtcars)
pcor.test(mpg,hp,wt)
```

```
##     estimate     p.value statistic  n gp  Method
## 1 -0.5469926 0.001451229 -3.518712 32  1 pearson
```

Varyansın yanında kovaryans ve korelasyon matrislerinin hesaplanması için `cov()`, `cor()` ve `cov2cor()` fonksiyonları kullanılır. Bu fonksiyonlarda x ve y kullanılacak verileri tanımlamak için kullanılır. `cov2cor()` foknsiyonu ise kovaryans matrisini korelasyon matrisine dönüştürmek için kullanılır. Bu temel fonksiyonlar haricinde **corpcor** paketi de korelasyon hesaplamaları için çeşitli yöntemler sunmaktadır.

```
cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))
cov2cor(V)
```

Eğer çok değişkenli bir veri seti ile çalışılacaksa *toplam varyans* (total variance) ve *genelleştirilmiş varyans* (generalized variance) kavramlarından da bahsetmek gereklidir. Toplam varyans, birden fazla değişken varsa bunların varyansları toplamına eşittir. Ancak bu değer veriler arasındaki korelasyonu dikkate almadığından *Genelleştirilmiş Varyans* (varyans-kovaryans matrisinin determinantı) kullanılır. Genelleştirilmiş varyans ne kadar yüksek olursa verilerin dağınıklı o kadar yüksek demektir. Eğer genelleştirilmiş varyans sıfır çıkıyorsa veride doğrusal bağımlılık (linear dependency) sorunu vardır ve çok değişkenli yöntemlerin kullanılmasında bu sorunun giderilmesi gerekmektedir (Johnson ve Wichern 1999).

Genelleştirilmiş varyans verilerin büyüklüklerine bağlı olduğundan, verilerin özelliklerinden etkilenmemesi için *standartlaştırılmış değişkenlerin genelleştirilmiş varyansı* (generalized sample variance of the standardized variables) kullanılır. Bu değerin hesaplanması için varyans-kovaryans matrisi yerine korelasyon matrisinin determinantı alınır ve bu değerin yüksek olması verinin dağınık olduğu ve değişkenler arasıdaki ilişkinin de zayıf olduğu anlamına gelir.Bu varyansların hesaplanması için `det(cov(X))` ve `det(cor(X))` fonksiyonları kullanılabilir.

Tablo 3.3: R Dağılım Fonksiyonları

Dağılım Türü	Olasılık	Kümülatif	Kantil	Rassal
Tekdüze	dunif	punif	qunif	runif
Normal	dnorm	pnorm	qnorm	rnorm
Binom	dbinom	pbinom	qbinom	rbinom
Hipergeometrik	dhyper	phyper	qhyper	rhyper
Poisson	dpois	ppois	qpois	rpois
Geometrik	dgeom	pgeom	qgeom	rgeom

3.2.3 Basıklık ve Çarpıklık

Çarpıklık dağılımın simetrik olup olmadığını gösteren ölçüt iken, basıklık verinin ortalama etrafında ne kadar toplandığını ve kuyrukların uzunluğu hakkında bilgi verir. Çarpıklık ve basıklık hesaplaması için **psych** paketi ile sunulan `skew()` ve `kurtosi()` fonksiyonları kullanılabilir. Ayrıca **RFast** paketinde sunulan `kurt()` fonksiyonu da basıklık ve çarpıklık değerlerinin hesaplanması için kullanılabilir(Papadakis vd. 2018).

```
require(psych)
require(ggplot2)
with(mtcars, c(cor(mpg,hp),cov(mpg,hp)))
```

```
## [1]   -0.7761684 -320.7320565
```

```
skewness <- skew(mtcars$mpg)
kurtosis <- kurtosi(mtcars$mpg)
c(skewness,kurtosis)
```

```
## [1]  0.610655 -0.372766
```

3.3 Dağılımlar

R'de temel olarak 20 dağılım türüne ait fonksiyonlar sunulmakta ve bu fonksiyonlarda dört seçenek kullanılmasına imkan tanınmaktadır. Bunlardan ilki *olasılık dağılım fonksiyonu* (pdf-d), ikincisi *kümülatif dağılım fonksiyonu* (cdf-p) , üçüncüsü *kantil fonksiyonu* (quantile-p) ve sonuncusu dağılımdan rassal sayı üretmek için kullanılan rassal sayı fonksiyondur (random-r) .

Tablo 3.3 en çok kullanılan dağılımların fonksiyonları ve açıklamada ise bu fonksiyonların nasıl kullanılacağı gösterilmektedir. Mesela tekdüze (uniform) dağılım için *dunif()* fonksiyonu x sayısı için olasılık dağılım fonksiyonu değerini verirken, *punif()* kümülatif dağılım fonksiyonu değerini verecektir. *runif()* ise tekdüze bir dağılımdan rassal bir sayı seçilmesini sağlamaktadır.

```r
# Dağılımlara ilişkin fonksiyonlar
# Uniform distribution
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)

# Binomial distribution
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)

# Normal dağılım
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)

# Hypergeometric Distribution
dhyper(x, m, n, k, log = FALSE)
phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
rhyper(nn, m, n, k)

# Poisson Distribution
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)

# Geometric Distribution
dgeom(x, prob, log = FALSE)
pgeom(q, prob, lower.tail = TRUE, log.p = FALSE)
qgeom(p, prob, lower.tail = TRUE, log.p = FALSE)
rgeom(n, prob)
```

3.3.1 Merkezi Limit Teoremi

İstatistiğin en çok kullanılan teoremlerinden biri olan Merkezi Limit Teoremi , ortalaması μ ve standart sapması σ olan bir anakütleden rassal olarak n büyüklüğünde bir örneklem seçildiğinde, n büyüdükçe örneklemin ortalamasının, ortalaması μ ve standart sapması $\sigma_x = \frac{\sigma}{\sqrt{n}}$ olan normal bir dağılıma yaklaşacağını söyler. Eğer anakütle normal dağılıyorsa,

örneklem büyüklüğünden bağımsız olarak örneklemde normal dağılacaktır. Diğer bir şekilde ifade edilecekse, dağılımı normal olmayan bir anakütleden seçilse bile yeteri kadar büyük bir örneklemin ortalaması anakütle ortalaması μ etrafında normal bir dağılım gösterecektir. Bunun sağladığı en büyük kolaylık, olasılık dağılım fonksiyonunun bilinmediği veya hesaplanmasının çok karmaşık olduğu durumlarda Merkezi Limit teoreminden faydalanarak momentler hesaplanabilmektedir.

Özellikle derslerde merkezi limit teoremini göstermek için çeşitli simülasyonlar yapılmaktadır. Bu simulasyonlar döngüler kullanılarak yapılabileceği gibi **UsingR** paketinde yer alan `simple.sim()` fonksiyonu ile de kullanılabilir(Verzani 2018). Buna örnek olarak öncelikle $\lambda = 6$ olan rassal olarak oluşturulan 1000 gözlemlik bir x veri seti olsun. İkinci aşamada ise `simple.sim()` fonksiyonu kullanılarak rassal olarak 10 bin defa veri seti oluşturulup her birinin ortalamasının alınarak `sim` veri seti oluşturulsun.

```
require(UsingR)
set.seed(1234)
x = rpois(1000, 6)

f <- function() {
  t = rpois(1000, 6)
  y = mean(x)
  return(y)}

sim <- simple.sim(10000,f)
```

Bu veri setlerinin grafikleri Şekil 3.1 ile gösterilmektedir. Görüldüğü üzere sadece değişkeni içeren grafik Poisson dağılımın uygun şekilde çarpık bir dağılıma sahipken, bu değişkenlerin ortalamalarının alındığı sim grafiği merkezi limit teoremine uygun bir şekilde normal bir dağılım göstermektedir.

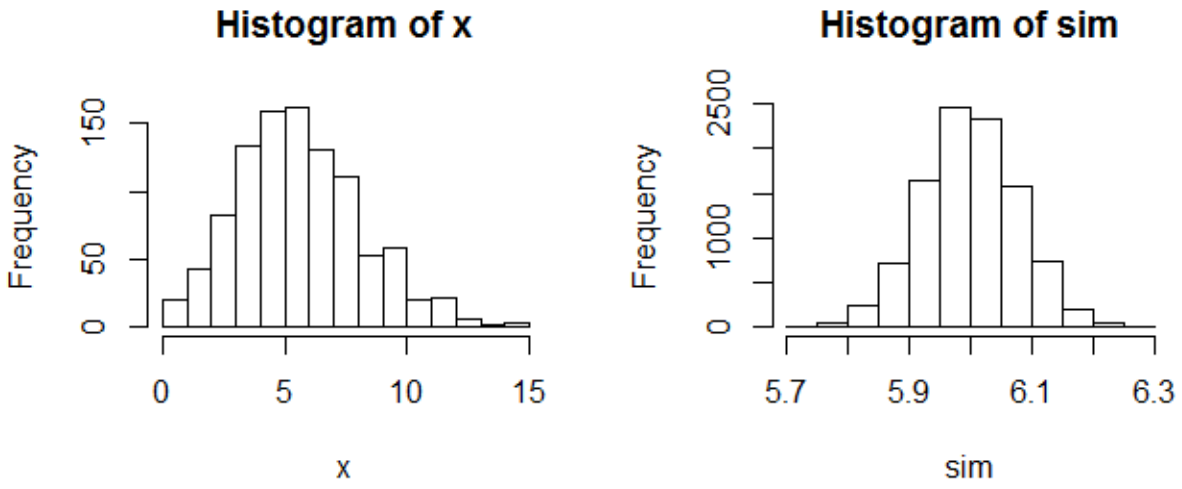
3.4 Güven Aralıkları ve Hipotez Testi

Örneklem ortalaması $\bar{X}$, anakütle ortalamasının μ nokta tahmini olmakla birlikte, tek başına tahmin için yetersiz kalmaktadır. Bu nedenle, tek bir nokta belirlemek yerine genellikle anakütle ortalamasını belirli bir güven aralığında (belirli bir olasılıkla) içerecek bir aralık oluşturulur ve bu aralık güven aralığı olarak adlandırılır.

Eğer varyansı (σ^2) bilinen bir anakütleden seçilen n büyüklüğünde bir örneklem varsa, bu durumda anakütle ortalaması (μ) için oluşturulacak güven aralığı:

$$\bar{X} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \tag{3.1}$$

olarak yazılır. Burada z_α istenilen güven aralığı için belirlenen Z-değerini göstermektedir.



Şekil 3.1: Merkezi limit teoremi simulasyonu

Ancak çoğu zaman anakütlenin varyansı da bilinmediğinden *Student-t dağılımı* kullanılarak anakütlenin normal dağılıma sahip olduğu varsayımı altında örneklem standart hatası (s) kullanılarak güven aralığı oluşturulur.

$$\bar{X} - t_{\alpha/2,df}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{X} + t_{\alpha/2,df}\frac{s}{\sqrt{n}} \tag{3.2}$$

Oranlar için de benzer bir şekilde güven aralıkları hesaplanabilir. Eğer örneklem büyüklüğü n ile oran - p- çarpımı 10'dan büyükse normal bir dağılımdan elde edildiği varsayılır ($np \geq 10$ ve $n(1-p) \geq 10$) ve bu durumda güven aralığı

$$p \pm z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} \tag{3.3}$$

olarak hesaplanır. R'de güven aralıklarının hesaplanması için **teachingDemos** paketi ile sunulan `z.test()` fonksiyonu ve **stats** paketi ile sunulan `t.test()` ve `prop.test()` fonksiyonları kullanılır (Greg Snow 2016). Bu fonksiyonlarda x ve y örneklemleri, *mu* karşılaştırılacak ortalamayı, *alternative* testin tek yönlü veya çift-yönlü olup olmadığını, *conf.level* güven aralıklarını ve *stdev* standart sapmayı (z-testi için) belirtmek için kullanılır. `prop.test()` fonksiyonu için *correct* öğesi ise Yates süreklilik düzeltmesinin uygulanmasını sağlar. `t.test()` fonksiyonunda yer alan *paired* öğesi aynı kişiye ait farklı koşullarda yapılan gözlemleri (paired = TRUE) karşılaştırmak için kullanılır.

```
require(TeachingDemos)
z.test(x, mu = 0, stdev, alternative = c("two.sided", "less", "greater"),
  sd = stdev, n=length(x), conf.level = 0.95, ...)
```

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

prop.test(x, n, p = NULL,
          alternative = c("two.sided", "less", "greater"),
          conf.level = 0.95, correct = TRUE)
```

Eğer parametrik olmayan bir test kullanılmak istenirse `wilcox.test()` fonksiyonu Wilcoxon testi ile kullanılabilir. Bu test aksi tanımlanmazsa ortanca değere göre güven aralığı olşturacaktır. Bu testin öğeleri de yukarıda sunulan öğelerle aynı şekilde tanımlanır.

```
wilcox.test(x, y = NULL,
            alternative = c("two.sided", "less", "greater"),
            mu = 0, paired = FALSE, exact = NULL, correct = TRUE,
            conf.int = FALSE, conf.level = 0.95, ...)
```

Eğer ikiden fazla kategori varsa bu durumda Ki-kare χ^2 testinin kullanılması gereklidir. Ki-kare testi verinin belirli bir anakütleden alınıp alınmadığını (goodness-of fit test) veya kategorik değişkenlerin birbirinden bağımsız olup olmadığını (independence test) test etmek için kullanılır. Özellikle anket sonuçlarını değerlendirmede bu testten faydalanılmakta olup, çok sayıda soru olması halinde boyut azaltmak içinde Çoklu Bağıntı Analizi (Multiple Correspondence Analysis) kullanılmaktadır. uygulanırken kategorilerin birbirinden bağımsız olmasına, her bir faktör değerinde en az 5 gözlem olmasına ve bağımsızlık derecesinin (d.f.) 2'den büyük olmasına dikkat edilmelidir.

Ki-kare testini yapmak için `chisq.test()` fonksiyonu kullanılmaktadır. Bu fonksiyonda x sayısal bir vektör veya matrisi, y varsa karşılaştırılacak faktör veya sayısal vektörü, p ise x ile aynı boyutta olasılık değerlerini içeren vektörü tanımlamak için kullanılır.

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

Bu testler kullanılırken karşılaşılan önemli sorunlardan biri örneklem büyüklüğünün ne kadar olması gerekliğidir. Özellikle de gruplar karşılaştırılacaksa testlerin istatistiki anlamda geçerli olması için belirli bir büyüklüğün üzerinde olması gereklidir. Bunun için **pwr** paketi ile sunulan `pwr.norm.test()`, `pwr.t.test()` ve `pwr.p.test()` fonksiyonları kullanılabilir (Champely 2018). Bu fonksiyonda *power* testin istenilen gücünü *n*, *n1*, *n2* örneklem sayıları, *d* ve *h* etki büyüklüğünü (effect size) belirtmek için kullanılır. Eğer gruplar karşılaştırılacaksa ve örneklem büyüklükleri eşit değilse `pwr.t2n.test()` ve `pwr.2p2n.test()` fonksiyonları kullanılır.

```
# pwr paketi fonksiyonları
require(pwr)
pwr.norm.test(d = NULL, n = NULL, sig.level = 0.05, power = NULL,
              alternative = c("two.sided","less","greater"))
```

```
pwr.t.test(n = NULL, d = NULL, sig.level = 0.05, power = NULL,
           type = c("two.sample", "one.sample", "paired"),
           alternative = c("two.sided", "less", "greater"))

pwr.t2n.test(n1 = NULL, n2 = NULL, d = NULL, sig.level = 0.05,
             power = NULL, alternative = c("two.sided",
                                           "less","greater"))

pwr.p.test(h = NULL, n = NULL, sig.level = 0.05, power = NULL,
           alternative = c("two.sided","less","greater"))

pwr.2p.test(h = NULL, n = NULL, sig.level = 0.05, power = NULL,
            alternative = c("two.sided","less","greater"))

pwr.2p2n.test(h = NULL, n1 = NULL, n2 = NULL, sig.level = 0.05,
              power = NULL,
              alternative = c("two.sided", "less","greater"))
```

3.5 Hipotez Testleri ile Değişkenlerin Değerlendirilmesi

Çok değişkenli analiz yöntemleri veya regresyon gibi daha karışık modeller kullanılmadan önce değişkenlerin değerlendirilmesinde veya örneklemlerin karşılaştırılmasında Z-testi veya t-testi gibi yönemler kullanılarak temel karşılaştırmalar yapılabilmektedir. Özellikle değişken kategorik ise bu testler daha sık karşımıza çıkmaktadır.

Parametrik testlere örnek olarak ilk olarak `mtcars` veri setindeki yakıt tüketimi (mpg) verisi incelenecektir. Farazi olarak, bir galon benzinle 25 mil gidildiği hipotezi test edilsin. Test sonuçlarına göre bir galon benzinle ortalama 20 mil yol gidilebilmektedir.

```
data(mtcars)
summary(mtcars$mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
```

```
t.test(mtcars$mpg, mu = 25, altenative = "two.sided")
```

```
##
##  One Sample t-test
##
## data:  mtcars$mpg
## t = -4.6079, df = 31, p-value = 6.587e-05
## alternative hypothesis: true mean is not equal to 25
```

```
## 95 percent confidence interval:
##  17.91768 22.26357
## sample estimates:
## mean of x
##  20.09062
```

İki örneklemin karşılaştırılması için t-testi kullanılırken varyansların eşit olup olmadığı varsayımına (var.equal) dikkat edilmeldir. Buna örnek olarak 50 bin elmasa ait verinin bulunduğu `diamonds` veri setinde "Fair" ve "Premium" olarak kesilmiş elmasların ortalama fiyatlarının, eşit olmayan varyans, varsayımı altında karşılaştırılması örneği verilmiştir.

```
require(knitr)
require(dplyr)
require(ggplot2)
data(diamonds)
head(diamonds,3)
fair <- diamonds %>%
  filter(cut == "Fair") %>%
  select(price)
premium <- diamonds %>%
  filter(cut == "Premium") %>%
  select(price)
t.test(fair$price, premium$price, var.equal = FALSE)
```

Sonuçlarda görüldüğü üzere p-değeri 0.019 olup %5 anlamlılık düzeyinde "Fair" ve "premium" kesime sahip elmasların ortalama fiyatlarının birbirine eşit olduğunu ileri süren sıfır hipotezi reddedilmektedir.

```
Welch Two Sample t-test
data:  fair$price and premium$price
t = -2.3453, df = 2210.6, p-value = 0.0191
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -414.05655  -36.94333
sample estimates:
mean of x mean of y
 4358.758  4584.258
```

Parametrik olmayan teste örnek olarak **UsingR** paketinde yer alan ve 199 CEO maaşlarını gösteren `exec.pay` veri seti kullanılacaktır. Veri setinin özelliklerine bakıldığında ortalamasının 60 bin dolar olduğunu ancak verinin aşırı çarpık olduğu görülmektedir. Bu nedenle normal dağılım varsayımı altına kullanılan testler yerine parametrik olmayan Wilcoxon testi kullanılmalıdır.

```
require(UsingR)
data(exec.pay)
summary(exec.pay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   14.00   27.00   59.89   41.50 2510.00
```

```
wilcox.test(exec.pay, conf.int = TRUE)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  exec.pay
## V = 19306, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 0
## 95 percent confidence interval:
##  25.99998 32.99994
## sample estimates:
## (pseudo)median
##       29.00002
```

Test sonuçlarından görüldüğü üzere %95 güven aralığı 25-32 bin dolar arasındadır.

Eğer iki kategoriden oluşan bir veri varsa bunların kıyaslanması için `prop.test()` fonksiyonu örneklem ortalamalarının karşılaştırılması için kullanılabilir.

Mesela 995 kişi ile yapılan bir araştırmada gençlerin yaşlılardan daha az horlayıp horlamadığına bakılmak istensin. Katılımcılardan 220'ü 40 yaş ve altı (genç grubu), 775'i ise 40 yaşından büyük (yaşlı grubu) olsun. Yaşlı gruptaki katılımcıların 320'si, genç gruptakilerin ise 40'ının bu sorundan muzdarip olduğunu varsayalım. Bu durumda yapılacak test şu şekilde olacaktır:

```
prop.test(x = c(40,320), n = c(220,775), alternative = "two.sided")
```

```
##
##  2-sample test for equality of proportions with continuity
##  correction
##
## data:  c(40, 320) out of c(220, 775)
## X-squared = 38.635, df = 1, p-value = 5.11e-10
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.2956399 -0.1665302
## sample estimates:
##    prop 1    prop 2
## 0.1818182 0.4129032
```

Test sonuçlara bakıldığında p-değeri çok küçük olduğundan genç ve yaşlı grubu arasında horlamadan muzdarip olanlar arasında istatistiki olarak fark olmadığını ileri süren sıfır hipotezi reddedilmiştir. Ayrıca bu testin sonucunda %95 güven aralığındaki değerler de verilmiştir.

Diğer bir örnek olarak eğitime katılım ve etnik durum üzerine bir araştırmada belirli bir etnik grubun kızlarının okula göndermediği yönünde bir tez verilebilir. Bu örnek için **MASS** paketinde yer alan `quine` veri seti (Avustralya'da bir bölge okulundaki 146 öğrencinin bilgileri

yer almaktadır) kullanılmaktadır. Öncelikle `table()`fonksiyonu kullanılarak cinsiyet ve etnik duruma ilişkin tablo oluşturulmuş, sonra bu tablo kullanılarak `prop.test()` fonksiyonu süreklilik düzeltmesi olmadan ve %95 güven aralığı ile iki yönlü olarak test edilmiştir.

```
library(MASS)
data(quine)
head(quine)
```

```
##   Eth Sex Age Lrn Days
## 1   A   M  F0  SL    2
## 2   A   M  F0  SL   11
## 3   A   M  F0  SL   14
## 4   A   M  F0  AL    5
## 5   A   M  F0  AL    5
## 6   A   M  F0  AL   13
```

```
attach(quine)
tab <- table(Eth,Sex)
prop.test(tab, alternative = "two.sided", conf.level = 0.95,
          correct = FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  tab
## X-squared = 0.0040803, df = 1, p-value = 0.9491
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.1564218  0.1669620
## sample estimates:
##    prop 1    prop 2
## 0.5507246 0.5454545
```

```
detach(quine)
```

Bu örnekte sıfır hipotezi kız çocuklarının eğitime katılımında etnik olarak bir fark olmadığı şeklindedir. Söz konusu sonuçlara bakıldığında p-değeri 0.94 gibi çok yüksek bir değer çıktığından sıfır hipotezi reddedilememekte ve kız çocuklarının eğitime katılım oranında etnik bakımdan istatistiki olarak bir fark olmadığı görülmektedir.

`prop.test()` fonksiyonu matrisler ile de kullanılabilir. Bu durumda satır ve sütun isimlerinin belirtilmiş olması gereklidir. Ayrıca elde edilen değerleri `prop.table()` fonksiyonu ile oransal olarak göstermek mümkündür. Örnek olarak emniyet kemeri kullananlar ile kullanmayanların kazadan sağ çıkma oranlarının olduğu bir araştırmada aşağıdaki sonuçlar elde edilmiş olsun. Bu sonuçlar ile emniyet kemeri kullananların ve kullanmayanların vefat oranlarının farklı olmadığı yönündeki sıfır hipotezi reddedilmektedir.

```r
kaza <- matrix(c(178,144,135,47), ncol=2)
colnames(kaza) <- c('Hayatta','Vefat')
rownames(kaza) <- c('Kemersiz','Kemerli')
prop.table(kaza)
```

```
##            Hayatta      Vefat
## Kemersiz 0.3531746 0.26785714
## Kemerli  0.2857143 0.09325397
```

```r
prop.test(kaza)
```

```
##
##  2-sample test for equality of proportions with continuity
##  correction
##
## data:  kaza
## X-squared = 16.848, df = 1, p-value = 4.05e-05
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.27155920 -0.09891401
## sample estimates:
##    prop 1    prop 2
## 0.5686901 0.7539267
```

Ki-kare testine örnek olarak, hepatit hastalığı ile dövme yaptırılan yer (Ruhsatlı, Ruhsatsız ve Dövme yaptırılmamış) ilişkisine dair bir çalışmada elde edilen sonuçlar şu şekilde olsun:

```r
hep <- matrix(c(17,35,8,53,22,491), ncol = 2)
colnames(hep) <- c('Pozitif','Negatif')
rownames(hep) <- c('Ruhsatlı Yer','Diğer','Dövme Yok')
prop.table(hep)
```

```
##                 Pozitif    Negatif
## Ruhsatlı Yer 0.02715655 0.08466454
## Diğer        0.05591054 0.03514377
## Dövme Yok    0.01277955 0.78434505
```

```r
chisq.test(hep)
```

```
##
##  Pearson's Chi-squared test
##
## data:  hep
## X-squared = 230.76, df = 2, p-value < 2.2e-16
```

Elde edilen test sonuçlarına göre dövme yaptırma ve hepatit hastalığı arasında ilişki olmadığına dair sıfır hipotezi %5 önem düzeyinde reddedilmektedir.

3.6 Varyans Analizi - ANOVA

Yukarıda kısaca anlatıldığı gibi iki bağımsız örneklemin ortalamalarının karşılaştırılması için t-testi kullanılmaktadır. Eğer ikiden fazla grup varsa Varyans Analizinden (Analysis of Variance-ANOVA) faydalanılır. Mesela yukarıda verilen elmas örneğinde elmasların kesimine göre ikili karşılaştırma yapmaktansa kesimlerin ortalama fiyatlarının birbirine eşit olduğu hipotezini test etmek için ANOVA kullanılır. ANOVA ve bunun çok değişkenli modeli olan MANOVA modelleri doğrusal modeller olduğundan faktörlerin (kategorik değişkenlerin) modele etkisi varyansa yapılan katkı (karelerinin toplamı) kullanılarak ortaya çıkarılmaya çalışılmaktadır.

ANOVA'daki temel mantık, k sayıda gruptan ve toplam N gözlemden oluşan bir örneklemde, grupların ortalamaları μ_i birbirine istatistiki olarak eşitse, bunlar aynı zamanda anakütlelerin ağırlıklı ortalamasından oluşan bir büyük ortalamaya (grand mean - $\bar{\mu}$) eşit olacaktır. Anakütle varyanslarının birbirine eşit olduğu ve her bir gözlemin birbirinden bağımsız olarak elde edildiği varyasımı altında grupların büyük ortalamadan farkının kareleri toplamının (SS_B) grupların kendi ortalamalarından farkının kareleri toplamına (SS_W) oranı F-dağılımı gösterecek ve bu değerin belirli bir kritik eşiği aşması halinde grupların ortalamalarının birbirine eşit olmadığı sonucu çıkacaktır.

Tablo 3.4 ANOVA F-testine ilişkin değerleri göstermektedir. Eğer F-test değeri kritik eşiğin üzerinde ise veya p-değeri kritik eşiğin altında ise grup ortalamalarının birbirine eşit olduğu sıfır hipotezi reddedilir.

Tablo 3.4: ANOVA Tablosu

Varyansın Kaynağı	Kareler Toplamı	S.d.	Ortalama Kareler	Test is.	Olasılık Değ.
Gruplar arası	SSB	k - 1	MSB	F	p
Grup içi	SSW	N - k	MSW		
Toplam	SST	N - 1			

Eğer bir faktör altında ikiden fazla grup bulunuyorsa ve sadece bu faktöre göre karşılaştırma yapılacaksa buna *tek yönlü ANOVA* (one-way ANOVA), iki faktörün etkisine bakılacaksa *iki yönlü ANOVA* (two-way ANOVA) söz konusudur. İki yönlü ANOVA'da iki faktör kullanıldığından her bir faktörün ana etkisi (main effect) ile birlikte kategorik değişkenlerin kendileri arasında etkileşimin etkisinin (interaction effect) de incelenmesi gereklidir. Faktör A ve Faktör B'nin etkisinin değerlendirildiği iki faktörlü bir modelde A ve B ana etkileri (main effect) ve AB etkileşim etkisi (interaction effect) olmak üzere üç etki olacaktır. Bu etkilerin istatistiki olarak anlamlı olup olmadığını değerlendirmek için F-testi kullanılarak faktörün etkisinin dahil olduğu ve olmadığı modeller karşılaştırılmakta ve buna göre değerlendirme yapılmaktadır. Eğer istatistiki olarak anlamlı bir etkileşim etkisi varsa bu durumda faktörlerin ana etkisini değerlendirmek tek başına yeterli olmayacaktır.

Faktör A ve Faktör B'nin kullanıldığı bir modelde ana ve etkileşim etkileri şu şekilde hesaplanır:

- Etkileşimin Etkisi: SS(AB|A,B) = SS(A,B,AB) – SS(A,B)

- Faktör A'nın etkisi: SS(A|B,AB) = SS(A,B,AB) – SS(B,AB)
- Faktör B'nin etkisi: SS(B|A,AB) = SS(A,B,AB) – SS(A,AB)
- Etkileşimsiz Faktör A'nın etkisi: SS(A|B) = SS(A,B) – SS(B)
- Etkileşimsiz Faktör B'nin etkisi: SS(B|A) = SS(A,B) – SS(A)

Bu etkilerin hesaplanması için altı tip kareler toplamı geliştirilmiştir. Bunlar sırayla Tip I, Tip II şeklinde devam etmekte olup, en yaygın kullanılanları Tip II ve Tip III kareler toplamıdır.

Tip I Kareler Toplamı: Tip I kareler toplamında , her etkinin karelerin toplamı, etkinin dahil olduğu modelden elde edilen kareleri toplamı ile etkinin dahil olmadığı modelden elde edilen kareleri toplamı arasındaki fark ile hesaplanır. Her etki için anlamlılık testleri de etki tarafından hesaplanan tahmini karelerin toplamında artış üzerinde gerçekleştirilir. Tip I kareler toplamı bu yüzden bazen sıralı veya hiyerarşik kareler toplamı olarak adlandırılır.Bu yöntemde öncelikle `SS(A)` hesaplanarak Faktör A'nın etkisi, sonra `SS(B|A)` hesaplanarak Faktör B'nin etkisi ve en son `SS(AB|B,A)` ile etkileşimin etkisi hesaplanır.

Tip I'in kareler toplamının önemli bir özelliği, her bir etkiye atfedilebilecek karelerin toplamının, tüm model kareleri toplamını oluşturmasıdır. Böylece, Tip I kare toplamı, tüm model için öngörülen kareler toplamı için ayrıştırma sağlar. Bu diğer tiplere göre bazı avantajlar sağlasa da, belirli bir etkiye atfedilebilecek karelerin toplamının, genellikle, etkilerin modele girildiği sıraya bağlı olmasına yol açmakta ve bu da Tip I'in bazı modellerde hipotezlerin test edilmesi için kullanılabilirliğini sınırlamaktadır.

Tip II Kareler Toplamı: Tip II Kareler toplamı bazen kısmi sıralı kareler toplamı olarak da adlandırılır. Bu tip kareler toplamı Tip I'de olduğu gibi bir etkiyi diğer etkilere göre hesaplar. Bununla birlikte, bir etki için karelerin toplamı, eşit veya daha düşük derecedeki tüm diğer etkilerin etkisini kontrol ederek hesaplanır. Yani Faktör A'nın etkisi için `SS(A|B)`, Faktör B'nin etkisi için `SS(B|A)` ve etkileşimin hesaplanması için `SS(AB|B,A)` kullanılmaktadır. Tip I kareleri toplamından farklı olarak, Tip II kareleri toplamında etkilerin modele giriş sırasının önemi yoktur, bu sebeple çoklu regresyon, ANOVA ve MANOVA modellerinin değerlendirilmesinde daha çok tercih edilmektedir. Ancak, burada dikkat edilmesi gereken husus, modelin genel olarak oluşturulması yani etkileşim etkisini de içerecek şekilde oluşturulması ve etkileşim anlamlı değilse ana etkilerin incelenmesine geçilmesi gereklidir.

Tip III Kareler Toplamı: Tip II kareler toplamının dezavantajlarından birisi kategorilerdeki gözlem sayısının dengeli olmaması halinde hatalı değerlendirmelere yol açmasıdır. Ayrıca etkileşim etkisi olduğu zaman modelin genel oluşturulması ve anlamlı bir etkileşim etkisi yoksa ana etkilerin incelenmesine geçilmelidir. Tip III Kareler Toplamı bu dezavantajlar için geliştirilmiştir. Bu yöntemde Faktör A'nın ana etkisi için `SS(A|B,AB)` ve Faktör B'nin ana etkisi için `SS(B|A,AB)` kullanılmakta, yani etkileşim etkisi de hesaplanmaktadır. Bu nedenle, bu yöntem istatistik programları tarafından da hesaplamalar için tercih edilen yöntemdir.Tip III'in yönteminde karşılaşılan bazı eksikliklerinin giderilmesi için Tip VI ve Tip V kareleri toplamı da geliştirilmiştir.

Ancak diğer programlarda olduğu gibi R'de sunulan fonksiyonlarda Tip II ve Tip III kareler toplamı tercih edildiğinden, Tip VI ve Tip V kareler toplamı için kullanıcı tarafından fonksiyon yazılması gerekmektedir.

ANOVA doğrusal bir model olduğundan kullanılacak modelin bir formül olarak belirtilmesi gereklidir. Tablo 3.5 ANOVA modeli oluşturulurken kullanılacak formüllerin yazımı ve bunların açıklamasını göstermektedir. Ayrıca ANOVA uç değerlere duyarlı olduğundan uç değerlerin dikkate alınması gereklidir.

Tablo 3.5: ANOVA'da Kullanılan Formüller

Formül	Açıklaması
$Y \sim X$	X Y'nin açıklayıcısıdır
$Y \sim X + Z$	X ve Z, Y'nin açıklayıcısıdır
$Y \sim X * Z$	X, Z, X $\times$ Z Y'nin açıklayıcıdır
$Y \sim X \vert Z$	X, Z veri iken Y'nin açıklayıcısıdır

R'de ANOVA için `aov()` ve `anova()` fonksiyonları kullanılmaktadır.İlk fonksiyonda *formula* öğesi kullanılacak formülü, *data* veri setini tanımlamak için kullanılır.Bufonksiyon ile olşturulan nesne `summary()` fonksiyonu ile görüntülenebilir. `anova()`fonksiyonu için ise öncelikle `lm()`fonksiyonu ile doğrusal bir model oluşturulması ve bu modelin `anova` fonksiyonunda kullanılması gerekir. İkinci fonksiyonun avantajı tekrar `summary()` fonksiyonuna ihtiyaç duyulmamaktadır.

```
aov(formula, data = NULL, projections = FALSE, qr = TRUE,
    contrasts = NULL, ...)

anova(object, ...)

lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

ANOVA için başka bir fonksiyonda **car** paketinde yer alan `Anova()` fonksiyonudur. Bu fonksiyon içinde yine `lm()` fonksiyonu kullanılarak doğrusal bir model oluşturulması gereklidir. Bu fonksiyonda yukarıdakilerden farklı olarak Tip-2 ve Tip-3 Kareler Toplamı *type* öğesi ile seçilebilmektedir. Yukarıdaki fonksiyonlar Tip-1 Kareler Toplamına göre hesaplandığından iki yönlü ANOVA kullanılarak elde edilen sonuçlara dikkat edilmelidir.

```
Anova(mod, type=c("II","III", 2, 3),
    test.statistic=c("LR", "Wald", "F"),
    error, error.estimate=c("pearson", "dispersion", "deviance"),
    vcov.=NULL, singular.ok, ...)
```

Eğer tek yönlü ANOVA modelinde faktörün etkisi anlamlı çıktıysa, gruplar arasındaki karşılaştırmaların yapılması için `pairwise.t.test()` fonksiyonu kullanılabilir. Bu fonksiyonda x öğesi bağımlı değişken vektörünü, g öğesi grup değişkenleri, *pool.sd* öğesi havuzlanmış standart sapma (pooled SD) kullanılıp kullanılmayacağını (grupların varyansının aynı ise TRUE olacaktır) ve *p.adjust.method* öğesi ise Tip-1 hata oranını düzenlemek için kullanılacak yöntemi ("none", "bonferroni", "holm", "hochberg", "hommel", "BH", veya "BY") tanımlamak için kullanılır. Bu yöntemlerden Bonferroni yöntemi en çok tercih edilen yöntemlerdendir.

```
pairwise.t.test(x, g, p.adjust.method = p.adjust.methods,
                pool.sd = !paired, paired = FALSE,
                alternative = c("two.sided", "less", "greater"),
                ...)
```

ANOVA hata terimlerinin (residual) normal dağıldığı ve grup varyanslarının eşit olduğu varsayımı altında kullanılmaktadır. Grup varyanslarının eşitliğini yani homojen olup olmadığını kontrol etmek için `leveneTest()` fonksiyonu kullanılabilir. Bu fonksiyonda *y* öğesi ANOVA formulü şeklinde yazılırken, *data* öğesinde de kullanılacak veri seti tanımlanır. Elde edilen sonuçlarda p-değeri 0.05'in altında ise eşit varyans varsayımı reddedilir. Hata terimlerinin dağılımı için ise ANOVA modeli oluşturulduktan sonra `res()`fonksiyonu ile elde dilen hata terimlerinin dağılımına bakılması gerekir.

```
leveneTest(y, data, ...)
```

ANOVA modeli oluşturulduktan sonra gruplar arasındaki farklılıkları görmek için Tukey Honest Significant Differences (Tukey HSD) testi uygulanmaktadır. Bunun için `TukeyHSD`fonksiyonu kullanılmakta olup, bu fonksiyonda *x* öğesi oluşturulan `aov()` fonksiyonu ile oluşturulan ANOVA nesnesini, *which* öğesi ile hangi faktöre göre karşılaştırma yapılacağı, *ordered* öğesi ile bu faktörün sıralı olup olmadığı (TRUE sıralı demektir) ve *conf.level* ile grup-güven aralıkları tanımlanmaktadır. Bu test için **agricolae** paketinde yer alan `HSD.test()`fonksiyonu da kullanılabilir.

```
TukeyHSD(x, which, ordered = FALSE, conf.level = 0.95, ...)
```

İki-yönlü ANOVA kullanılıyorsa ve etkileşim varsa etkileşim etkisini görmek için `intercation.plot()` fonksiyonu ile grafik hazırlanabilir. Bu fonksiyonda *x.factor* öğesi grafiğin X-ekseninde bulunacak faktörü, *trace.factor* diğer faktörü ve *response* ise açıklanan değişkeni gtanımlamak için kullanılır. Diğer öğeler ise temel grafik fonksiyonlarında olduğu gibi tanımlanır.

```
interaction.plot(x.factor, trace.factor, response, fun = mean,
                 type = c("l", "p", "b", "o", "c"), legend = TRUE,
                 trace.label = deparse(substitute(trace.factor)),
                 fixed = FALSE,
                 xlab = deparse(substitute(x.factor)),
                 ylab = ylabel,
                 ylim = range(cells, na.rm = TRUE),
                 lty = nc:1, col = 1, pch = c(1:9, 0, letters),
                 xpd = NULL, leg.bg = par("bg"), leg.bty = "n",
                 xtick = FALSE, xaxt = par("xaxt"), axes = TRUE,
                 ...)
```

Etkileşim etkileri anlamlı çıktığı zaman ana etkilerin tek başına değerlendirilmemesi gerekir. Bu nedenle **emmeans** paketinde yer alan fonksiyonlar kullanılarak etkileşim ve ana etkiler değerlendirilir. Ancak **emmeans** paketinde yer alan fonksiyonlar `Anova()` fonksiyonu ile elde edilen nesnelerin kullanılmasın uygun olmadığı için **emmeans** paketinin bir önceki versiyonu

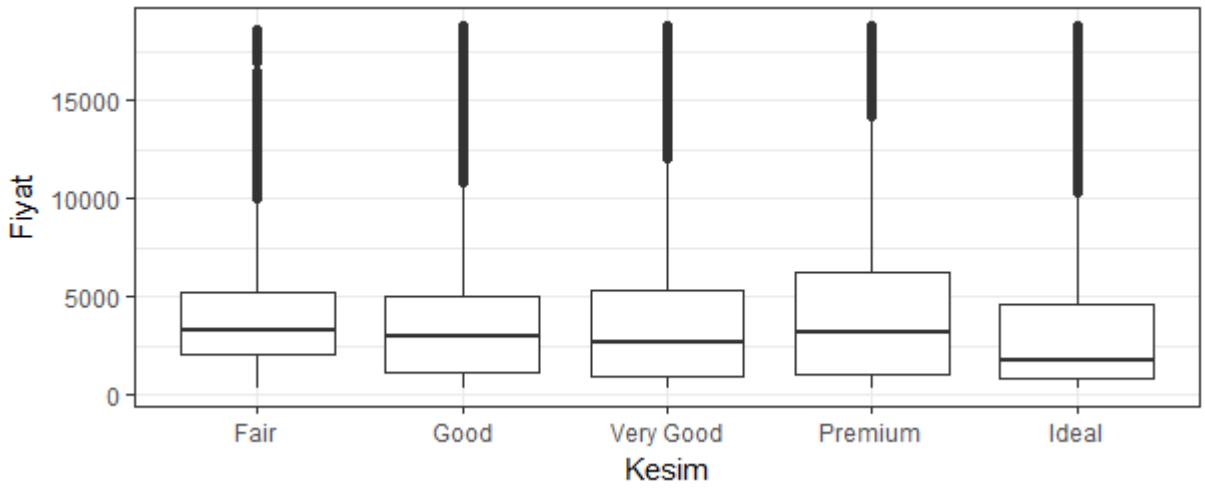
olan **lsmeans** paketi kullanılabilir. Bu pakette yer alan `lsmeans()` fonksiyonu kullanılır. Burada *object* mesnesi için oluşturulan model ve *specs* kısmına ise *pairwise ~ Etkileşim terimi* tanımlanarak kullanılır. Burada *adjust* öğesi "tukey", "scheffe" , "sidak", "bonferroni" ile p-değerinin düzeltilmesi yapılabilir.

```
lsmeans(object, specs, ...)
```

3.6.1 ANOVA örneği

Örnek olarak `diamonds` veri setinde elmas kesimine göre ortalama fiyatların farklı olup olmadığını ANOVA modeli ile incelenecektir. örnek model olduğundan uç değerlere ve normal dağılım varsayımını sağlayıp sağlamadığı kontrol edilmemiştir. Öncelikle grafik ile değerlere bakıldığında Şekil 3.2 ile gösterildiği üzere ortalama fiyatlar birbirine çok yakın olmakla birlikte kesime göre fiyatlar ciddi farklılıklar da göstermektedir.

```
require(ggplot2)
data(diamonds)
ggplot(data = diamonds) + geom_boxplot(aes(cut,price))+
  theme_bw() + xlab("Kesim") + ylab("Fiyat")
```



Şekil 3.2: Elmas kesimi ve fiyatları

Ortalama fiyatların birbirine eşit olup olmadığını test etmek için `aov()` fonksiyonu kullanılarak ANOVA yapılırsa şu sonuçlar elde edilecektir:

```
data(diamonds)
anova.1 <- aov(price~cut, data = diamonds)
summary(anova.1)
```

```
##                Df    Sum Sq   Mean Sq F value Pr(>F)
## cut             4 1.104e+10 2.760e+09   175.7 <2e-16 ***
```

```
## Residuals   53935 8.474e+11 1.571e+07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(anova.1)
```

```
## Call:
##    aov(formula = price ~ cut, data = diamonds)
##
## Terms:
##                         cut    Residuals
## Sum of Squares   11041745359 847431390159
## Deg. of Freedom            4        53935
##
## Residual standard error: 3963.847
## Estimated effects may be unbalanced
```

`anova()`fonksiyonu ile ANOVA yapılırsa şu sonuçlar elde edilecektir:

```
fit <- lm(price~cut, data = diamonds)
anova.2 <- anova(fit)
anova.2
```

```
## Analysis of Variance Table
##
## Response: price
##              Df     Sum Sq    Mean Sq F value    Pr(>F)
## cut           4 1.1042e+10 2760436340  175.69 < 2.2e-16 ***
## Residuals 53935 8.4743e+11   15712087
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`Anova()` fonksiyonu ile ANOVA yapılırsa şu sonuçlar elde edilecektir:

```
require(car)
fit <- lm(price ~ cut, data = diamonds)
anova.3 <- Anova(fit)
anova.3
```

```
## Anova Table (Type II tests)
##
## Response: price
##               Sum Sq    Df F value    Pr(>F)
## cut       1.1042e+10     4  175.69 < 2.2e-16 ***
## Residuals 8.4743e+11 53935
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Söz konusu sonuçlardan da görüldüğü üzere p-değeri çok küçük olup kesime göre ortalama

fiyatların birbirine eşit olduğu sıfır hipotezi reddedilmekte ve ortalama fiyatlardan en az birisinin diğerlerine eşit olmadığı sonucu ortaya çıkmaktadır.

Faktörün etkisi istatistiki olarak anlamlı çıktığından grupların karşılaştırılması için `pairwise.t.test()` fonksiyonu kullanılabilir. Bu fonksiyonda hata düzeltme yöntemi "bonferroni" seçilmiştir. Sonuçlardan, Fair-Premium ve Good-Very Good gruplarının birbirinden istatistiki olarak farklı olmadığı ancak diğerlerinde istatistiki olarak anlamlı bir farklılığın olduğu görülmektedir.

```
pairwise.t.test(diamonds$price, diamonds$cut,
                p.adjust.method = "bonferroni")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  diamonds$price and diamonds$cut
##
##           Fair    Good    Very Good Premium
## Good      0.0016  -       -         -
## Very Good 0.0034  1.0000  -         -
## Premium   0.3077  < 2e-16 < 2e-16   -
## Ideal     < 2e-16 5.7e-13 < 2e-16   < 2e-16
##
## P value adjustment method: bonferroni
```

Faktördeki gruplar arasında farkı görmek için TukeyHSD testi uygulandığında gruplar ortalamasındaki farklar ve buna ilişkin güven aralıkları ve p-değerleri gösterilmektedir. Görüldüğü üzere yukarıdaki sonuçla tutarlı şekilde Fair-Premium ve Good-Very Good arasında farklar istatistiki olarak anlamlı fark olmadığı, diğer gruplar arasında ise bu farkın olduğu görülmektedir.

```
TukeyHSD(anova.1, ordered = FALSE)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = price ~ cut, data = diamonds)
##
## $cut
##                         diff         lwr        upr     p adj
## Good-Fair          -429.89331  -740.44880  -119.3378 0.0014980
## Very Good-Fair     -376.99787  -663.86215   -90.1336 0.0031094
## Premium-Fair        225.49994   -59.26664   510.2665 0.1950425
## Ideal-Fair         -901.21579 -1180.57139  -621.8602 0.0000000
## Very Good-Good       52.89544  -130.15186   235.9427 0.9341158
## Premium-Good        655.39325   475.65120   835.1353 0.0000000
## Ideal-Good         -471.32248  -642.36268  -300.2823 0.0000000
```

```
## Premium-Very Good   602.49781   467.76249   737.2331 0.0000000
## Ideal-Very Good    -524.21792  -647.10467  -401.3312 0.0000000
## Ideal-Premium     -1126.71573 -1244.62267 -1008.8088 0.0000000
```

Elmas fiyatlarının belirleyicisi olarak kesim haricinde renk (color) değişkenin de rol oynadığı hipotezini test etmek için yine ANOVA kullanalım. Burada iki faktör söz konusu olduğundan iki yönlü ANOVA kullanılacaktır. Ancak iki faktör arasındaki etkileşiminde dikkate alınması gereklidir. Oluşturulan modelin sonuçları aşağıdaki gibi olacaktır:

```
require(car)
fit.2 <- lm(price ~ cut*color, data = diamonds)
anova.4 <- Anova(fit.2, type = "III")
anova.4
```

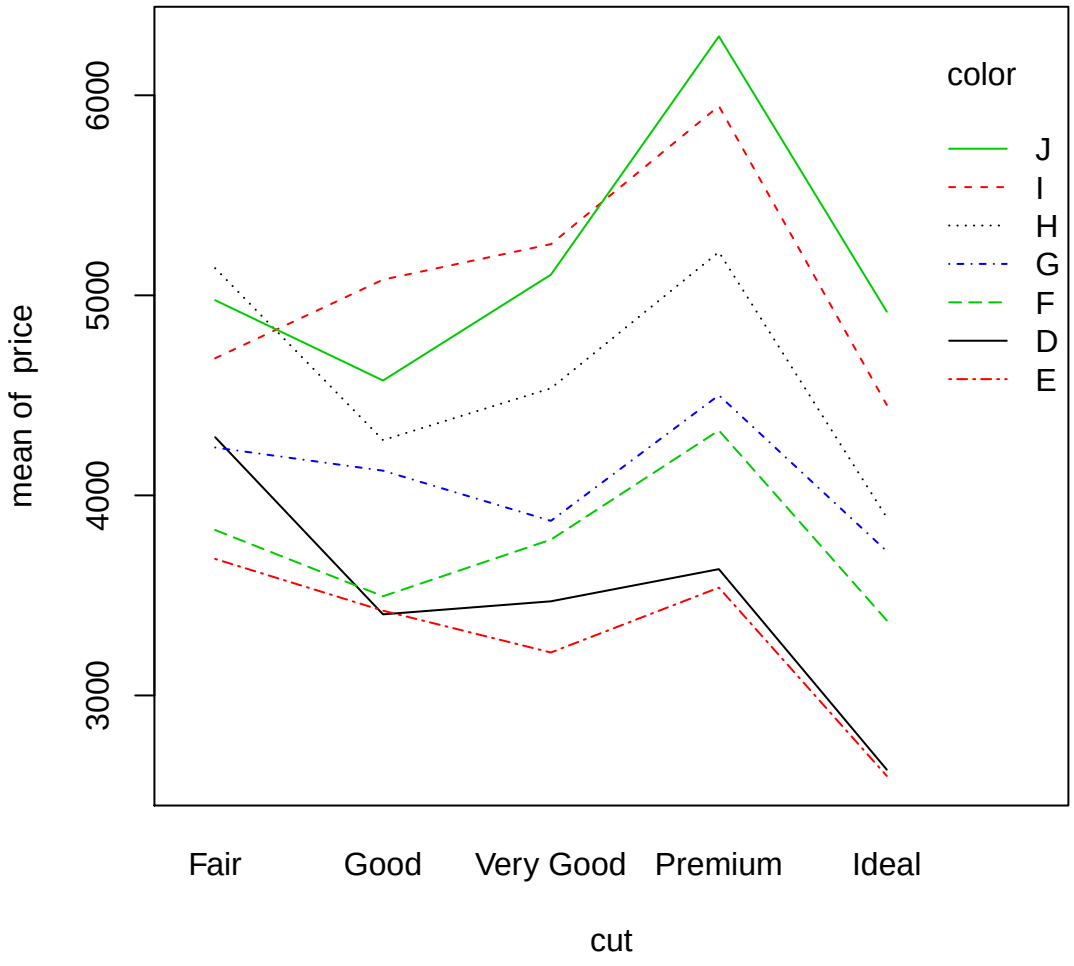
```
## Anova Table (Type III tests)
##
## Response: price
##                 Sum Sq    Df    F value    Pr(>F)
## (Intercept) 3.7606e+11     1 24713.0009 < 2.2e-16 ***
## cut         8.7891e+09     4   144.3969 < 2.2e-16 ***
## color       9.4602e+09     6   103.6142 < 2.2e-16 ***
## cut:color   1.6535e+09    24     4.5274 1.001e-12 ***
## Residuals   8.2027e+11 53905
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Bu sonuçlara göre etkileşim etkisi istatistiki olarak anlamlı olduğundan faktörlerin etkisini doğrudan değerlendirmek yerine her bir cut veya color için ayrı ayrı değerlendirmek gereklidir. Grafik 3.3 incelenen veri seti için etkileşim grafiğini göstermektedir. Paralel seyretmeyen çizgiler ve kırıklardan görüldüğü üzere önemli etkileşim etkisi mevcuttur.

```
with(diamonds,{
     interaction.plot(cut, color, price, col = c(1:4))})
```

Etkileşimin etkisini değerlendirmek için `lsmeans` fonksiyonu ile karşılaştırma yapılmış ve `cld` fonksiyonu ile karşılaştırmanın kolay olması için harflerle gösterimi sağlanmıştır. Tabloda yer alan son sütun (.group), aynı harfleri taşıyan grupların ortalamaları istatistiki olarak birbirinden farklı değildir şeklinde yorumlanır. Tablo çok fazla yer kapladığından burada gösterilmemiş, kontrol edilmesi okuyucuya bırakılmıştır.

```
require(lsmeans)
model <- lsmeans(fit.2, pairwise~cut:color, adjust = "tukey")
cld(model, alpha=0.05, Letters=letters, details= FALSE)
```



Şekil 3.3: ANOVA etkileşim etkisi grafiği

Bölüm 4

Açıklayıcı Veri Analizi

Temel istatistiki analizlerden sonra Açıklayıcı Veri Analizi (Exploratory Data Analysis-EDA) gelmektedir. Çözümleyici Veri Analizi veya Nitel Veri Analiz olark da Türkçe'ye çevrilen Açıklayıcı Veri Analizi, Amerikan Çevre Koruma Ajansı (EPA) tarafından verinin genel yapısını görmeye ve beklenmeyen dağılımları tespit etmeye yarayan bir analiz olarak tanımlamaktadır. Açıklayıcı veri analizinin diğer bir tanımı ise görsel teknikleri kullanarak verinin dağılımını anlamak, uç değerleri tespit etmek, kullanılacak modellerin varsayımları ile veriyi karşılaştırmak için kullanılacak tekniklerin bütünü olarak değerlendirmektedir. Açıklayıcı veri analizi 1977'de ünlü matematikçi ve istatistikçi John Tukey'in yazdığı "Exploratory Data Analysis" kitabı ile genel kabul gören bir uygulama haline gelmiş ve ilerleyen dönemde bilgisayarların ve analiz programlarının gelişmesiyle modellemede temel bir unsur olmuştur.

Açıklayıcı veri analizinde değişkenin kendi içindeki ve değişkenler arasındaki ilişkiler ortaya çıkarılmaya çalışıldığından aşağıda sıralanan sorulara yanıt aranır (Wickham 2017b):

1. Değişkenin en sık ve en az aldığı değerler nelerdir?
2. Bu değerler değişkene ilişkin beklentilerle uyuşuyor mu?
3. Beklentilerle uyuşmayan durum neden kaynaklanmaktadır?
4. Değişkende veya değişkenlerde kümelenme var mıdır?
5. Varsa kümeler hangi yönden benzerlik hangi yönden farklılık göstermektedir?
6. Bu kümelenme beklentilerle uyuşmakta mıdır?
7. Kümelenmeyi açıklayabilecek başka değişkenler var mıdır?
8. Uç değerler neden ortaya çıkmaktadır? Bu verinin derlenmesinden kaynaklanan bir olgu mudur?
9. İki değişken arasındaki ilişki beklentilerle uyuşmakta mıdır?
10. Bu ilişkinin derecesi/düzeyi ne kadardır?
11. Bu ilişkiyi etkileyebilecek başka değişkenler olabilir mi?

Bu sorulara verilecek yanıtlar değişkenler arasındaki istatistiksel ilişkileri ortaya çıkararak değişkenler arasında matematiksel (fonksiyonel) ilişki oluşturmayı sağlayan regresyon teknikleri için altyapıyı hazırlamaktadır. Uygulamada istatistiki ilişkiler belli hususları açıklamaya yardımcı olsa da değişkenler arasında nedensellik ilişkisini açıklamak için yeterli olmadığından, bunun matematiksel ilişki haline dönüştürülmesi gerektiği hususu gözden kaçırılmamalıdır.

4.1 Grafiklerin Oluşturulması

Grafiklerin oluşturulması için ikinci bölümde detaylı olarak anlatılan **ggplot2** paketi ve bu uygulamaları geliştiren **GGally** paketi ile R diğer programlara kıyasla daha gelişmiş uygulamalar sunmaktadır.

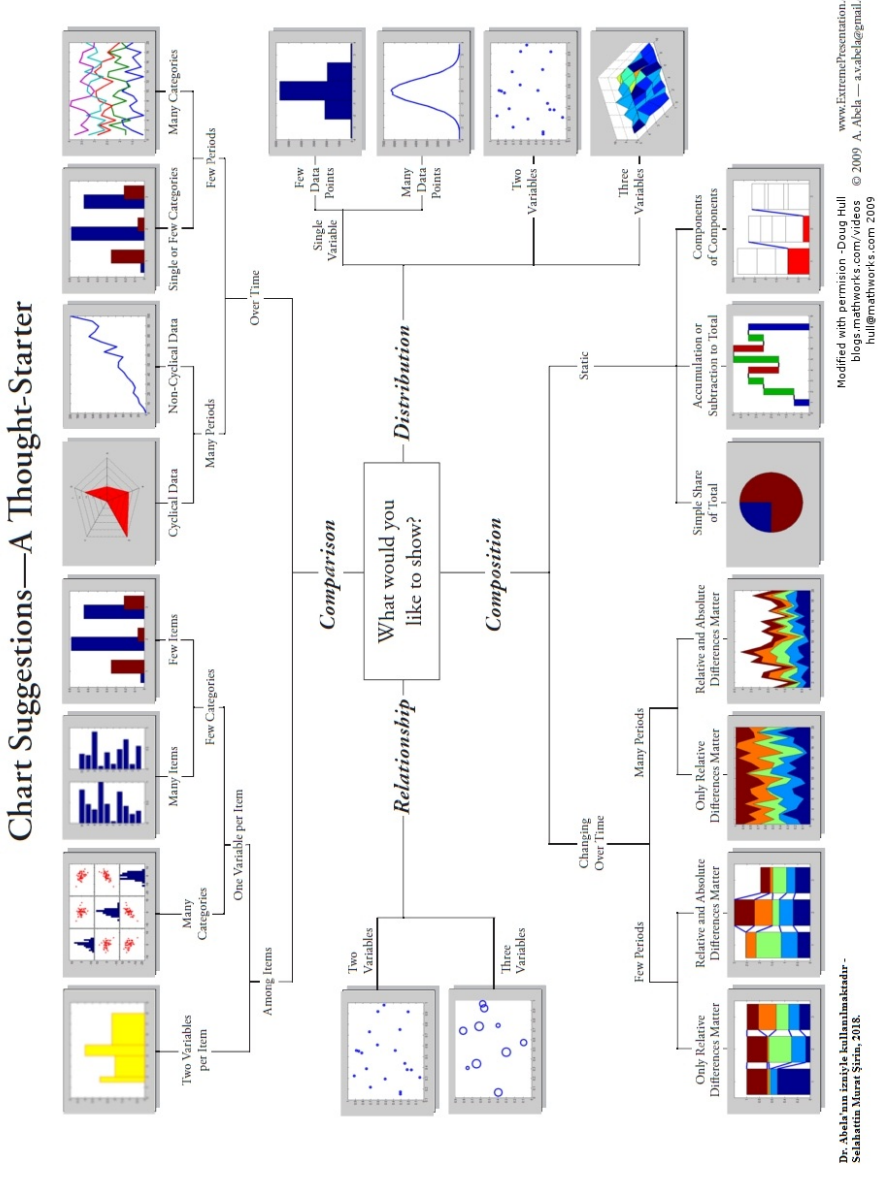
Verilerin değerlendirilmesi için grafikler hazırlanırken öncelikle değişken sayısı ve açıklanmak istenilen ilişkiye göre grafiğin seçilmesi, ayrıca değişkenlerin (1) Merkezi Dağılımı , (2) Genel Dağılımı ve (3) Dağılımın şeklini gösterecek şekilde grafiklerin hazırlanması gerekmektedir. Şekil 4.1 ile Dr. Abela tarafından hazırlanan ve The Extreme Presentation Method adresinden alınabilecek tabloda değişkenler arasında bakılmak istenilen ilişki (relationship), karşılaştırma (comparison), dağılım (distribution) ve yapıya (composition) göre kullanılabilecek grafik türleri gösterilmektedir. Ayrıca Juice Analytics tarafından hazırlanan Chart Chooser internet adresinde etkileşimli olarak grafik seçilebilmektedir.

Analiz için ilk başvurulan grafiklerden sıklık dağılımları yani histogramlar değişken hakkında önemli bilgiler vemektedir. Histogram, N sayıda veri K sınıfa (bins) bölünerek oluşturulur. Oluşturulan sınıflar aynı genişlikte olmalı yani aralıklar aynı olmalıdır. Sınıfların aralıkları $X_{max} - X_{min}/k)$ olarak belirlenir. Eğer küsuratlı bir sayı çıkarsa en yakındaki tam sayıya yuvarlanır. Histograma benzer diğer bir grafik de **ggplot2** paketi ile sunulan `geom_freqploy()` fonksiyonu ile hazırlanabilir. Histogramdan farklı olarak bu fonksiyonda sıklık dağılımları çubuk yerine çizgilerle gösterilmektedir.

Histogramlarla birlikte yaygın olarak kullanılan ikinci grafik *Box-Whisker Plot* yani kutu grafikleridir. Bu grafikler en düşük ve en yüksek değerlere ek olarak çeyrek (kartil-quartile) değerlerini de göstererek veri setinin hem dağılımı hem de uç değerleri hakkında bilgi sunar.

Bu yöntemler az sayıda değişken için kullanışlı olsa da değişken sayısı arttıkça her bir değişkenin incelenmesi zaman ve kaynak israfına yol açmaktadır. Bu nedenle veri setlerinin kısa bir sürede incelenmesine imkan tanıyacak fonksiyonlardan faydalanılmaktadır. Bu fonksiyonlardan ilki **tabplot** paketi ile sunulan ve değişkenlere ilişkin özet bilgileri sunan `tableplot()` fonksiyonudur (Tennekes ve Jonge 2017). Bu fonksiyonda *dat* öğesi ile kullanılacak veri seti, *select* ile kullanılacak değişkenler, *subset* ile varsa seçilecek satırlar, *sortCol* ile verilerin hangi sütuna göre sıralanacağı, *scales* ile eksen ölçekleri (mevcut seçenek veriye göre otomatik seçer), *pals* ile kullanılacak renk paleti ("Set1", "Set2", "Set3", "Set4", "Set5", "Set6", "Set7", "Set8", "Paired", "HCL1", "HCL2", ve "HCL3") ve diğer özellikler tanımlanabilir.

```
require(tabplot)
# tableplot fonksiyonu
tableplot(dat, select, subset = NULL, sortCol = 1, decreasing = TRUE,
  nBins = 100, from = 0, to = 100, nCols = ncol(dat), sample = FALSE,
  sampleBinSize = 1000, scales = "auto", numMode = "mb-sdb-ml",
  max_levels = 50, pals = list("Set1", "Set2", "Set3", "Set4"),
  change_palette_type_at = 20, rev_legend = FALSE, colorNA = "#FF1414",
  colorNA_num = "gray75", numPals = "OrBu", limitsX = NULL,
```



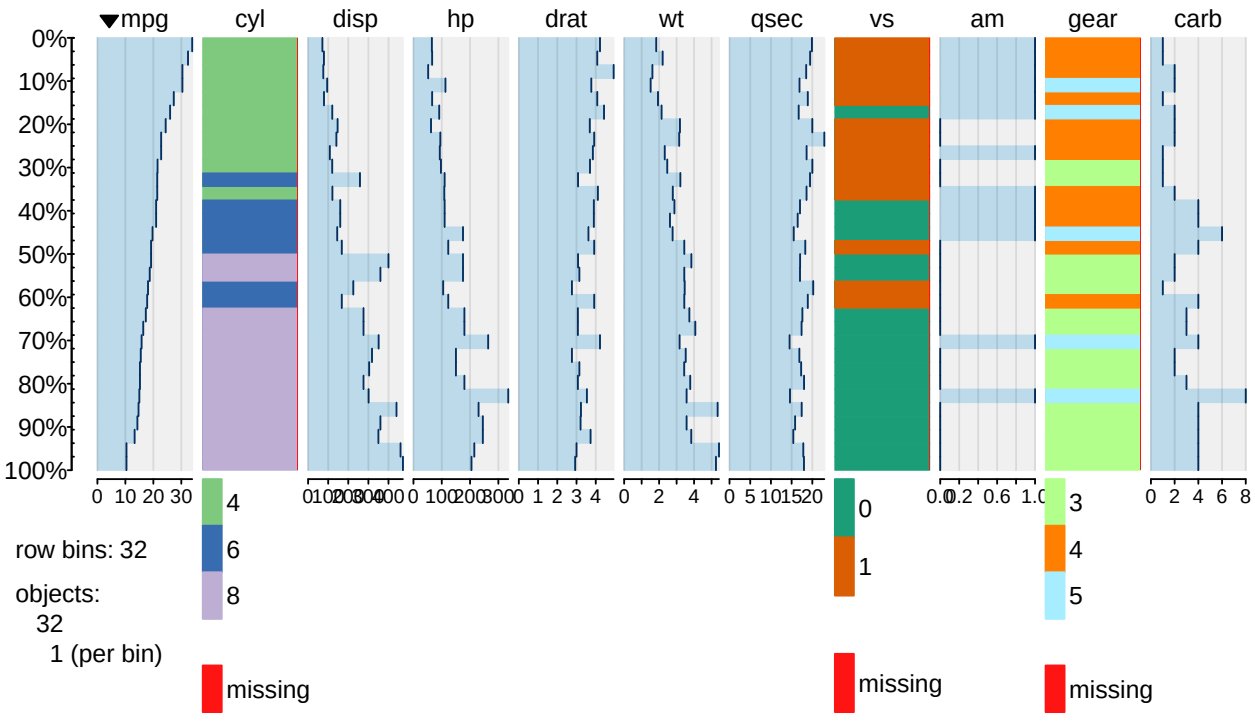
Şekil 4.1: Açıklayıcı veri analizi grafikleri

```
  bias_brokenX = 0.8, IQR_bias = 5, select_string = NULL,
  subset_string = NULL, colNames = NULL, filter = NULL, plot = TRUE,
  ...)
```

Bu fonksiyon kullanılırken dikkat edilmesi gereken bir husus, kategorik değişkenler (factors) farklı renklerde kullanılabileceğinden veri setinde "numeric" sınıfı değişkenler var ise bunların öncelikle kategorik hale getirilmesi gerekmektedir. Ayrıca kırmızı renk eksik verilerin gösterilmesi için mevcut renk olarak belirlendiğinden renklendirmede kırmızı haricindeki renkler seçilmelidir. Bu fonksiyon hakkında daha fazla bilgi için Tabplot Vignette internet sitesine bakılabilir.

Şekil 4.2 ile `tableplot()` fonksiyonu kullanılarak `mtcars` veri setinin grafikle sunumu gösterilmektedir. Grafikten görüldüğü üzere vites sayısı, silindir sayısı ve VS değerleri kategorik hale getirilmiş ve bunlar üç farklı renk paleti ile gösterilmiştir.

```
require(tabplot)
require(tidyr)
data(mtcars)
mtcars$gear <- as.factor(mtcars$gear)
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$vs <- as.factor(mtcars$vs)
tableplot(mtcars, pals = list(vs = "Set3", cyl = "Set4", gear = "Set7"))
```



Şekil 4.2: Tableplot fonksiyonu ile mtcars veri seti grafiği

tabplotd3 paketi ise verilerin sunumu için `tableplot()` fonksiyonu ile aynı şekilde hazırlanan

ama internet tarayıcısında kullanılan interaktif `itabplot()` fonksiyonu sunmaktadır (Jonge ve Tennekes 2016). Bu fonksiyonda X öğesinde belirtilen veri seti internet tarayıcısında (Chrome, Firefox vb.) interaktif olarak açılmakta yani yakınlaştırma, uzaklaştırma ve sıralama yapılarak değişkenler incelenebilmektedir. Ayrıca `tabplot()` fonksiyonunda kullanılan parametreler burada da kullanılabilmektedir.

```
require(tabplotd3)
 itabplot(x, ...)
 # Örnek
 itabplot(mtcars)
```

tabplot paketi verilerin özet olarak sunulması için güzel grafikler hazırlamakla birlikte aralarındaki ilişkileri göstermek için kısıtlı fonksiyonlar sunmaktadır. Bu nedenle ilişkilerin gösterilmesi için **qgraph** paketinden de faydalanılabilir (Epskamp vd. 2018).

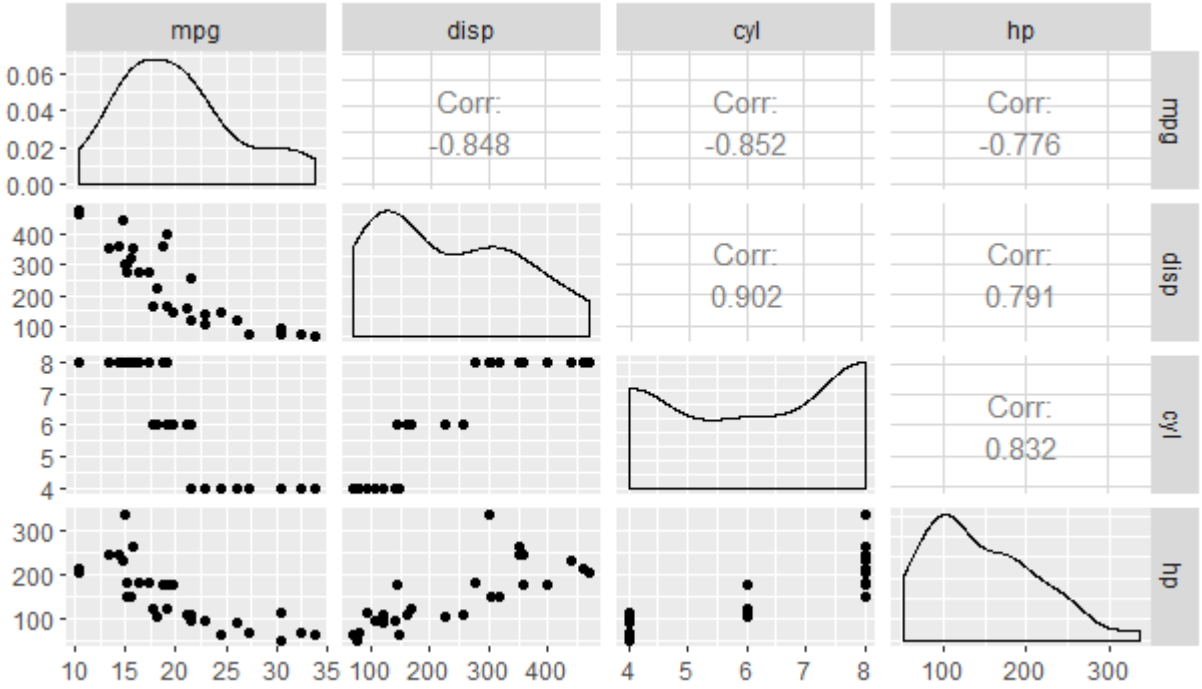
Veri setindeki değişkenlerin genel bir resmi çekildikten sonra sonra ikinci aşama değişkenlerin birbirleri arasındaki ilişkiyi gösteren dağılım grafiklerine (scatter plot) bakılması gerekir. Bu grafikler değişkenler arasındaki ilişkiyi (doğrusal, doğrusal olmayan), ilişkinin yönü (pozitif, negatif veya ilişkisiz) ve ilişkinin kuvveti (zayıf, güçlü) hakkında ilk bilgileri sunar. Açıklayıcı veri analizinde değişkenlerin dağılımı ve değişkenler arası korelasyonu göstermek için ilk olarak **GGally** paketinde yer alan `ggpairs()` fonksiyonu kullanılabilir. Bu fonksiyon için detaylı bilgiye GGally-Extension to ggplot bağlantısından ulaşılabilir.

```
# ggpairs fonksiyonu
ggpairs(data, mapping = NULL, columns = 1:ncol(data), title = NULL,
  upper = list(continuous = "cor", combo = "box_no_facet", discrete =
  "facetbar", na = "na"), lower = list(continuous = "points", combo =
  "facethist", discrete = "facetbar", na = "na"), diag = list(continuous =
  "densityDiag", discrete = "barDiag", na = "naDiag"), params = NULL, ...,
  xlab = NULL, ylab = NULL, axisLabels = c("show", "internal", "none"),
  columnLabels = colnames(data[columns]), labeller = "label_value",
  showStrips = NULL, legend = NULL, cardinality_threshold = 15,
  legends = stop("deprecated"))
```

Bu fonksiyon ile değişkenler arasındaki korelasyon ve değişkenlerin dağılımı çok kolay bir şekilde grafiğe aktarılmaktadır. Ayrıca değişkenler arasındaki ilişki için grafik seçimi de yapılabilmektedir. Şekil 4.3 `ggpairs()` fonksiyonu kullanılarak hazırlanan `mtcars` veri setindeki dört değişkenin grafiğini göstermektedir. Söz konusu grafikte sağ üst köşe değişkenler arasındaki korelasyonu, sol alt köşe değişkenlerin dağılım grafiğini ve köşegenlerde değişkenlerin dağılımını göstermektedir.

```
# mtcars veri setinin ggpairs fonksiyonu ile incelenmesi
require(dplyr)
require(ggplot2)
require(GGally)
data(mtcars)
mtcars %>%
```

```
  select(mpg, disp, cyl, hp) %>%
    ggpairs(.)
```



Şekil 4.3: ggpairs ile veri setinin incelenmesi

Söz konusu fonksiyona ek olarak **ggpubr** ve **gcookbook** paketleri de yayın kalitesinde açıklayıcı veri analizinde kullanılacak grafik örnekleri sunmaktadır. Bunun için RPubs-Exploring Data with R internet sitesine bakılabilir.

Ayrıca korelasyon matrisinin gösterilmesi için **corrplot** paketinde yer alan ve koreleasyon grafiklerinin çok detaylı olarak hazırlanmasına imkan tanıyan `corrplot()` fonksiyonu yaygın olarak kullanılmaktadır(Wei ve Simko 2017). Fonksiyonda *corr* grafik için kullanılacak matrisi, *method* kullanılacak görseli (number olduğunda korelasyon rakam olarak gösterilir), *type* matrisin yarım veya tam olarak hangi türde gösterileceğini, *order* söz konusu matriste bazı ilişkilerin ortaya çıkartılması için istenilirse kullanılacak yöntemi (FPC: Birinci temel birleşen sıralaması, hclust: Hiyerarşik Kümelenme, alphabet: İsim sırası) belirlemek için kullanılır. *is.corr* seçeneği "FALSE" olarak belirtilirse değişkenlerin korelasyonu değil veri setinde yer alan sayısal özellikler gösterilir. Bu seçenek özellikle mesafe değerlerinin (distance metrics) gösterilmesinde oldukça kullanışlıdır. `corrplot.mixed()` fonksiyonu ise birden fazla korelasyon grafiğinin bir arada gösterilmesi için kullanılır. Her iki fonksiyonun kapsamlı açıklamasına yardım dosyalarına bakılabilir.

```
# corrplot fonksiyonu
corrplot(corr,
         method = c("circle", "square", "ellipse", "number", "shade",
                    "color", "pie"),
```

```
  type = c("full", "lower", "upper"), add = FALSE,
  col = NULL, bg = "white", title = "",is.corr = TRUE, diag = TRUE,
  outline = FALSE, mar = c(0, 0, 0, 0), addgrid.col = NULL,
  addCoef.col = NULL, addCoefasPercent = FALSE,
  order = c("original", "AOE", "FPC", "hclust", "alphabet"),
  hclust.method = c("complete", "ward",
                    "ward.D", "ward.D2", "single", "average",
                    "mcquitty", "median", "centroid"),...)

# corrplot.mixed fonksiyonu
corrplot.mixed(corr, lower = "number", upper = "circle",
               tl.pos = c("d","lt", "n"), diag = c("n", "l", "u"),
               bg = "white", addgrid.col = "grey",
               plotCI = c("n", "square", "circle", "rect"), ...)
```

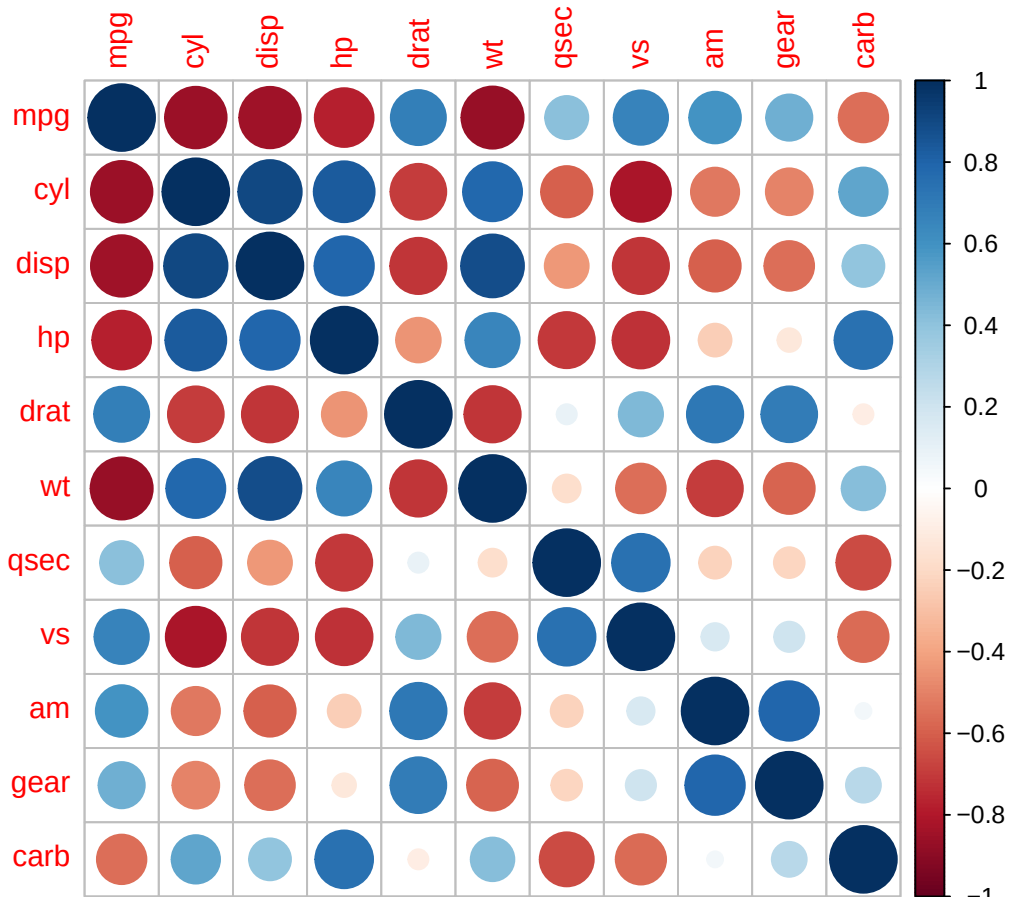
Şekil 4.4 `mtcars` veri setinde yer alan değişkenlerin korelasyonunu renkli olarak göstermektedir.

```
require(ggplot2)
require(gcookbook)
require(corrplot)
data(mtcars)
mcor <- cor(mtcars)
corrplot(mcor)
```

`corrplot()` fonksiyonu az sayıda değişkenin açıklanması için yeterli iken değişken sayısı arttıkça görsel olarak karışıklığa neden olabilir. Bu sorunun önüne geçmenin bir yolu değişkenler arası ağ grafiği oluşturarak değişkenlerin ilişkilerinin görsel hale getirilmesidir. Bunun için `cor()` fonksiyonu ile oluşturulan korelasyon matrisi **qgraph** paketinde yer alan `qgraph()` fonksiyonu ile değişkenler arasındaki ilişkiyi gösteren bir ağ grafiği haline getirilebilir. Söz konusu fonksiyonda *input* korelasyon nesnesini (`cor()` fonksiyonu nesnesini), *posCol* pozitif ilişki için kullanılacak rengi, *negCol* negatif ilişki için kullanılacak rengi, *layout* grafiğin görünümü (circle:Çember, spring:dağınık), *minimum*, *maximum* ve *threshold* özellikle korelasyon için alınacak alt üst sınırlar ve eşik değerleri, *details* grafikle birlikte temel değerlerin gösterilip (TRUE) gösterilmeyeceğini (FALSE), *palette* ve *theme* ise kullanılacak renk paletini ve temayı seçmek için kullanılır. Bu fonksiyon korelasyon haricinde diğer yöntemler için kullanılan fonksiyonlardan elde edilen nesnelerin de gösterilmesi için (SEM, Lavaan, vb.) kullanılabilir.

```
# qgraph fonksiyonu
qgraph(input, posCol="",negCol="", layout, minimum, maximum,
       details= FALSE, threshold, palette,
       theme, ...)
```

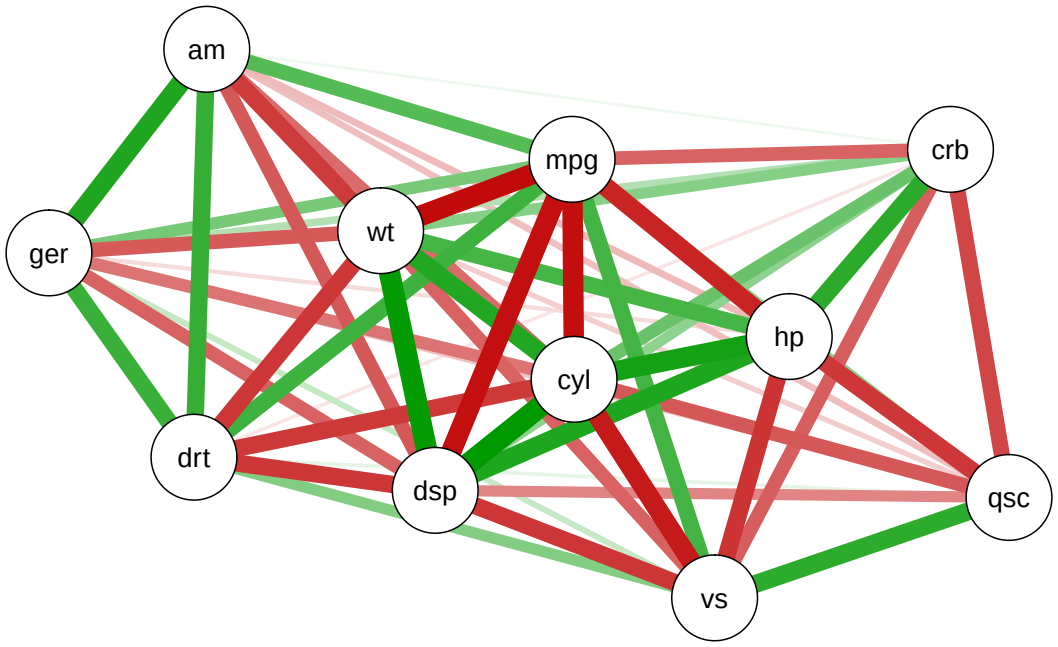
`mtcars` veri setinin `qgraph()` fonksiyonu ile hazırlanan grafik Şekil 4.5 ile gösterilmekte olup, kırmızı çizgi negatif ilişkiyi yeşil çizgi ise pozitif ilişkiyi göstermektedir.



Şekil 4.4: Korelasyon grafiği

```
require(corrplot)
require(qgraph)
data(mtcars)
mcor2 <- cor(mtcars)
qgraph(mcor2,layout = "spring")
```

Değişkenler arasındaki ilişkiyi gösteren başka bir grafik de χ grafiğidir. Bu grafik için iki değişkenin gözlemlerini kullanarak χ_i ve λ_i değerleri elde edilir. Burada χ_i değeri χ^2 değerinin kare kökü olup normal dağılımda bu değerlerin sıfır etrafında rassal olarak dağılması beklenir. λ_i değeri ise ölçümün dağılımın merkezinden mesafesini göstermektedir (Everitt ve Hothorn 2011). R'de χ grafiği için **MVA** paketindeki `chiplot()` fonksiyonu kullanılır. Bu fonksiyonda X ve Y öğeleri değişkenleri tanımlamak için kullanılır ve diğer grafik özellikleri de belirlenebilir(Everitt ve Hothorn 2015). Şekil 4.6 `mtcars` veri setinde mpg ve hp değişkenleri için hazırlanan χ grafiğini göstermektedir. Söz konusu grafikte değişkenlerin önemli bir kısmı kesikli çizgiler ile gösterilen aralığın dışında olduğundan "mpg" ve "hp" arasında negatif ters ve kuvvetli bir ilişki vardır.



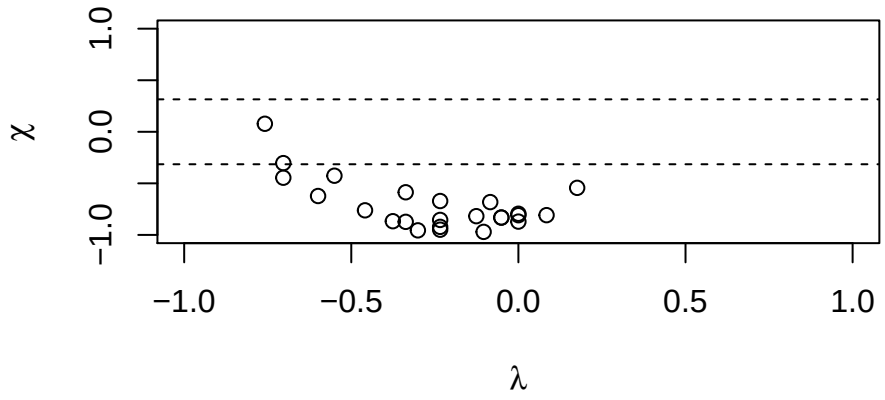
Şekil 4.5: Qgraph fonksiyonu ile mtcars veri setinin gösterimi

```
chiplot(x, y, ...)
```

```
require(MVA)
data(mtcars)
with(mtcars, chiplot(mpg,hp))
```

Değişkenler arasındaki ilişkiyi göstermek için son dönemde yaygın olarak kullanılmaya başlanılan bir başka grafik de eş yükselti (contour plot) grafikleridir. Bu grafikler haritadaki kullanımına eşdeğer olarak ilişki üç değişken arasındaki ilişkiyi iki boyutta göstererek ilişki düzeyinin aynı olduğu alanları göstermek için kullanılır. Bunun için **ggplot2** paketinde yer alan `geom_contour()` fonksiyonu ve iki değişken için `geom_density_2d()` fonksiyonu kullanılabileceği gibi **plotly** paketi de kullanılabilir. Burada söz konusu grafiklerin **ggplot2** paketinde nasıl hazırlanacağı gösterilecek olup, **plotly** paketi ile eşyükselti grafiğinin nasıl hazırlanacağına Contour Plots in R internet sitesinden bakılabilir.

Veri analizi için kullanılan başka bir grafik de *kök ve yaprak* (stem and leaf) grafikleridir. Bu grafik türünde sayısal değerler kök ve yaprak olarak ayrılır vesayıların sıklığını ve değerlerini göstermek için kullanılır. Onluk değerlerde baş kısımda onlar, "|" işaretinin sağındaki birler başamağı iken, yüzler ve daha büyük değerlerde "|" işaretinin sağında kaç sıfır olacağı belirtilir. Ayrıca sayılarda yuvarlama da yapılarak değerler gösterilir. Örnekte `mtcars` veri setindeki "hp" değişkenin kök-yaprak grafiği gösterilmektedir. Bu grafikte de görüldüğü üzere onluk değerler sıfırla gösterilmiş, ve yüzler basamağı sıfırdan farklı olanlar sıralanmıştır. İlk değerde onlar basamağı 5 olan bir adet, 6 olan 1 adet 7 olan 3 adet ve 9 olan bir adet sayı bulunmaktadır.



Şekil 4.6: Ki grafiği

Ancak "hp" değerlerine bakıldığında onlar basamağı 6 olan dört sayı varken bunlardan sadece biri 65'ten küçük olduğundan bu sayı korunmuş ve diğerleri 70'e yuvarlanmıştır.

```
data(mtcars)
stem(mtcars$hp)
```

```
##
##   The decimal point is 2 digit(s) to the right of the |
##
##   0 | 5677799
##   1 | 0011111122
##   1 | 55888888
##   2 | 123
##   2 | 556
##   3 | 4
```

Açıklayıcı veri analizinde değişkenlerin kendi arasındaki ilişkilerle birlikte uç değer olup olmadığının da incelenmesi gereklidir. Birden fazla değişkenin olduğu çok boyutlu bir durumda uç değerin tespit edilmesi için bazı yöntemlere başvurulmakla birlikte bunun görsel olarak sunulması için **OutliersO3** paketinden faydalanılabilir(Unwin 2018). Bu pakette yer alan fonksiyonlardan `O3prep()` fonksiyonu ile uç değere ilişkin grafiklerin hazırlanması sağlanılır. Fonksiyondaki *data* öğesi kullanılacak veri setini, K en çok kaç değişkenin dikkate alınacağını, *method* uç değer tespiti için kullanılacak yöntemi belirtmek için kullanılır. Bu yöntem ve algoritmalar, kendi paketleri ile de kullanılabilen "HDo", "PCS", "FastPCS", "BAC", "AdjOut", "DDC", "MCD" yöntemi olup yalnızca birisi seçilebileceği gibi birkaçı birlikte kullanılabilir.

```
require(OutliersO3)
O3prep(data, k1=1, K=ncol(data), method="HDo", ...)
```

```
method = c("HDo", "PCS", "FastPCS", "BAC", "AdjOut", "DDC", "MCD")
```

`O3prep()` fonksiyonu ile incelenecek veri hazırlandıktan sonra eğer tek bir yöntemle uç değere bakılacaksa `O3plotT()` fonksiyonu, birden fazla yöntemle uç değerlere bakılacaksa `O3plotM()` fonksiyonu ile hazırlanan nesne incelenir. Söz konusu fonksiyonlarda *outResults* hazırlanan nesneyi, *caseNames* eğer değişken isimlerinden farklı isim kullanılacaksa bu isimleri ve *o3control* ise renklendirme seçeneklerini belirtmek için kullanılır. Söz konusu fonksiyonlar neticesinde *nOut* ile kç uç değer bulunduğu gösterlirken, *gO3* ile de yöntemlerden elde edilen grafik gösterilmektedir.

```
O3plotT(outResults, caseNames=as.character(1:nrow(outResults$data)),
 O3control=O3plotColours())

O3plotM(outResults, caseNames=as.character(1:nrow(outResults$data)),
 O3control=O3plotColours())
```

Söz konusu fonksiyonlara ilişkin bir örnek için kimyasal ürünler üreten bir fabrikada amonyaktan nitrik asit elde edilirken takip edilen değişkenlere ilişkin Brownlee (1960) çalışmasında kullanılan `stackloss` veri seti kullanılacaktır.[1]

Yapılan örnekte elde edilen sonuçlar `outmulti$nOut`ile gösterilmektedir. Burada kullanılan yöntemlere göre elde edilen uç değer sayısı verilmektedir.

```
require(OutliersO3)
data("stackloss")
outdata <- O3prep(stackloss,
                  method =c("HDo", "BAC", "DDC"), tols=0.05)
outmulti <- O3plotM(outdata)
outmulti$nOut
```

```
## HDo BAC DDC
##  14   0   4
```

```
outmulti$gO3
```

Şekil 4.7 ise uç değerlerin grafiksel gösterimini sunmaktadır.

4.2 Tabloların Oluşturulması

Açıklayıcı veri analizinde ikinci aşama dağılım tablosunun (contingency table) hazırlanmasıdır. Görsel incelemeden sonra değişkenlerin sıklıklarına ilişkin bu tablonun hazırlanması için **gmodels** paketinde bulunan `CrossTable()` fonksiyonu kullanılabilir. Bu fonksiyonda x ve

[1] Brownlee, K. A. (1960, 2nd ed. 1965) Statistical Theory and Methodology in Science and Engineering. New York: Wiley. pp. 491–500.

Şekil 4.7: OutliersO3 paketi örneği

y öğeleri kullanılacak değişken vektörlerini, *digits* tablolarda noktada sonra kaç basamak alınacağını, *max.width* tablonun kaçıncı sütundan itibaren kesilerek aşağıya aktarılacağını, *prop.r* satır yüzdelerinin gösterilip gösterilmeyeceğini, *prop.c* sütun yüzdelerinin gösterilip gösterilmeyeceğini, *prop.t* satır yüzdelerin gösterilip gösterilmeyeceğini, *expected* χ^2 testi yapılarak beklenen yüzdelerin gösterilip gösterilmeyeceğini belirtir.

```
# CrossTable fonksiyonu
CrossTable(x, y, digits=3, max.width = 5, expected=FALSE,
           prop.r=TRUE, prop.c=TRUE, prop.t=TRUE, prop.chisq=TRUE,
           chisq = FALSE, fisher=FALSE, mcnemar=FALSE,
           resid=FALSE, sresid=FALSE, asresid=FALSE,
           missing.include=FALSE,
           format=c("SAS","SPSS"), dnn = NULL, ...)
```

Aşağıda yer alan örnekte `mtcars` veri setinde yer alan V/S değeri (vs) ve vites sayısı (gear) için bir dağılım tablosu hazırlanmıştır.

```
require(gmodels)
CrossTable(mtcars$vs,mtcars$gear,prop.t=TRUE, prop.r=TRUE, prop.c=TRUE,
           expected=FALSE,chisq=FALSE, format="SPSS")
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## | Chi-square contribution |
## |             Row Percent |
## |          Column Percent |
```

```
## |           Total Percent |
## |-------------------------|
##
## Total Observations in Table:  32
##
##              | mtcars$gear
##    mtcars$vs |         3  |         4  |         5  | Row Total  |
## -------------|------------|------------|------------|------------|
##            0 |        12  |         2  |         4  |        18  |
##              |     1.504  |     3.343  |     0.501  |            |
##              |    66.667% |    11.111% |    22.222% |    56.250% |
##              |    80.000% |    16.667% |    80.000% |            |
##              |    37.500% |     6.250% |    12.500% |            |
## -------------|------------|------------|------------|------------|
##            1 |         3  |        10  |         1  |        14  |
##              |     1.934  |     4.298  |     0.645  |            |
##              |    21.429% |    71.429% |     7.143% |    43.750% |
##              |    20.000% |    83.333% |    20.000% |            |
##              |     9.375% |    31.250% |     3.125% |            |
## -------------|------------|------------|------------|------------|
## Column Total |        15  |        12  |         5  |        32  |
##              |    46.875% |    37.500% |    15.625% |            |
## -------------|------------|------------|------------|------------|
##
##
```

Açıklayıcı veri analizinde yaygın bir şekilde kullanılan tablolardan birisi de "pivot" tablolarıdır. Excel kullanıcıları tarafından sıklıkla başvurulan pivot tablolar için maalesef R'de fazla bir seçenek bulunmamaktadır. **rpivotTable** paketinde yer alan `rpivotTable()` fonksiyonu ile dinamik (js) pivot tablosu oluşturmak mümkünken **dplyr** paketinde yer alan fonksiyonlar da pivot tablosuna benzer tablonun oluşturulmasını sağlar.

```
# rpivotTable fonksiyonu
rpivotTable(data, rows = NULL, cols = NULL, aggregatorName = NULL,
  vals = NULL, rendererName = NULL, sorter = NULL, exclusions = NULL,
  inclusions = NULL, ..., width = NULL, height = NULL)
```

Her ne kadar pivot tablo için doğrudan kullanılacak bir fonksiyon olmasa da **dplyr** ve **tidyr** paketi kullanılarak pivot tabloya benzeyen tablolar oluşturulabilir. Excel benzeri pivot tablo oluşturmak için örnek olarak Excel örneklerinde de kullanılan ülkelerin meyve-sebze ithalatına ilişkin tablo kullanılacaktır. Söz konusu tablo kitabın Github sayfasından indirilebileceği gibi Easy-excel.com adresinden de indirilebilir.

```
require(readr)
pivottablo <-
  read_delim("https://raw.githubusercontent.com/RiUA/riua/master/pivottablo.csv",
```

```
            ";", escape_double = FALSE,
            col_types = cols(Deger = col_number(),
                             Kategori = col_character(),
                             Tarih = col_date(format = "%d.%m.%Y")),
                             trim_ws = TRUE)
```

Örnekte ürün cinsinden toplam ithalat değerleri Tablo 4.1 ve ürün kategorisine göre ülke bazında toplam ithalat değerleri Tablo 4.2 ile gösterilmiştir. Son olarak oluşturulan tablolar Excel sayfası gibi geniş hali ile yazılacak olunursa Tablo 4.3 şeklinde gösterilebilir.

```
require(dplyr)
require(knitr)
urun <-  pivottablo %>%
  group_by(Urun) %>%
  summarise(Toplam_Deger = sum(Deger))
```

	Urun	Toplam_Deger
1	Apple	191257.00
2	Banana	340295.00
3	Beans	57281.00
4	Broccoli	142439.00
5	Carrots	136945.00
6	Mango	57079.00
7	Orange	104438.00

Tablo 4.1: Ürün Toplam İthalat Değeri

```
require(dplyr)
require(knitr)
require(xtable)
urun2 <-  pivottablo %>%
  group_by(Ulke, Kategori) %>%
  summarise(Toplam_Deger = sum(Deger))
```

```
require(tidyr)
urun3 <- spread(urun2,Kategori, Toplam_Deger)
```

Bunlar haricinde özellikle anketler için `crosstab()` fonksiyonu kullanılabilir. Bu fonksiyon hakkında detaylı uygulamalara Crosstab - a cross-tabulation function for use with survey data internet adresinden erişilebilir.

4.3 Açıklayıcı Veri Analizi için Paketler

	Ulke	Kategori	Toplam_Deger
1	Australia	Fruit	91221.00
2	Australia	Vegetables	40492.00
3	Canada	Fruit	82338.00
4	Canada	Vegetables	12407.00
5	France	Fruit	125931.00
6	France	Vegetables	15125.00
7	Germany	Fruit	66430.00
8	Germany	Vegetables	88738.00
9	New Zealand	Fruit	62392.00
10	New Zealand	Vegetables	4390.00
11	United Kingdom	Fruit	87786.00
12	United Kingdom	Vegetables	85351.00
13	United States	Fruit	176971.00
14	United States	Vegetables	90162.00

Tablo 4.2: Ürün Kategorisine Göre Toplam İthalat Değerleri

	Ulke	Fruit	Vegetables
1	Australia	91221.00	40492.00
2	Canada	82338.00	12407.00
3	France	125931.00	15125.00
4	Germany	66430.00	88738.00
5	New Zealand	62392.00	4390.00
6	United Kingdom	87786.00	85351.00
7	United States	176971.00	90162.00

Tablo 4.3: Ürün Kategorisine Göre Toplam İthalat Değerleri

Bir önceki bölümde ve bu bölümde belirtilen uygulamalar için görüldüğü üzere çok farklı paketler ve fonksiyonlar kullanılmaktadır. Özellikle yeni başlayanlar için hangi fonksiyonu veya hangi paketı kullanacağına karar vermek önemli bir zaman kaybına yol açabilir. Bunun önüne geçmek için geliştirilen bazı paketler ile Açıklayıcı Veri Analizi için gereken analiz ve grafik hazırlama işlemler çok kolaylıkla yapılabilmektedir.

Bu paketlerden ilki **DescTools** paketi olup bu pakette istatistiki analiz için kullanılan 440 civarında fonksiyon sunulmaktadır (Signorell 2018). Bu fonksiyonlardan en yaygın kullanılanlardan ilki `Desc()` fonskiyonudur. Bu fonksiyon verinin tüm istatistiki özelliklerini ve dağılım grafiğini sunmaktadır. Bu fonksiyonda x incelenecek veri setini, *plotit* grafiğin çizilip çizilmeyeceğini, *main* verinin açıklanmasını, *wrd* ise MS Word uygulamasına aktarılıp aktarılmayacağını belirtmek için kullanılır.

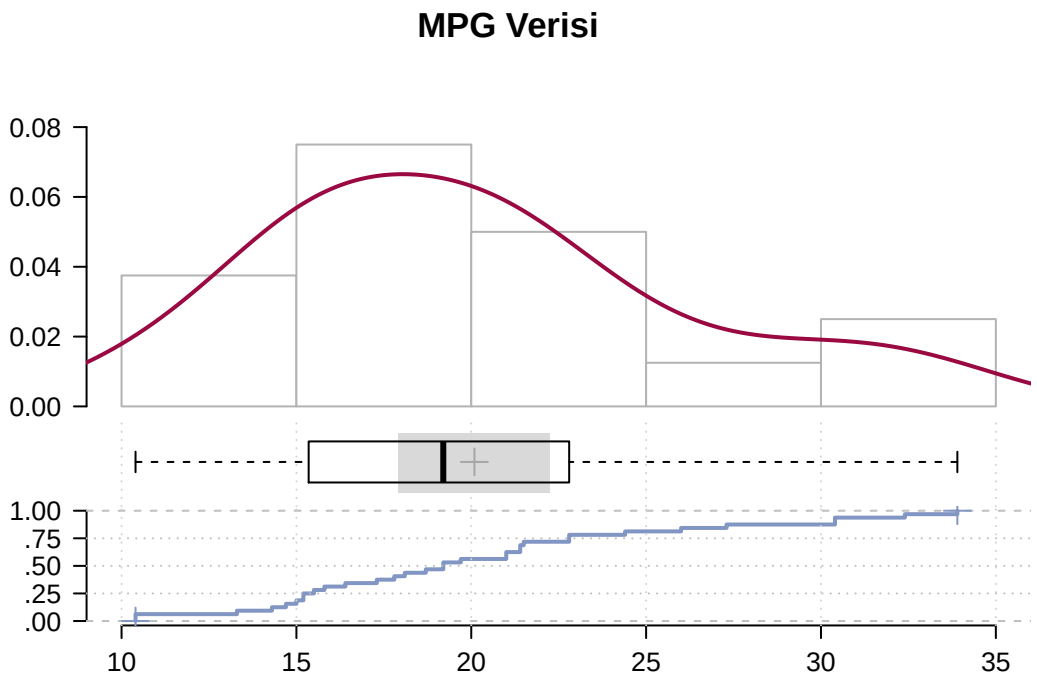
```
require(DescTools)
Desc(x, ..., main = NULL, plotit = NULL, wrd = NULL)
```

Örnek olarak `mtcars` veri setinde aracın yakıt tüketimi (mpg) bu fonksiyon ile incelendiğinde

Şekil 4.8 ile göserilen değişkenin dağılım grafiği elde edilmekte ve değişkene ilişkin şu sonuçlar verilmektedir:

```
require(DescTools)
data(mtcars)
Desc(mtcars$mpg, plotit = TRUE, main= "MPG Verisi")
```

```
## ------------------------------------------------------------------------
## MPG Verisi
##
##   length       n     NAs  unique      0s    mean  meanCI
##       32      32       0      25       0  20.091  17.918
##           100.0%    0.0%            0.0%          22.264
##
##      .05     .10     .25  median     .75     .90     .95
##   11.995  14.340  15.425  19.200  22.800  30.090  31.300
##
##    range      sd   vcoef     mad     IQR    skew    kurt
##   23.500   6.027   0.300   5.411   7.375   0.611  -0.373
##
## lowest : 10.4 (2), 13.3, 14.3, 14.7, 15.0
## highest: 26.0, 27.3, 30.4 (2), 32.4, 33.9
```



Şekil 4.8: DescTools grafiği

Kategorik veya diğer sınıflar için de benzer özellikler sunulmaktadır. Mesela *gear* değişkeni kategorik olarak değerlendirilirse, değişkenin kategori sayısı, gözlem sıklığı ve oranı verilecektir.

```
Desc(as.factor(mtcars$gear), plotit=FALSE, main ="Gear Değişkeni")
```

```
## ------------------------------------------------------------------------
## Gear Değişkeni
##
##   length      n    NAs unique levels  dupes
##       32     32      0      3      3      y
##          100.0%   0.0%
##
##    level  freq   perc  cumfreq  cumperc
## 1      3    15  46.9%       15    46.9%
## 2      4    12  37.5%       27    84.4%
## 3      5     5  15.6%       32   100.0%
```

Değişkenlerin tek tek incelenmesinin yanı sıra ikili veya gruplar halinde incelenmesi de bu fonksiyonla mümkün olmaktadır. Mesela *gear* ve *mpg* değişkenlerinin ilişkisine bakılmak istenildiğinde kategori bazında istatistiki değerler ve kartil değeleri verilmektedir.

```
require(DescTools)
Desc(as.factor(gear)~mpg, data = mtcars, plotit = FALSE,
     main = "Gear ve Mpg Verisi")
```

```
## ------------------------------------------------------------------------
## Gear ve Mpg Verisi
##
## Summary:
## n pairs: 32, valid: 32 (100.0%), missings: 0 (0.0%), groups: 3
##
##
##                3        4        5
## mean      16.107   24.533   21.380
## median    15.500   22.800   19.700
## sd         3.372    5.277    6.659
## IQR        3.900    7.075   10.200
## n             15       12        5
## np       46.875%  37.500%  15.625%
## NAs            0        0        0
## 0s             0        0        0
##
## Kruskal-Wallis rank sum test:
##   Kruskal-Wallis chi-squared = 14.323, df = 2, p-value = 0.0007758
##
##
## Proportions of as.factor(gear) in the quantiles of mpg:
```

```
##
##          Q1      Q2      Q3      Q4
##   3   87.5%   66.7%   25.0%    0.0%
##   4    0.0%   22.2%   62.5%   71.4%
##   5   12.5%   11.1%   12.5%   28.6%
```

Açıklayıcı Veri Analizi için kullanılabilecek diğer bir paket **DataExplorer** paketidir (Cui 2018). Bu pakette sunulan en önemli fonksiyon `create_report()` fonksiyonu olup, bu fonksiyon ile temel grafikler ve analizler bir .html dosyası olarak rapor olarak sunulmaktadır. Söz konusu raporda veri setindeki değişkenlerin özellikleri, veri setinin ağ grafiği, kayıp değerleri, verinin dağılımı ve değişkenler arasındaki korelasyon sunulmaktadır. Bu fonksiyonda *data* incelenecek veri setini, *output_file* elde edilecek çıktı dosyasının ismini, *output_dir* ise raporun hangi klasöre yükleneceğini belirtmek için kullanılır.

```
require(DataExplorer)
create_report(data, output_file = "report.html",
              output_dir = getwd(), ...
```

Eğer verilere ilişkin detaylı bir rapor hazırlanması gerekiyorsa **dataMaid** paketi ile sunulan `summarize()`, `makeDataReport()` ve `makeCodebook()` fonksiyonları da kullanılabilir (Petersen ve Ekstrøm 2018). Bu fonksiyonlardan ilki değişkenin temel özelliklerini gösterir. Bu fonksiyondaki *v* öğesi incelenecek vektörü veya veri setini, *reportstyleOutput* öğesi sonuçların matris olak gösterilip gösterilmeyeceğini ve *summaries* öğesi ise hangi bilgilerin sunulacağını belirtmek için kullanılır.

```
summarize(v, reportstyleOutput = FALSE, summaries = setSummaries(), ...)
```

Örnek olarak *mtcars* veri setindeki mpg değişkenine bakılacak olunursa, foksiyon şu şekilde kullanılır:

```
data(mtcars)
dataMaid::summarize(mtcars["mpg"], reportstyleOutput = TRUE)
```

```
## $mpg
##      Feature                   Result
## [1,] "Variable type"           "numeric"
## [2,] "Number of missing obs."  "0 (0 %)"
## [3,] "Number of unique values" "25"
## [4,] "Median"                  "19.2"
## [5,] "1st and 3rd quartiles"   "15.43; 22.8"
## [6,] "Min. and max."           "10.4; 33.9"
```

`makeDataReport()` ve `makeCodebook()` fonksiyonları ise veri setine ilişkin detaylı bilgileri Rmarkdown dosyası olarak hazırlar ve çıktı olarak ".pdf" uzantılı bir dosya oluşturur. Bu dosyada verilerin özellikleri ve dağılımına ilişkin bir histogram grafiği bulunmaktadır. Bu fonksiyonda *reportTitle* öğesi ile elde edilen raporun başlığı belirtilebilir.

```
makeCodebook(data, vol = "", reportTitle = NULL, ...)
```

Ekler

Ek-1 LATEX Kılavuzu

LaTeX 2ε Cheat Sheet

Document classes

`book` Default is two-sided.
`report` No `\part` divisions.
`article` No `\part` or `\chapter` divisions.
`letter` Letter (?).
`slides` Large sans-serif font.
Used at the very beginning of a document: `\documentclass{`*class*`}`. Use `\begin{document}` to start contents and `\end{document}` to end the document.

Common `documentclass` options

`10pt/11pt/12pt` Font size.
`letterpaper/a4paper` Paper size.
`twocolumn` Use two columns.
`twoside` Set margins for two-sided.
`landscape` Landscape orientation. Must use `dvips -t landscape`.
`draft` Double-space lines.
Usage: `\documentclass[`*opt,opt*`]{`*class*`}`.

Packages

`fullpage` Use 1 inch margins.
`anysize` Set margins: `\marginsize{`*l*`}{`*r*`}{`*t*`}{`*b*`}`.
`multicol` Use *n* columns: `\begin{multicols}{`*n*`}`.
`latexsym` Use LaTeX symbol font.
`graphicx` Show image: `\includegraphics[width=`*x*`]{`*file*`}`.
`url` Insert URL: `\url{`*http://...*`}`.
Use before `\begin{document}`. Usage: `\usepackage{`*package*`}`

Title

`\author{`*text*`}` Author of document.
`\title{`*text*`}` Title of document.
`\date{`*text*`}` Date.
These commands go before `\begin{document}`. The declaration `\maketitle` goes at the top of the document.

Miscellaneous

`\pagestyle{empty}` Empty header, footer and no page numbers.
`\tableofcontents` Add a table of contents here.

Document structure

`\part{`*title*`}`
`\chapter{`*title*`}`
`\section{`*title*`}`
`\subsection{`*title*`}`
`\subsubsection{`*title*`}`
`\paragraph{`*title*`}`
`\subparagraph{`*title*`}`
Use `\setcounter{secnumdepth}{`*x*`}` suppresses heading numbers of depth $> x$, where `chapter` has depth 0. Use a `*`, as in `\section*{`*title*`}`, to not number a particular item—these items will also not appear in the table of contents.

Text environments

`\begin{comment}` Comment (not printed). Requires `verbatim` package.
`\begin{quote}` Indented quotation block.
`\begin{quotation}` Like `quote` with indented paragraphs.
`\begin{verse}` Quotation block for verse.

Lists

`\begin{enumerate}` Numbered list.
`\begin{itemize}` Bulleted list.
`\begin{description}` Description list.
`\item` *text* Add an item.
`\item[`*x*`]` *text* Use *x* instead of normal bullet or number. Required for descriptions.

References

`\label{`*marker*`}` Set a marker for cross-reference, often of the form `\label{sec:item}`.
`\ref{`*marker*`}` Give section/body number of marker.
`\pageref{`*marker*`}` Give page number of marker.
`\footnote{`*text*`}` Print footnote at bottom of page.

Floating bodies

`\begin{table}[`*place*`]` Add numbered table.
`\begin{figure}[`*place*`]` Add numbered figure.
`\begin{equation}[`*place*`]` Add numbered equation.
`\caption{`*text*`}` Caption for the body.
The *place* is a list valid placements for the body. `t`=top, `h`=here, `b`=bottom, `p`=separate page, `!`=place even if ugly. Captions and label markers should be within the environment.

Text properties

Font face

Command	*Declaration*	*Effect*
`\textrm{`*text*`}`	`{\rmfamily` *text*`}`	Roman family
`\textsf{`*text*`}`	`{\sffamily` *text*`}`	Sans serif family
`\texttt{`*text*`}`	`{\ttfamily` *text*`}`	`Typewriter family`
`\textmd{`*text*`}`	`{\mdseries` *text*`}`	Medium series
`\textbf{`*text*`}`	`{\bfseries` *text*`}`	**Bold series**
`\textup{`*text*`}`	`{\upshape` *text*`}`	Upright shape
`\textit{`*text*`}`	`{\itshape` *text*`}`	*Italic shape*
`\textsl{`*text*`}`	`{\slshape` *text*`}`	*Slanted shape*
`\textsc{`*text*`}`	`{\scshape` *text*`}`	Small Caps shape
`\emph{`*text*`}`	`{\em` *text*`}`	*Emphasized*
`\textnormal{`*text*`}`	`{\normalfont` *text*`}`	Document font
`\underline{`*text*`}`		Underline

The command (*ttt*) form handles spacing better than the declaration (*t*`\`*tt*) form.

Font size

`\tiny` tiny
`\scriptsize` scriptsize
`\footnotesize` footnotesize
`\small` small
`\normalsize` normalsize
`\large` large
`\Large` Large
`\LARGE` LARGE
`\huge` huge
`\Huge` Huge
These are declarations and should be used in the form `{\small ...}`, or without braces to affect the entire document.

Verbatim text

`\begin{verbatim}` Verbatim environment.
`\begin{verbatim*}` Spaces are shown as ␣.
`\verb!text!` Text between the delimiting characters (in this case ‘!’) is verbatim.

Justification

Environment	*Declaration*
`\begin{center}`	`\centering`
`\begin{flushleft}`	`\raggedright`
`\begin{flushright}`	`\raggedleft`

Miscellaneous

`\linespread{`*x*`}` changes the line spacing by the multiplier *x*.

Text-mode symbols

Symbols

`&` `\&` `_` `\_` `...` `\ldots` `•` `\textbullet`
`$` `\$` `^` `\^{}` `|` `\textbar` `\` `\textbackslash`
`%` `\%` `~` `\~{}` `#` `\#` `§` `\S`

Accents

`ò` `\'o`	`ó` `\'o`	`ô` `\^o`	`õ` `\~o`	`ō` `\=o`
`ȯ` `\.o`	`ö` `\"o`	`ǫ` `\c o`	`ŏ` `\v o`	`ő` `\H o`
`ç` `\c c`	`ọ` `\d o`	`o͡o` `\b o`	`o͡o` `\t oo`	`œ` `\oe`
`Œ` `\OE`	`æ` `\ae`	`Æ` `\AE`	`å` `\aa`	`Å` `\AA`
`ø` `\o`	`Ø` `\O`	`ł` `\l`	`Ł` `\L`	`ı` `\i`
`ȷ` `\j`	`¡` `!‘`	`¿` `?‘`		

Delimiters

`‘` `‘` `“` `‘‘` `{` `\{` `[` `[` `(` `(` `<` `\textless`
`’` `’` `”` `’’` `}` `\}` `]` `]` `)` `)` `>` `\textgreater`

Dashes

Name	*Source*	*Example*	*Usage*
hyphen	-	X-ray	In words.
en-dash	--	1–5	Between numbers.
em-dash	---	Yes—or no?	Punctuation.

Line and page breaks

`\\` Begin new line without new paragraph.
`\\*` Prohibit pagebreak after linebreak.
`\kill` Don’t print current line.
`\pagebreak` Start new page.
`\noindent` Do not indent current line.

Miscellaneous

`\today` March 28, 2017.
`$\sim$` Prints ∼ instead of `\~{}`, which makes ˜.
`~` Space, disallow linebreak (`W.J.~Clinton`).
`\@.` Indicate that the `.` ends a sentence when following an uppercase letter.
`\hspace{`*l*`}` Horizontal space of length *l* (Ex: $l = 20$pt).
`\vspace{`*l*`}` Vertical space of length *l*.
`\rule{`*w*`}{`*h*`}` Line of width *w* and height *h*.

Tabular environments

`tabbing` environment

`\=` Set tab stop. `\>` Go to tab stop.
Tab stops can be set on “invisible” lines with `\kill` at the end of the line. Normally `\\` is used to separate lines.

Şekil 4.9: LATEX Kılavuzu-1

`tabular` environment

```
\begin{array}[pos]{cols}
\begin{tabular}[pos]{cols}
\begin{tabular*}{width}[pos]{cols}
```

`tabular` column specification

`l`	Left-justified column.
`c`	Centered column.
`r`	Right-justified column.
`p{`*width*`}`	Same as `\parbox[t]{`*width*`}`.
`@{`*decl*`}`	Insert *decl* instead of inter-column space.
`|`	Inserts a vertical line between columns.

`tabular` elements

`\hline`	Horizontal line between rows.
`\cline{`x-y`}`	Horizontal line across columns x through y.
`\multicolumn{`n`}{`*cols*`}{`*text*`}`	A cell that spans n columns, with *cols* column specification.

Math mode

For inline math, use `\(...\)` or `$...$`. For displayed math, use `\[...\]` or `\begin{equation}`.

Superscript^x	`^{x}`	Subscript$_x$	`_{x}`
$\frac{x}{y}$	`\frac{x}{y}`	$\sum_{k=1}^{n}$	`\sum_{k=1}^n`
$\sqrt[n]{x}$	`\sqrt[n]{x}`	$\prod_{k=1}^{n}$	`\prod_{k=1}^n`

Math-mode symbols

$\leq$	`\leq`	$\geq$	`\geq`	$\neq$	`\neq`
$\times$	`\times`	$\div$	`\div`	$\pm$	`\pm`
$\circ$	`^{\circ}`	$\circ$	`\circ`	$\prime$	`\prime`
∞	`\infty`	$\neg$	`\neg`	$\wedge$	`\wedge`
$\supset$	`\supset`	$\forall$	`\forall`	$\in$	`\in`
$\subset$	`\subset`	$\exists$	`\exists`	$\notin$	`\notin`
$\cup$	`\cup`	$\cap$	`\cap`	$\mid$	`\mid`
$\dot{a}$	`\dot a`	$\hat{a}$	`\hat a`	$\bar{a}$	`\bar a`
α	`\alpha`	β	`\beta`	γ	`\gamma`
ϵ	`\epsilon`	ζ	`\zeta`	η	`\eta`
θ	`\theta`	ι	`\iota`	κ	`\kappa`
λ	`\lambda`	μ	`\mu`	ν	`\nu`
π	`\pi`	ρ	`\rho`	σ	`\sigma`
υ	`\upsilon`	ϕ	`\phi`	χ	`\chi`
ω	`\omega`	Γ	`\Gamma`	Δ	`\Delta`
Λ	`\Lambda`	Ξ	`\Xi`	Π	`\Pi`
Υ	`\Upsilon`	Φ	`\Phi`	Ψ	`\Psi`

$\approx$	`\approx`		
$\cdot$	`\cdot`		
$\cdots$	`\cdots`		
$\vee$	`\vee`		
$\rightarrow$	`\rightarrow`		
$\Rightarrow$	`\Rightarrow`		
$\Leftrightarrow$	`\Leftrightarrow`		
$\tilde{a}$	`\tilde a`		
ε	`\varepsilon`		
ϑ	`\vartheta`		
ξ	`\xi`		
τ	`\tau`		
ψ	`\psi`		
Θ	`\Theta`		
Σ	`\Sigma`		
Ω	`\Omega`		

Bibliography and citations

When using BibTeX, you need to run `latex`, `bibtex`, and `latex` twice more to resolve dependencies.

Citation types

`\cite{`*key*`}`	Full author list and year. (Watson and Crick 1953)
`\citeA{`*key*`}`	Full author list. (Watson and Crick)
`\citeN{`*key*`}`	Full author list and year. Watson and Crick (1953)
`\shortcite{`*key*`}`	Abbreviated author list and year. ?
`\shortciteA{`*key*`}`	Abbreviated author list. ?
`\shortciteN{`*key*`}`	Abbreviated author list and year. ?
`\citeyear{`*key*`}`	Cite year only. (1953)

All the above have an `NP` variant without parentheses; Ex. `\citeNP`.

BibTeX entry types

`@article`	Journal or magazine article.
`@book`	Book with publisher.
`@booklet`	Book without publisher.
`@conference`	Article in conference proceedings.
`@inbook`	A part of a book and/or range of pages.
`@incollection`	A part of book with its own title.
`@misc`	If nothing else fits.
`@phdthesis`	PhD. thesis.
`@proceedings`	Proceedings of a conference.
`@techreport`	Tech report, usually numbered in series.
`@unpublished`	Unpublished.

BibTeX fields

`address`	Address of publisher. Not necessary for major publishers.
`author`	Names of authors, of format ....
`booktitle`	Title of book when part of it is cited.
`chapter`	Chapter or section number.
`edition`	Edition of a book.
`editor`	Names of editors.
`institution`	Sponsoring institution of tech. report.
`journal`	Journal name.
`key`	Used for cross ref. when no author.
`month`	Month published. Use 3-letter abbreviation.
`note`	Any additional information.
`number`	Number of journal or magazine.
`organization`	Organization that sponsors a conference.
`pages`	Page range (`2,6,9--12`).
`publisher`	Publisher's name.
`school`	Name of school (for thesis).
`series`	Name of series of books.
`title`	Title of work.
`type`	Type of tech. report, ex. "Research Note".
`volume`	Volume of a journal or book.
`year`	Year of publication.

Not all fields need to be filled. See example below.

Common BibTeX style files

`abbrv`	Standard	`abstract`	`alpha` with abstract
`alpha`	Standard	`apa`	APA
`plain`	Standard	`unsrt`	Unsorted

The LaTeX document should have the following two lines just before `\end{document}`, where `bibfile.bib` is the name of the BibTeX file.

```
\bibliographystyle{plain}
\bibliography{bibfile}
```

BibTeX example

The BibTeX database goes in a file called *file*`.bib`, which is processed with `bibtex file`.

```
@String{N = {Na\~ture}}
@Article{WC:1953,
  author  = {James Watson and Francis Crick},
  title   = {A structure for Deoxyribose Nucleic Acid},
  journal = N,
  volume  = {171},
  pages   = {737},
  year    = 1953
}
```

Sample LaTeX document

```
\documentclass[11pt]{article}
\usepackage{fullpage}
\title{Template}
\author{Name}
\begin{document}
\maketitle

\section{section}
\subsection*{subsection without number}
text \textbf{bold text} text. Some math: $2+2=5$
\subsection{subsection}
text \emph{emphasized text} text. \cite{WC:1953}
discovered the structure of DNA.

A table:
\begin{table}[!th]
\begin{tabular}{|l|c|r|}
\hline
first  &  row  &  data \\
second &  row  &  data \\
\hline
\end{tabular}
\caption{This is the caption}
\label{ex:table}
\end{table}

The table is numbered \ref{ex:table}.
\end{document}
```


Şekil 4.10: LATEX Kılavuzu-2

Kaynakça

Arel-Bundock, Vincent. 2016. "WDI: World Development Indicators (World Bank)". https://cran.r-project.org/web/packages/WDI/WDI.pdf.

Arnold, Jeffrey B. 2016. "ggthemes: Extra Themes, Scales and Geoms for 'ggplot2"'. https://cran.r-project.org/package=ggthemes.

Bache, Stefan Milton, ve Hadley Wickham. 2016. "magrittr: A Forward-Pipe Operator for R". https://cran.r-project.org/web/packages/magrittr/magrittr.pdf.

Bates, Douglas, ve Martin Maechler. 2016. "Matrix: Sparse and Dense Matrix Classes and Methods". https://cran.r-project.org/package=Matrix.

Bengtsson, Henrik. 2016. "R.utils: Various Programming Utilities". https://cran.r-project.org/web/packages/R.utils/R.utils.pdf.

Buuren, Stef van, Karin Groothuis-Oudshoorn, Alexander Robitzsch, Gerko Vink, Lisa Doove, ve Shahab Jolani. 2015. "mice: Multivariate Imputation by Chained Equations". https://cran.r-project.org/package=mice.

Calaway, Rich, ve Steve Weston. 2015. "foreach: Provides Foreach Looping Construct for R". https://cran.r-project.org/package=foreach.

Champely, Stephane. 2018. "pwr: Basic Functions for Power Analysis". https://cran.r-project.org/package=pwr.

Cui, Boxuan. 2018. "DataExplorer: Data Explorer". https://cran.r-project.org/package=DataExplorer.

Dowle, Matt, Arun Srinivasan, Jan Gorecki, Tom Short, Steve Lianoglou, ve Eduard Antonyan. 2016. "data.table: Extension of 'data.frame"'. https://cran.r-project.org/web/packages/data.table/data.table.pdf.

Epskamp, Sacha, Giulio Costantini, Jonas Haslbeck, Angelique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, ve Denny Borsboom. 2018. "qgraph: Graph Plotting

Methods, Psychometric Data Visualization and Graphical Model Estimation". https://cran.r-project.org/package=qgraph.

Everitt, B, ve T Hothorn. 2011. *An Introduction to Applied Multivariate Analysis with R*. Use R! Springer New York. https://books.google.com.tr/books?id=BB51TWe94EwC.

Everitt, Brian S., ve Torsten Hothorn. 2015. "MVA: An Introduction to Applied Multivariate Analysis with R". https://cran.r-project.org/package=MVA.

Greg Snow. 2016. "TeachingDemos: Demonstrations for Teaching and Learning". https://cran.r-project.org/package=TeachingDemos.

Harrell Jr, Frank E. 2016. "Hmisc: Harrell Miscellaneous". https://cran.r-project.org/web/packages/Hmisc/Hmisc.pdf.

Hlavac, Marek. 2015. "stargazer: beautiful LATEX, HTML and ASCII tables from R statistical output". https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdf.

Honaker, James, Gary King, ve Matthew Blackwell. 2016. "Amelia: A Program for Missing Data". https://cran.r-project.org/package=Amelia.

Johnson, Richard A., ve Dean W. Wichern. 1999. *Applied Multivariate Statistical Analysis*. 4th baskı. Upper Saddle River,NJ.

Jonge, Edwin de, ve Martijn Tennekes. 2016. "tabplotd3: Tabplotd3, interactive inspection of large data". https://cran.r-project.org/package=tabplotd3.

Leo, Lahti, Biecek Przemyslaw, Kainu Markus, ve Huovari Janne. 2016. "eurostat: Tools for Eurostat Open Data". https://cran.r-project.org/web/packages/eurostat/eurostat.pdf.

Matloff, N. 2011. *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press. https://books.google.com.tr/books?id=4jL3sfmP1l8C.

Papadakis, Manos, Michail Tsagris, Marios Dimitriadis, Stefanos Fafalios, Ioannis Tsamardinos, Matteo Fasiolo, Giorgos Borboudakis, John Burkardt, Changliang Zou, ve Kleanthi Lakiotaki. 2018. "Rfast: A Collection of Efficient and Extremely Fast R Functions". https://cran.r-project.org/package=Rfast.

Persson, Eric. 2016. "OECD: Search and Extract Data from the OECD". https://cran.r-project.org/package=OECD.

Petersen, Anne Helby, ve Claus Thorn Ekstrøm. 2018. "dataMaid: A Suite of Checks for Identification of Potential Errors in a Data Frame as Part of the Data Screening Process". https://cran.r-project.org/package=dataMaid.

R Core Team. 2016. "foreign: Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase"'. https://cran.r-project.org/package=foreign.

Revelle, William. 2016. "Procedures for Psychological, Psychometric, and Personality Research". http://personality-project.org/r/psych-manual.pdf.

Sarkar, Deepayan. 2016. "lattice: Trellis Graphics for R A powerful and elegant high-level

data visuali". https://cran.r-project.org/package=lattice.

Schwartz, Marc. 2015. "WriteXLS: Cross-Platform Perl Based R Function to Create Excel 2003 (XLS) and Excel 2007 (XLSX) Files". https://cran.r-project.org/web/packages/WriteXLS/WriteXLS.pdf.

Signorell, Andri. 2018. "DescTools: Tools for Descriptive Statistics". https://cran.r-project.org/package=DescTools.

Stekhoven, Daniel J. 2016. "missForest: Nonparametric Missing Value Imputation using Random Forest". https://cran.r-project.org/package=missForest.

Studer, Martin. 2016. "XLConnect: Excel Connector for R". https://cran.r-project.org/web/packages/XLConnect/XLConnect.pdf.

Tennekes, Martijn, ve Edwin de Jonge. 2017. "tabplot: Tableplot, a Visualization of Large Datasets". https://cran.r-project.org/package=tabplot.

Tillé, Yves. 2016. "sampling: Survey Sampling". https://cran.r-project.org/package=sampling.

Unwin, Antony. 2018. "OutliersO3: Draws Overview of Outliers (O3) Plots". https://cran.r-project.org/package=OutliersO3.

Verzani, John. 2018. "UsingR: Data Sets, Etc. for the Text 'Using R for Introductory Statistics', Second Edition". https://cran.r-project.org/package=UsingR.

Warnes, Gregory R., Ben Bolker, Gregor Gorjanc, Gabor Grothendieck, Ales Korosec, Thomas Lumley, Don MacQueen, Arni Magnusson, ve Jim Rogers. 2016. "gdata: Various R Programming Tools for Data Manipulation". https://cran.r-project.org/package=gdata.

Warnes, Gregory R., Ben Bolker, Thomas Lumley, ve Randall C Johnson. 2015. "gmodels: Various R Programming Tools for Model Fitting". https://cran.r-project.org/web/packages/gmodels/gmodels.pdf.

Wei, Taiyun, ve Viliam Simko. 2017. "corrplot: Visualization of a Correlation Matrix". https://cran.r-project.org/package=corrplot.

Wickham, Hadley. 2014. *Advanced R*. Chapman & Hall/CRC The R Series. Taylor & Francis. https://books.google.com.tr/books?id=PFHFNAEACAAJ.

———. 2016. "stringr: Simple, Consistent Wrappers for Common String Operations". https://cran.r-project.org/package=stringr.

———. 2017a. "forcats: Tools for Working with Categorical Variables". https://cran.r-project.org/web/packages/forcats/forcats.pdf.

———. 2017b. *R for Data Science*. 1st baskı. O'Reilly Media. http://r4ds.had.co.nz/.

———. 2018. "pryr: Tools for Computing on the Language". https://cran.r-project.org/package=pryr.

Wickham, Hadley, Winston Chang, ve RStudio. 2016. "ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics". https://cran.r-project.org/package=ggplot2.

Wickham, Hadley, Jim Hester, ve Romain Francois. 2017. "readr: Read Rectangular Text

Data". https://cran.r-project.org/package=readr.

Wickham, Hadley, ve Romain Francois. 2016. "dplyr: A Grammar of Data Manipulation". https://cran.r-project.org/package=dplyr.

Wickham, Hadley, ve Lionel Henry. 2017. "tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions". https://cran.r-project.org/package=tidyr.

Dizin