



R ile İstatistiksel Programlamaya Giriş: Temel Kavramlar ve Uygulamalar

Risk✓*Active*

Outline

1. R'a Giriş
2. Temel Komut ve Kavramlar
3. Hesaplama Aracı Olarak R
4. Programlama Dili Olarak R
5. tidyverse: Data Manipülasyonu
6. tidyverse: Seçilmiş Örnekler
7. Data İşleme
8. "R Base" Türü Grafik Çizimi
9. "ggplot2" ile Grafik Çizimi

R'a Giriş

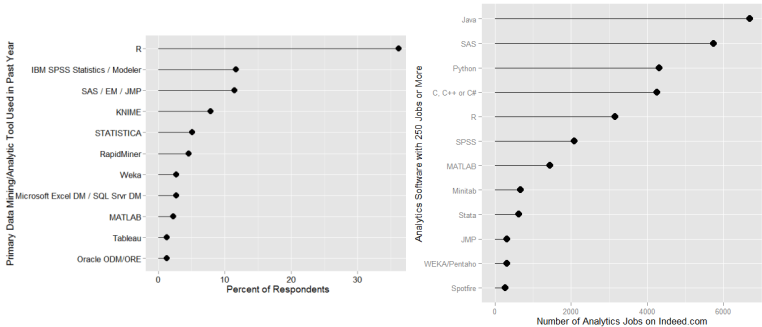
- Temeli 1976 yılından bu yana Bell Laboratuvarları'nda istatistiksel programlama dili olarak geliştirilen **S** diline dayanır.
 - S dili, araştırma ve veri analizi için geliştirilmiştir.
 - Daha sonraları S-Plus olarak piyasaya sürülmüştür.
- R dili, S ve S-Plus dillerinin bir türevi olarak 1990'lı yıllarda Auckland Üniversitesi İstatistik Bölümünde, **Ross Ihaka** ve **Robert Gentleman** tarafından açık kaynak kodlu bir platform olarak geliştirilmiştir.
- 1997 yılında dünyanın çeşitli yerlerindeki araştırmacılar R'yi geliştirmek için bir araya gelmiştir. Bu gruba “**R core team**” adı verilmiştir.
- R dilinin ilk sürümü “R core team” tarafından 29 Şubat 2000 tarihinde yayınlanmıştır.

- R GNU (Genel Kamu Lisansı veya GPL) S'dir.
 - Veri işleme, hesaplama ve grafik gösterimi için bir dil ve çevre sağlar.
- Geniş bir yelpazede istatistiki ve grafiksel teknikleri içerir.
 - doğrusal (linear) ve doğrusal olmayan (non-linear) modelleme, istatistik testleri, zaman-serileri analizi, sınıflandırma, kümeleme ... vb.
- Açık kaynak kodlu olması itibariyle geliştirilmeye çok yatkındır.

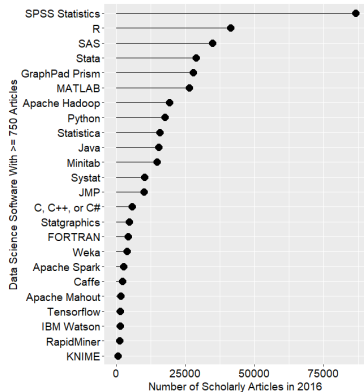
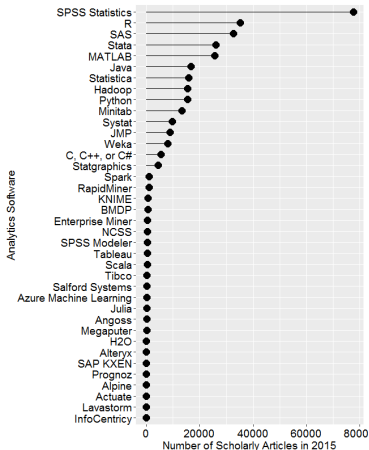
R ve Diğer Yazılımlar

- 2015 yılı baz alınarak yayınlanan makale R ile diğer yazılımları farklı kıstaslara göre kıyaslamaktadır.

<http://r4stats.com/articles/popularity/>

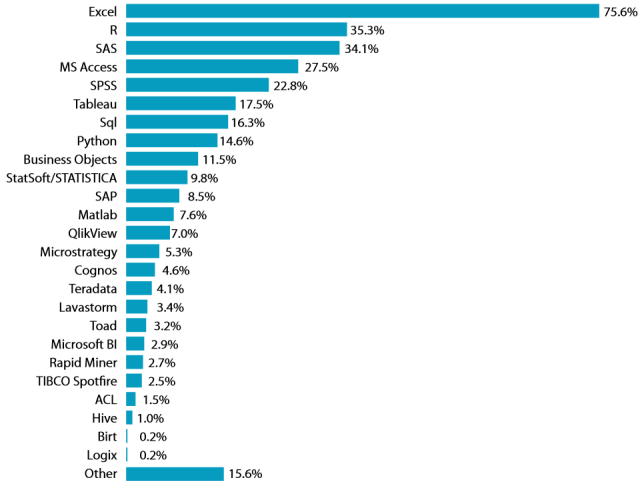


R ve Diğer Yazılımlar

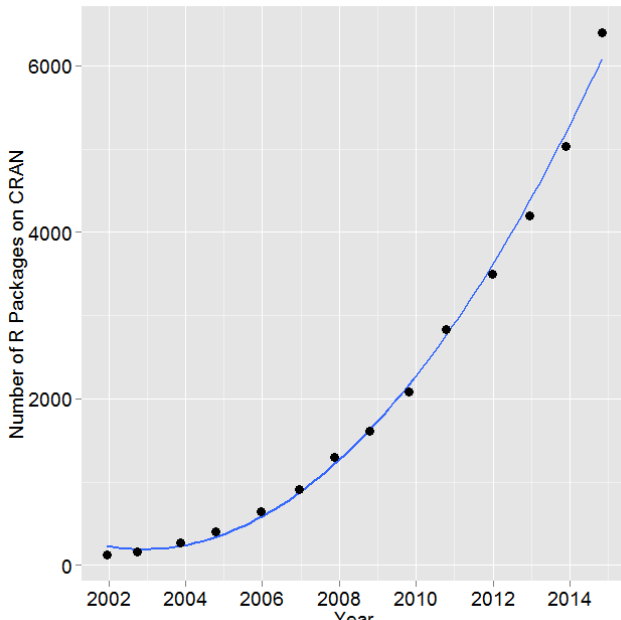


R ve Diğer Yazılımlar

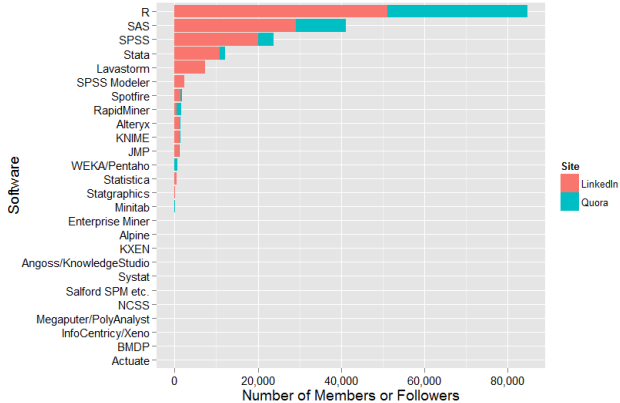
What self-service analytic tool are you currently using?



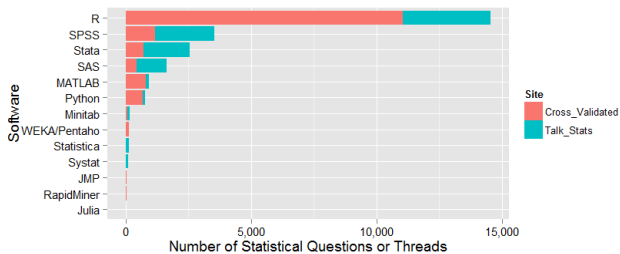
R ve Diğer Yazılımlar



R ve Diğer Yazılımlar



R ve Diğer Yazılımlar



Neden R?

- R kullanıcıları sihirbaz gibidirler.
 - R yaygın olarak kullanılan SPSS, EViews gibi istatistik paket programlardan farklıdır. R bir istatistik paket program değil **istatistiksel yazılım dilidir.**
 - Matematik, İstatistik, Ekonometri, Finans alanlarında çalışma yapan araştırmacılar tarafından geliştirilen fonksiyonları kullanabilirsiniz.
 - Açık kaynak kodlu olması nedeniyle fonksiyonların ne yaptığı kullanıcılar tarafından incelenip, doğrulanabilir.
 - Yeterli deneyim sonrası kendi fonksiyon paketlerinizi yazabilir ve yayınlayabilirsiniz.
 - Diğer diller ve programlar ile entegre olur.
 - İşletim sisteminden bağımsız çalışır.

Neden R?

- Dinamik raporlamalar (**RMarkdown**) veya web siteleri (**RShiny**) hazırlanabilir ve yayınlanabilir.

Stochastic Option Pricing

Configuration

Option Pricing

Volatility Surface

Simulations

Vanilla Options FOR/DOM

Trade date:
2016-11-19

Spot date:
2016-11-23

Expiry date:
2016-12-31

Delivery date:
2016-12-31

Style:
European

Type:
Call

Option Terms

Spot:
1,05

Strike:
1,06

Notional:
1000000

Risk Free (domestic) %:
0,381

Risk Free (foreign) %:
-0,933

Heston Volatility

Volatility Calculation:
by Calibration by Manual

Volatility %:
9,4837

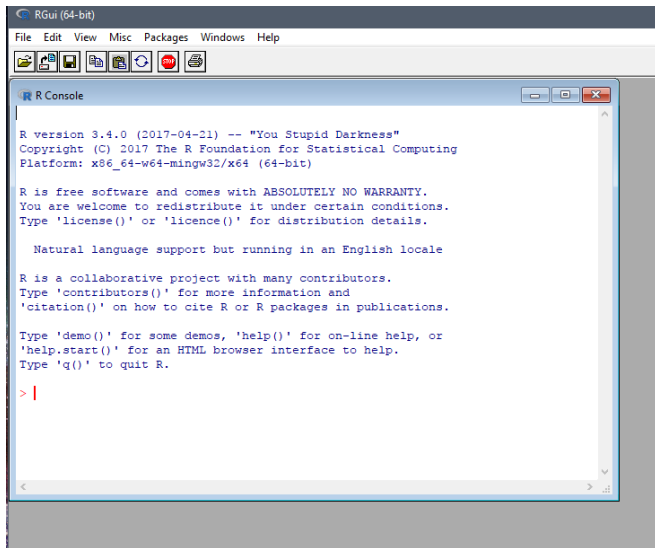
Option Pricing Results

	Premium %	Premium Amount
B-S (Heston Volatility)	0.7190%	7196.2223
Heston Analytic	0.7190%	7196.2708
Heston MC	0.6392%	6392.4995
Heston MC Lower	0.5750%	5750.1291
Heston MC Upper	0.7035%	7034.8699

R Terminal ve RStudio Kurulumu

- R, açık kaynak kodlu bir yazılımdır ve kendi editörünü de sunmaktadır. R editörü grafiksel bir arayüz olmayıp eski tip bir yazılım konsoludur. Ancak bireysel kullanım sürümü ücretsiz olan **RStudio** programı modern ve kullanışlı bir arayüz sunmaktadır.
- R Terminal'i kurabilmek için öncelikle <https://cran.r-project.org/> adresine girip işletim sistemine uygun program indirilmelidir.
 1. Açılan sayfada "Download R for Windows" linkine tıklanır.
 2. Sayfanın üstünde yer alan "base" linkine tklanır.
 3. "Download R 3.5.1 for Windows" linkine tıklandığında kurulum dosyası indirilmeye başlanacaktır.

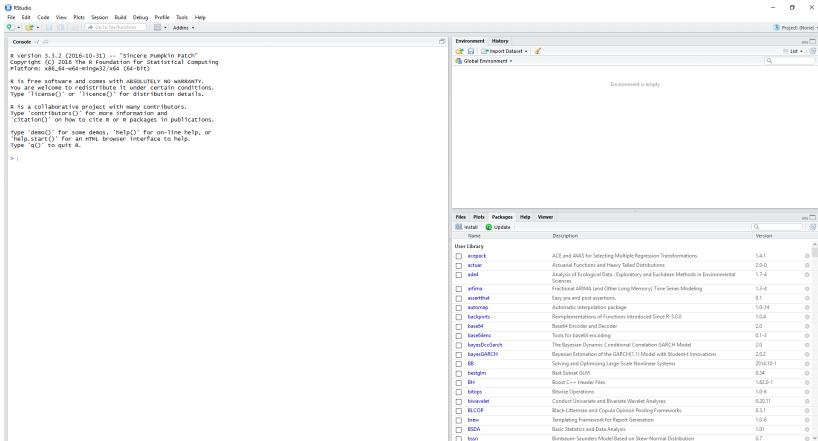
R Terminal ve RStudio Kurulumu



R Terminal ve RStudio Kurulumu

- Bir sonraki adımda RStudio'nun kurulumuna geçilecektir.
- RStudio, <https://rstudio.com/products/rstudio/download/> adresine girip işletim sistemi için uygun olan link seçilerek indirilip kurulacaktır. Kurulum sonrası program başlatıldığında aşağıdaki ekran görüntülenecektir.

R Terminal ve RStudio Kurulumu



R Oturumu (Session) ve Yönetimi

- Çalışma klasörü, veri dosyalarının okunacağı ve kaydedileceği klasördür.
- Çalışma klasörünün görüntülenmesi için `getwd()` fonksiyonu kullanılır. (wd = working directory, çalışma klasörü anlamına gelmektedir.)

```
getwd()
```

- Çalışma klasörünü değiştirmek için `setwd()` fonksiyonu kullanılır.

```
setwd("C:/Users/User1/Documents")
```

Yardım Alma (Help)

- R'in yardım dökümanlarına toplu şekilde ulaşmak için `help.start()` kullanılır.
- Örneğin `sd()` standart sapma fonksiyonu ile ilgili yardım almak için aşağıdaki komutlardan bir tanesi kullanılabilir.

```
help(sd)
```

```
?sd
```

```
#R Site Search kullanarak online destek almak için
```

```
RSiteSearch("sd")
```

Yardım Alma (Help)

- Bir fonksiyonun kullanımı için gerekli olan argümanları hızlıca görebilmek için `args()` veya `str()` fonksiyonları kullanılır.

```
args(sd)
```

```
## function (x, na.rm = FALSE)  
## NULL
```

```
str(sd)
```

```
## function (x, na.rm = FALSE)
```

Örnek:

0 ile 1 arasında rassal olarak oluşturulan 100 adet sayının standart sapmasını bulalım.

```
d <- runif(n=100, min=0, max=1)
sd(x=d)
```

```
## [1] 0.2978984
```

Destek Sayfaları

- <http://cran.r-project.org>: R dilini geliştiren ekibin sayfasıdır. R ile ilgili tüm güncellemeler bu sitede yayınlanmaktadır.
- <http://r-bloggers.com>: Çok sayıda farklı ve kısa makalelerin yer aldığı sitedir.
- <http://r-statistics.com>: R ile istatistiksel uygulamalar konusunda faydalanılabilecek bir sitedir.
- <http://blog.rstudio.org>: RStudio programının blog sitesidir.
- <http://stackoverflow.com>: Sadece R değil tüm programlama dilleri ile yardım alınabilecek, çok sayıda aktif üyeye sahip oldukça kullanışlı bir sitedir.

- <http://github.com>: Sadece R değil tüm programlama dilleri ile yardım alınabilecek, çok sayıda aktif üyeye sahip oldukça kullanışlı bir sitedir.
- <https://www.listendata.com>
- <http://REgitimMerkezi.com>

- RStudio tarafından hazırlanan **CheatSheet**'ler, paketlerde yer alan fonksiyonların kullanımlarını içeren oldukça pratik kaynaklardır.
- <https://www.rstudio.com/resources/cheatsheets/> sayfasından ulaşılabilir. En çok kullanılanlar;
 - Data Transformation with **dplyr**
 - Dates and times **lubridate**
 - Data Visualization with **ggplot2**
 - Base **R**
 - Machine Learning Modelling in **R**

Temel Komut ve Kavramlar

R'da Temel İşlemler

- Standart aritmetik operatörler: +, -, *, / ve ^

```
(3 + 3 * 5 - 81 / 9) ^ 2
```

```
## [1] 81
```

- Matematiksel fonksiyonlar ve değişkenler

```
10 %% 3      #mod alma
```

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

```
sqrt(9)      #karekök
```

```
## [1] 3
```

R'da Temel İşlemler

```
abs(-0.5)      #mutlak değer
```

```
## [1] 0.5
```

```
sign(-0.2)     #işaret fonksiyonu
```

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

```
factorial(5)   #faktöriyel
```

```
## [1] 120
```

```
exp(1)         #e1
```

```
## [1] 2.718282
```

R'da Temel İşlemler

```
log(2.271)    #doğal logaritma
```

```
## [1] 0.8202203
```

```
log2(16)      #2 tabanında log.
```

```
## [1] 4
```

```
log10(1000)   #10 tabanında log.
```

```
## [1] 3
```

```
sin(pi/2)     #sinüs
```

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

R'da Temel İşlemler

```
cos(pi)      #cosinüs
```

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

```
round(pi, digits = 4)
```

```
## [1] 3.1416
```

```
options(digits = 6); pi
```

```
## [1] 3.14159
```

```
ceiling(3.4) #büyük tam sayıya yuvarlar
```

```
## [1] 4
```

R'da Temel İşlemler

```
floor(3.44)  #küçük tam sayıya yuvarlar
```

```
## [1] 3
```

Temel Komutlar

- R'da veri tanımlamak için "<-" veya "=" işaretleri kullanılmaktadır.
- R'da yorum satırı "#" işareti ile sağlanır. "#" işaretinden sonraki ifadeler program tarafından değerlendirilmez.

```
#veri girişi  
a <- 5  
b <- -2  
x <- c(8,36,37) #c, "combine" kelimesinin ilk  
                #harfidir. Vektör oluşturur.  
y <- c(1:50)  
z <- 3 * pi + 4^a - b
```

- Çalışma ortamındaki tüm değişkenler `ls()` ile listelenir.

Temel Komutlar

```
ls()
```

```
## [1] "a" "b" "d" "x" "y" "z"
```

- Bir değişkeni silmek için `rm()` fonksiyonu kullanılır.

```
rm(d) #d değişkenini silelim.
```

```
ls() #değişkenleri listeleyelim.
```

```
## [1] "a" "b" "x" "y" "z"
```

- R konsolda yapılan son işlemin sonucunu bir değişkene `.Last.value` ifadesi ile alabiliriz.

```
10 / 3 + sqrt(10) + exp(pi * log(10))
```


Temel Komutlar

```
## [1] 1391.95
```

```
t <- .Last.value #son yapılan işlemin sonucunu saklar
```

- Çalışma ortamındaki tüm değişkenleri `rm(list=ls())` ile silebiliriz.

```
rm(list=ls()) #tüm değişkenler silindi.
```

```
ls() #değişkenleri listeleyelim.
```

```
## character(0)
```

Paket (Library) Kurulumu ve Yükleme

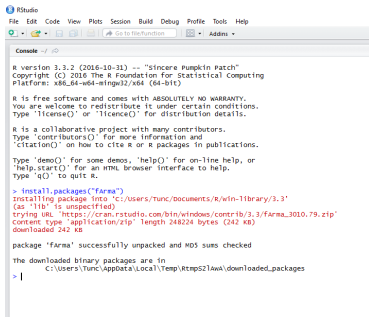
- R programı paket veya kütüphane (“package” veya “library”) olarak adlandırılan modüllerden oluşmaktadır.
- R ilk kurulduğunda temel “**base**” R sistemi kurulmuş olmaktadır.
- R, farklı paketlerin modüler kullanım sunmasına izin vermektedir ve istenildiğinde programa dahil edilebilmektedir.
- R dilinde yazılmak istenen bir fonksiyonun en baştan yazılması yerine varsa bu fonksiyonu barındıran bir paketi kullanmak zamandan tasarruf etmemizi sağlar.
- R’ın açık kaynak kodlu bir yazılım olması nedeniyle isteyen herkes bir paket geliştirerek diğer kullanıcıların kullanımına sunabilmektedir.

Library Kurulumu - CRAN

- R Cran'da Mayıs 2018 itibari ile 12550 adet paket bulunmaktadır. `available.packages()` fonksiyonu ile R Cran'daki tüm library'ler listelenebilir. R Cran'daki tüm library listesine <https://cran.r-project.org/web/packages/> adresinden de ulaşılabilmektedir.
- <http://cran.r-project.org/web/views/> adresinde paketler, konu başlıklarına göre sınıflandırılmıştır. Bazı sınıflar: Finance, Econometrics, Distributions, Extreme Value, Survival, TimeSeries ... vb.
- Library'ler farklı biçimlerde yüklenebilir.

```
install.packages("fArma")
```

Library Kurulumu - CRAN



```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to File/Function Go to File/Function Adds
Console
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

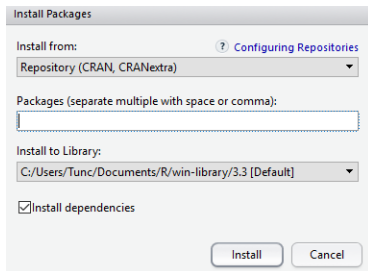
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("farma")
Installing package into 'C:/Users/Tunc/Documents/R/win-library/3.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/farma_3.010.79.zip'
content type 'application/zip' length 248224 bytes (242 Kb)
downloaded 242 Kb

package 'farma' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:/Users/Tunc/AppData/Local/Temp/ktmp521awa/downloaded_packages
> |
```

(a) Panel kullanarak



Install Packages

Install from: [? Configuring Repositories](#)

Repository (CRAN, CRANextra) ▼

Packages (separate multiple with space or comma):

Install to Library:

C:/Users/Tunc/Documents/R/win-library/3.3 [Default] ▼

☒ Install dependencies

Install Cancel

(b) Menü kullanarak

Library Kurulumu - CRAN

```
install.packages(c("fArma","fBonds")) # toplu library  
                                         # yükleme  
remove.packages("fArma")                # library silme  
remove.packages(c("fArma","fBonds"))    # toplu library  
                                         # silme
```

R Metrics Library Kurulumu

- R Metrics içerisinde yer alan tüm library'leri yüklemek için kullanılacaktır. R Metrics, fARMA, fOptions, fBasics, fBonds gibi kullanışlı paketleri içermektedir.

```
source("http://www.rmetrics.org/Rmetrics.R")  
install.Rmetrics()
```

Library Kullanımı

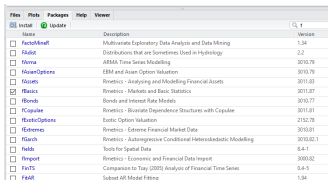
- Yüklenen paketlerin bir çoğu her oturum (session) açılışında aktif halde gelmemektedir.
- Yüklenmiş olan bir paketi kullanmak için `library(paket.ismi)` komutu kullanılır.

```
search() #aktif paketleri listeler
```

```
## [1] ".GlobalEnv" "package:stargazer" "package:sa  
## [4] "package:broom" "package:knitr" "package:xt  
## [7] "package:stats" "package:graphics" "package:gr  
## [10] "package:utils" "package:datasets" "package:me  
## [13] "Autoloads" "package:base"
```

Library Kullanımı

- Library'nin tamamını aktif hale getirmeden içinden belirli bir fonksiyonu kullanmak için `fArma::armaFit()` kod bloğu kullanılabilir.
- Aktif olan bir library deaktif edilebilmesi için `detach(package:paket.ismi)` komutu kullanılır.
- Ayrıca library'leri aktif/deaktif hale getirmek için RStudio'daki "Packages" sekmesi de kullanılabilir.



Files	Plots	Packages	Help	Viewer
Install	Update			
Name	Description	Version		
<input type="checkbox"/> FactoMineR	Multivariate Exploratory Data Analysis and Data Mining	1.34		
<input type="checkbox"/> FAdist	Distributions that are Sometimes Used in Hydrology	2.2		
<input type="checkbox"/> fArma	ARMA Time Series Modelling	3010.79		
<input type="checkbox"/> fAsianOptions	ESM and Asian Option Valuation	3010.79		
<input type="checkbox"/> fVaurs	Rmetrics - Analyzing and Modeling Financial Assets	3011.81		
<input checked="" type="checkbox"/> fRomics	Rmetrics - Markets and Basic Statistics	3011.87		
<input type="checkbox"/> fRonds	Bonds and Interest Rate Models	3010.77		
<input type="checkbox"/> fRcopulae	Rmetrics - Bivariate Dependence Structures with Copulae	3011.81		
<input type="checkbox"/> fRexoticOptions	Exotic Option Valuation	2132.78		
<input type="checkbox"/> fRxtremes	Rmetrics - Extreme Financial Market Data	3010.81		
<input type="checkbox"/> fRwch	Rmetrics - Autoregressive Conditional Heteroskedastic Modelling	3010.02.1		
<input type="checkbox"/> fRds	Tools for Spatial Data	0.4-1		
<input type="checkbox"/> fRreport	Rmetrics - Economic and Financial Data Import	3000.02		
<input type="checkbox"/> fRrTS	Companion to Tsay (2005) Analysis of Financial Time Series	0.4-5		
<input type="checkbox"/> fRrfit	Subset AR Model Fitting	1.34		

- CRAN haricindeki farklı repository'lerden de paketler yüklemek ve bunları çalıştırmak mümkündür.
- CRAN'dan sonra bilinen en geniş kaynak **Github**'tır.
- **devtools** kütüphanesi aracılığı ile kişilerin kullanıcı adı ve paket isimlerini kütüphaneler yüklenmektedir.

```
devtools::install_github("tidyverse/dplyr")
```


R'ı Güncelleme

- R zaman zaman güncellenmektedir. R'nin güncellenmesi için `installr` paketi kullanılmaktadır.

```
install.packages("installr")
```

```
### 1.Alternatif ###
```

```
installr::updateR()
```

```
### 2.Alternatif ###
```

```
library(installr)
```

```
updateR()
```

Finansal Uygulamalarda Kullanılan Paketler

- **PerformanceAnalytics**, **parma**, **urca**, **FinTS**, **NMOF**
- **Rmetrics**: **fArma**, **fAsianOptions**, **fAssets**, **fBasics**, **fBonds**, **fCopulae**, **fExoticOptions**, **fExtremes**, **fGarch**, **flmport**, **fNonlinear**, **fOptions**, **fPortfolio**, **fRegression**, **fSeries**, **fTrading**, **fUnitRoots**
- **QRM**, **distr**, **fitdistrplus**
- **quantmod**, **TTR**, **highfrequency**
- **rugarch**, **rmgarch**, **tseries**, **forecast**

Hesaplama Aracı Olarak R

Veri Tipleri

- R'da kullanılan beş temel veri tipi vardır.
 - numeric (reel sayılar)
 - integer (tam sayılar)
 - complex (kompleks sayılar)
 - character (metin)
 - logical (mantıksal, TRUE/FALSE)
- Değişken isimleri rakam ile başlayamaz. Değişken isimleri büyük-küçük harf duyarlıdır.
- **Inf, NA, NaN, NULL, TRUE, FALSE, break, else, for, function, if, in, next, repeat, return** ve **while**, değişken adı olarak kullanılamaz.
- **c, q, t, C, D, I, diff, length, mean, pi, range** ve **var**, zorunlu olmadıkça değişken adı olarak kullanılmamalıdır.

Nesneler, Objeler ve Temel R Bileşenleri

- Bir değişkenin hangi veri tipinde olduğunu öğrenmek için `class()` veya `typeof()` fonksiyonları kullanılır.

```
x <- 3.2  
class(x)
```

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

```
typeof(x)
```

```
## [1] "double"
```

```
y <- 3  
class(y)
```

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

Nesneler, Objeler ve Temel R Bileşenleri

```
z <- 3L  
class(z)
```

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

- Belirli bir veri tipinde birden fazla değeri içerecek vektör bir değişken oluşturulması için aşağıdaki kod yapısı kullanılır.

```
num <- numeric(5)  
num
```

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

```
str <- character(3)  
str
```

Nesneler, Objeler ve Temel R Bileşenleri

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

Mantıksal Karşılaştırma İşlemleri

== Eşit
!= Eşit değil
< Küçük
> Büyük
<= Küçük veya eşit
>= Büyük veya eşit
& Vektörel ve
&& Skaler ve
| Vektörel veya
|| Skaler veya
! Değil

Mantıksal Karşılaştırma İşlemleri

```
7 == (4 + 3 * 1)
```

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

```
FALSE != TRUE
```

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

```
"Ankara" == "ankara"
```

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

```
6 <= 5
```

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

```
"Adana" < "Bursa"
```

Mantıksal Karşılaştırma İşlemleri

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

```
TRUE <= FALSE
```

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

- **&** (ve) operatörü iki mantıksal değeri karşılaştırır. Her ikisi de TRUE yani 1'e eşit ise TRUE, diğer durumlarda FALSE (0) sonucu verir.

```
TRUE & FALSE
```

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

```
(5 != 4) & (7 > 3.5)
```

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

Mantıksal Karşılaştırma İşlemleri

- `|` (veya) operatörü iki mantıksal değeri karşılaştırır. Her ikisi de FALSE yani 0'a eşit ise FALSE, diğer durumlarda TRUE (1) sonucu verir.

```
TRUE | FALSE
```

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

```
(5 == 4) | (7 < 3.5)
```

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

- Vektörel “ve” (`&`) iki vektörün her bir elemanını karşılaştırıp her bir değer için bir sonuç üretir. Skaler ve (`&&`) ise tek bir değer için karşılaştırma yapar.

Mantıksal Karşılaştırma İşlemleri

```
x <- c(FALSE, TRUE, FALSE); y <- c(FALSE, FALSE, TRUE)
x & y
```

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

```
x[2] && y[2]
```

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

- Vektörel “veya” (||) iki vektörün her bir elemanını karşılaştırıp her bir değer için bir sonuç üretir. Skaler ve (||) ise tek bir değer için karşılaştırma yapar.

```
x <- c(FALSE, TRUE, FALSE); y <- c(FALSE, FALSE, TRUE)
x | y
```

Mantıksal Karşılaştırma İşlemleri

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

```
x[2] || y[2]
```

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

Veri Tipi Değiştirme

- Veri tipi değiştirmek için aşağıdaki fonksiyonlar kullanılır.

```
# as.numeric()  
# as.integer()  
# as.complex()  
# as.character()  
# as.logical()
```

```
x <- 3.92  
class(x)
```

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

```
x <- as.integer(x)  
x
```

Veri Tipi Değiştirme

```
## [1] 3
```

```
class(x)
```

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

- Bu method her zaman geçerli değildir.

```
x <- "Deneme"
```

```
class(x)
```

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

```
as.numeric(x)
```

```
## Warning: NAs introduced by coercion
```

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

Veri Tipi Değiştirme

- Veri tipini sorgulamak için aşağıdaki fonksiyonlar kullanılır. Bu fonksiyonların cevabı istenilen veri tipinde olup olmadığına göre TRUE veya FALSE sonucunu verecektir.

```
# is.numeric()  
# is.integer()  
# is.complex()  
# is.character()  
# is.logical()  
# is.null()  
# is.na()
```


Veri Tipi Değiştirme

```
x <- 5.69  
is.numeric(x)
```

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

```
is.integer(x)
```

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

Vektörler

- R'daki en temel nesneler vektörlerdir. En basit anlamda vektör, aynı tipte verilerden oluşan bir dizi olarak tanımlanabilir.
- Vektör veri tipi, aynı türden verilerden oluşan atomik vektördür. Yazılan programlarla ilgili olarak **“atomic vector”** ibaresini içeren bir uyarı veya hata mesajı alınırsa, burada kast edilen bütün elemanları aynı türde olan en basit vektör tipi olduğuna dikkat edilmelidir.
- Boş bir vektör oluşturmak için `vector()` fonksiyonu kullanılır.

```
# Vektör oluşturma
```

```
x <- vector(mode = "numeric", length = 10)
```

```
y <- vector(mode = "character", length = 3)
```

Vektörler

```
x <- c(3.14, 2.98, 4.56)
```

```
x[1]
```

```
## [1] 3.14
```

```
x[4]
```

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

Vektörler

- Ardaşık tam sayılardan oluşan bir vektör oluşturmaının bir yolu ":" işaretini kullanmaktır.

```
x <- c(7:15)
```

```
x
```

```
## [1] 7 8 9 10 11 12 13 14 15
```

- Birbirinden farklı veri tipindeki elemanları aynı vektörde birleştirmek istenildiğinde **character** > **numeric** > **logical** sıralamasına göre vektör oluşturulmuş olunacaktır.

```
x <- c(1.5, "merhaba", TRUE)
```

```
x
```

```
## [1] "1.5" "merhaba" "TRUE"
```

```
class(x)
```

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

Vektörler - seq()

- `seq()` fonksiyonu, aritmetik bir diziden meydana gelen bir vektör oluşturmak için kullanılır.

by argümanı artış miktarını verecektir.

```
seq(from = 10, to = 22, by = 3)
```

```
## [1] 10 13 16 19 22
```

by argümanı negatif değerlerde alabilmektedir.

```
seq(from = 20, to = 2, by = -4)
```

```
## [1] 20 16 12 8 4
```

Vektörler - seq()

```
# by argümanı yerine length.out ile artış miktarı  
# otomatik belirlenir.  
seq(from = 3, to = 10, length.out = 7)
```

```
[1] 3.00000 4.16667 5.33333 6.50000 7.66667 8.83333 10.00000
```

Vektörler - rep()

- `rep()` fonksiyonu, tekrarlı vektör oluşturmak için kullanılır.

```
rep(x = 11, times = 5)
```

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

```
rep(x = c(2,4,6), times = 3)
```

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

```
rep(x = c(2,4,6), each = 3)
```

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


Basit Vektör İşlemleri

```
x <- 3:6; y <- c(2, 4.5, 5.3, 7.2)
```

```
x + y
```

```
## [1] 5.0 8.5 10.3 13.2
```

```
x >=4
```

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

```
x < y
```

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

```
x * y
```

```
## [1] 6.0 18.0 26.5 43.2
```

Basit Vektör İşlemleri

```
sqrt(x)
```

```
## [1] 1.73205 2.00000 2.23607 2.44949
```

```
log(x)
```

```
## [1] 1.09861 1.38629 1.60944 1.79176
```

Basit Vektör İşlemleri - length()

- Bir vektörün uzunluğunu bulmak için `length()` fonksiyonu kullanılır.

```
t <- c(1:1000)
length(t)
```

```
## [1] 1000
```

Basit Vektör İşlemleri - Elemanların Seçilmesi

- Bir vektörün, (matrisin) elemanlarını görmek için [,] işaretleri kullanılır.

```
x <- 100:200
```

```
x[40]           #40. eleman
```

```
## [1] 139
```

```
x[c(1,4,6)]    #1, 4 ve 6. elemanlar
```

```
## [1] 100 103 105
```

```
x[41:45]       #41 den 45. elemana kadar
```

```
## [1] 140 141 142 143 144
```

Basit Vektör İşlemleri - Elemanların Seçilmesi

```
x <- 1:5
```

```
x[-3]      #3. eleman haricindeki elemanlar
```

```
## [1] 1 2 4 5
```

```
x[-c(1,5)] #1 ve 5. elemanlar haricindeki elemanlar
```

```
## [1] 2 3 4
```

- Bir vektördeki belirli bir sayıdan büyük, küçük veya iki sayı arasında gibi farklı kısıtları sağlayan elemanlarını seçtirmek mümkündür.

```
x <- 100:200
```

```
x[x < 105]
```

Basit Vektör İşlemleri - Elemanların Seçilmesi

```
## [1] 100 101 102 103 104
```

```
x[x >= 190]
```

```
## [1] 190 191 192 193 194 195 196 197 198 199 200
```

```
x[x > 122 & x < 130]
```

```
## [1] 123 124 125 126 127 128 129
```

Basit Vektör İşlemleri - Alt Küme Seçilmesi

- Belirli bir şartı sağlayan vektör elemanlarını listelemek için `subset()` fonksiyonu kullanılabilir.

```
x <- 1:100  
subset(x, x < 9)
```

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

- Belirli bir şartı sağlayan vektör elemanlarının sıra numarasını listelemek için `which()` fonksiyonu kullanılabilir.

```
x <- c(1, 3, 5, 7, 9, 12, 15)  
which(x == 12)
```

```
## [1] 6
```

Basit Vektör İşlemleri - Alt Küme Seçilmesi

```
which(x >= 9)
```

```
## [1] 5 6 7
```


Basit Vektör İşlemleri - c()

- Bir vektöre veri eklemek için `c()` fonksiyonu kullanılır.

```
x <- 3:7
```

```
x
```

```
## [1] 3 4 5 6 7
```

```
x <- c(2, x, 8:11)
```

```
x
```

```
## [1] 2 3 4 5 6 7 8 9 10 11
```

Basit Vektör İşlemleri - rev()

- Bir vektörü tersine çevirmek için `rev()` fonksiyonu kullanılır.

```
x <- 3:7
```

```
rev(x)
```

```
## [1] 7 6 5 4 3
```

Basit Vektör İşlemleri - rank()

- Bir vektördeki elemanların vektör içinde kaçınıcı sırada oldukları **rank()** fonksiyonu ile belirlenir.
- Eğer vektör içinde **NA** veriler mevcut ise, **na.last** argümanı kullanılır.
 - Argüman FALSE olursa NA değerler en başta görünür.
 - Argüman TRUE olursa NA değerler en sonda görünür.
 - “keep” olursa NA değerler sıralamada işleme alınmaz.

```
x <- c(5, 2, 8, 12, 1)
rank(x)
```

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

Basit Vektör İşlemleri - rank()

```
x <- c(5, 2, 8, NA, 12, 1, NA)
rank(x, na.last = TRUE)
```

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

```
rank(x, na.last = "keep")
```

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

Basit Vektör İşlemleri - all() & any()

- Bir vektördeki **tüm** elemanların bir şartı sağlayıp sağlamadıklarını test etmek için **all()** fonksiyonu kullanılır.

```
x <- 3:20 ; all(x > 0)
```

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

- Bir vektördeki **en az bir** elemanın bir şartı sağlayıp sağlamadıklarını test etmek için **any()** fonksiyonu kullanılır.

```
any(x < 3)
```

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

Basit Vektör İşlemleri - unique()

- Bir vektördeki tekrar eden elemanlar, `unique()` fonksiyonu ile temizlenir.

```
x <- c(2, 4, 5, 5, 2, 6, 7, 7, 6, 2, 6, 7)
unique(x)
```

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

Basit Vektör İşlemleri - sort()

- Bir vektördeki elemanlar `sort()` fonksiyonu ile sıralanır.

```
x <- c(-4, 2, -2, 0, 7, -14, 20)
sort(x)
```

```
## [1] -14  -4  -2   0   2   7  20
```

```
sort(x, decreasing =TRUE)
```

```
## [1]  20   7   2   0  -2  -4 -14
```

Basit Vektör İşlemleri - diff()

- Bir vektördeki ardışık elemanlar arasındaki farkı bulmak için `diff()` fonksiyonu kullanılır. `lag` argümanı ile farkı bulunacak terimler arasında kaç sıra olması istendiği belirtilebilir.

```
x <- c(3, 5, 9, 11, 14, 16, 12)
diff(x)
```

```
## [1] 2 4 2 3 2 -4
```

```
diff(x, lag = 2)
```

```
## [1] 6 6 5 5 -2
```


Basit Vektör İşlemleri - setdiff()

- İki vektörden birinde olup diğerinde olmayan elemanları seçmek için `setdiff()` fonksiyonu kullanılır.

```
x <- c(3, 5, 9, 11, 14, 16, 12)
y <- c(1:11)
setdiff(x,y)
```

```
## [1] 14 16 12
```

```
setdiff(y,x)
```

```
## [1] 1 2 4 6 7 8 10
```

Basit Vektör İşlemleri - Karşılaştırma

- İki vektörün terim terime eşit olup olmadığının sınaması yapılacaksa;

```
x <- c(3, 5, 11, 14, 12)
y <- c(3, 7, 11, 15, 12)
x == y
```

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

- Vektörlerin birbirleri ile eşit olup olmadığının sınaması yapılacaksa iki yöntem mevcuttur.

```
all(x == y)
```

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

Basit Vektör İşlemleri - Karşılaştırma

```
identical(x,y)
```

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

Temel İstatistiksel Fonksiyonlar

Fonksiyon	Açıklama
<code>sum(x)</code>	x vektörünün elemanlarının toplamı
<code>mean(x)</code>	ortalama
<code>median(x)</code>	medyan
<code>min(x)</code>	minimum değer
<code>max(x)</code>	maximum değer
<code>sd(x)</code>	standart sapma
<code>var(x)</code>	varyans
<code>cor(x,y)</code>	korelasyon
<code>cov(x,y)</code>	kovaryans
<code>quantile(x,p)</code>	x vektörünün p ile belirtilen yüzdelikleri

Temel İstatistiksel Fonksiyonlar

- Normal dağılıma göre `rnorm()` fonksiyonu ile 1000 adet rassal sayı üretelim ve temel istatistiklerini inceleyelim.

```
x <- rnorm(1000)
length(x); mean(x); sum(x); var(x); sd(x)
```

```
[1] 1000 [1] 0.0301645 [1] 30.1645 [1] 1.01789 [1] 1.00891
```

```
library(PerformanceAnalytics)
skewness(x); kurtosis(x)
```

```
[1] -0.0376149 [1] -0.232699
```

Listeler

- Listeleri vektörlerden ayıran en önemli özellik, farklı türden değişkenleri saklayabilmesidir.

```
x <- list(FALSE, 4, 1 + 2i, "merhaba")  
x
```

```
## [[1]]  
## [1] FALSE  
##  
## [[2]]  
## [1] 4  
##  
## [[3]]  
## [1] 1+2i
```

Listeler

```
##  
## [[4]]  
## [1] "merhaba"
```

```
str(x)
```

```
## List of 4  
## $ : logi FALSE  
## $ : num 4  
## $ : cplx 1+2i  
## $ : chr "merhaba"
```

```
x <- list(a = 40, b = 60, c = 90)  
x$a
```

```
## [1] 40
```

Listeler

```
x$c
```

```
## [1] 90
```

```
x <- list(isim = "Ali", yas = 24, kilo = 72, boy = 180)
```

```
x$yas
```

```
## [1] 24
```

```
x$isim
```

```
## [1] "Ali"
```


Listeler - Vektörden Liste Oluşturma

- Listeler özel bir vektör türüdür. Vektörleri kullanarak liste oluşturmak mümkündür.

```
k <- vector(mode = "list")  
k[["isim"]] <- "Gizem"  
k[["yas"]] <- 24  
k[[3]] <- 50  
k
```

```
## $isim  
## [1] "Gizem"  
##  
## $yas  
## [1] 24
```

Listeler - Vektörden Liste Oluşturma

```
##
```

```
## [[3]]
```

```
## [1] 50
```

```
k[[5]] <- 100
```

```
k[[4]]
```

```
## NULL
```

```
length(k)
```

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

Listeler - Elemanları Seçme

```
x <- 1:4  
y <- c("a","b","c","d")  
z <- c(TRUE,FALSE)
```

```
mylist <- list(x,y,z)  
mylist
```

```
## [[1]]  
## [1] 1 2 3 4  
##  
## [[2]]  
## [1] "a" "b" "c" "d"  
##
```

Listeler - Elemanları Seçme

```
## [[3]]
```

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

- Liste elemanlarına başlık `names()` fonksiyonu ile eklenebilir.

```
names(mylist) <- c("x","y","z")  
mylist$z
```

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

Listeler - Elemanları Seçme

- Liste elemanları birden fazla yöntemle seçilebilir:
 - `mylist$x`
 - `mylist[[1]]`
 - `mylist[["x"]]`
- Eğer `mylist[1]` ile seçim yapılırsa diğer seçimlerden farklı olacağına dikkat ediniz!

```
mylist$x
```

```
## [1] 1 2 3 4
```

```
mylist[[1]]
```

```
## [1] 1 2 3 4
```

Listeler - Elemanları Seçme

```
mylist[["x"]]
```

```
## [1] 1 2 3 4
```

```
mylist[1]
```

```
## $x
```

```
## [1] 1 2 3 4
```

Listeler - Elemanları Seçme

Örnek:

Bir listenin 2.elemanının 4.alt elemanını nasıl seçeriz?

```
x <- list(a = 1:200, b = 2:400, c = c(TRUE,FALSE,FALSE))
```

Listeler - Elemanları Seçme

```
x$b[4]
```

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

```
x[[2]][4]
```

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

```
x[["b"]][4]
```

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


Listeler - Eleman Ekleme

```
x <- list(a = c(92, 88, 50))  
x$b <- c(90,44)  
x
```

```
## $a  
## [1] 92 88 50  
##  
## $b  
## [1] 90 44
```

Matris ve Diziler

- Matrisler, iki boyutlu yani satır ve sütunları olan atomik vektörler olarak tanımlanabilir.

```
mat <- matrix(nrow = 2, ncol = 3)
mat
```

```
##      [,1] [,2] [,3]
## [1,]  NA  NA  NA
## [2,]  NA  NA  NA
```

```
dim(mat)
```

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

Matris ve Diziler - Matris Oluşturma

```
mat <- matrix(c(1:6), nrow = 2, ncol = 3)
mat
```

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

```
mat <- matrix(c(1:6), nrow = 2, ncol = 3, byrow = TRUE)
mat
```

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

Matris ve Diziler - Matris Elemanlarını Seçme

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

```
mat[1,1]; mat[2,2]; mat[1,3]
```

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

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

```
## [1] 3
```

Matris ve Diziler - Matris Elemanlarını Seçme

Soru:

Bir matrisin n.sütunu veya m.satırı nasıl seçilir?

```
mat[2,]    #2.satır
```

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

```
mat[,3]    #3.sütun
```

```
## [1] 3 6
```

Matris ve Diziler - Matris Elemanlarını Seçme

Soru:

3'den başlayıp 20'de bitecek eşit artım miktarlı 4×4 bir matris oluşturunuz ve yalnızca 2.satırı ile 2 ve 3. sütunlarını seçiniz.

Soru:

2'den başlayarak ardışık çift sayıları kullanarak 3×3 boyutlu bir matris oluşturunuz ve 2.satırı hariç geri kalan satırlarını seçiniz.

Soru:

1'den 12'ye kadar sayıları kullanarak 3×4 boyutlu bir matris oluşturunuz ve 4.sütunundaki sayıların 10'dan büyük olup olmadıklarını sınavınız.

Matris ve Diziler - Vektörlerden Matris Oluşturma

- Vektörler kullanarak matris elde etmek için `cbind()` (sütün birleştirme) ve `rbind()` (satır birleştirme) fonksiyonları kullanılmaktadır.

```
x <- 1:5; y <- 10:14  
cbind(x,y)
```

```
##      x  y  
## [1,] 1 10  
## [2,] 2 11  
## [3,] 3 12  
## [4,] 4 13  
## [5,] 5 14
```

Matris ve Diziler - Vektörlerden Matris Oluşturma

```
rbind(x,y)
```

```
##      [,1] [,2] [,3] [,4] [,5]  
## x      1    2    3    4    5  
## y     10   11   12   13   14
```


Matris ve Diziler - Satır ve Sütun İsimlendirme

- Matrislerde (ve daha sonra `data.frame()` de) satır ve sütun isimleri `rownames()` ve `colnames()` fonksiyonları ile verilmektedir.

```
mat <- matrix(4:9, ncol = 3)
rownames(mat) <- c("S1", "S2")
colnames(mat) <- c("K1", "K2", "K3")
```

```
##      K1 K2 K3
## S1   4  6  8
## S2   5  7  9
```

Temel Matris İşlemleri - Diagonal Matris

- Bir köşegen matrisi oluşturmak için `diag(x, nrow, ncol)` komutu kullanılır.

```
mat <- diag(c(2,3), nrow = 2)
```

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

```
mat <- diag(5, nrow = 3, ncol = 4)
```

```
##      [,1] [,2] [,3] [,4]  
## [1,]    5    0    0    0  
## [2,]    0    5    0    0  
## [3,]    0    0    5    0
```

Temel Matris İşlemleri - Transpoze Matris

- Bir matrisin transpozmesini almak için `t()` fonksiyonu kullanılır.

```
mat <- matrix(c(4:12), nrow = 3)
```

```
mat
```

```
##      [,1] [,2] [,3]  
## [1,]    4    7   10  
## [2,]    5    8   11  
## [3,]    6    9   12
```

```
t(mat)
```

Temel Matris İşlemleri - Transpoze Matris

##	[,1]	[,2]	[,3]
## [1,]	4	5	6
## [2,]	7	8	9
## [3,]	10	11	12

Temel Matris İşlemleri

```
x <- matrix(c(4:7), nrow = 2)
y <- matrix(c(1,5,7,11), nrow = 2)
```

```
x+y      # iki matrisin toplamı
```

```
##          [,1] [,2]
## [1,]         5  13
## [2,]        10  18
```

```
x-y      # iki matrisin farkı
```

```
##          [,1] [,2]
## [1,]         3  -1
## [2,]         0  -4
```

Temel Matris İşlemleri

```
2*x    # bir matrisin scaler ile çarpımı
```

```
##      [,1] [,2]  
## [1,]    8  12  
## [2,]   10  14
```

```
x*y    # iki matrisin scaler çarpımı
```

```
##      [,1] [,2]  
## [1,]    4  42  
## [2,]   25  77
```

```
x/y    # iki matrisin scaler bölümü
```

Temel Matris İşlemleri

```
##          [,1]      [,2]  
## [1,]      4 0.857143  
## [2,]      1 0.636364
```

```
x %*% y # iki matrisin çarpımı
```

```
##          [,1] [,2]  
## [1,]      34  94  
## [2,]      40 112
```

Temel Matris İşlemleri - Matrisin Tersi

- Bir matrisler çarpıldığında birim matrisi veren matrise ilk matrisin tersi denir. Matris tersini bulmak için `solve()` fonksiyonu kullanılır.

```
x <- matrix(c(4:7), nrow = 2)
```

```
x
```

```
##      [,1] [,2]
```

```
## [1,]    4    6
```

```
## [2,]    5    7
```

```
solve(x)
```


Temel Matris İşlemleri - Matrisin Tersi

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

```
x %*% solve(x)
```

```
##      [,1]      [,2]  
## [1,]  1 -1.77636e-15  
## [2,]  0  1.00000e+00
```

Temel Matris İşlemleri - Alt ve Üst Köşegen

- Bir matrisin elemanının alt köşegende olduğunun tespiti için `lower.tri(x, diag = FALSE)` ve üst köşegende olduğunun tespiti için ise `upper.tri(x, diag = FALSE)` fonksiyonu kullanılır. Sonuç mantıksal değerdir.

```
x <- matrix(c(4:7), nrow = 2)
```

Örnek:

Verilen x matrisinin üst köşegenini bulunuz.

```
A <- x
```

```
A[lower.tri(A)] = 0
```

Temel Matris İşlemleri - Alt ve Üst Köşegen

##	[,1]	[,2]
## [1,]	4	6
## [2,]	0	7

Soru:

3'den 27'ye kadar olan tek sayılar ile 3x3'lük bir matris oluşturunuz ve alt köşegeni bulunuz.

- Diziler, matrislerin bir üst yapısı olarak düşünülebilir.
Matrislerin aksine diziler n boyuta sahip olabilmektedirler.

```
d1 <- matrix(2, nrow = 3, ncol = 3)
d2 <- matrix(3, nrow = 3, ncol = 3)

dd <- array(data = c(d1,d2), dim=c(3,3,2))

dd[2,3,2]

## [1] 3
```

Data Frame

- Data.frame, R'da en çok kullanılan yapıların başında gelmektedir. Tablo şeklindeki verileri kaydetmek için kullanılır.
- Matrislerde her sütundaki veriler aynı türden olmak zorunda olmasına rağmen data.frame'de bu zorunluluk bulunmamaktadır.
- Data.frame, her elemanı eşit uzunlukta bir liste gibi düşünülebilir.

```
x <- data.frame(not = c(100, 90, 75),  
                isim = c("Ali", "Meltem", "Leyla"))  
x
```

Data Frame

```
##      not      isim
## 1 100      Ali
## 2  90 Meltem
## 3  75  Leyla
```

```
nrow(x)      #sattır sayısı
```

```
## [1] 3
```

```
ncol(x)      #sütun sayısı
```

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

Data Frame - Oluşturma

```
cocuk <- c("A", "B", "C")  
yas <- c(10, 2, 8)  
kilo <- c(50, 20, 40)  
  
x <- data.frame(cocuk, yas, kilo,  
                stringsAsFactors = FALSE)
```

```
##   cocuk yas kilo  
## 1     A  10  50  
## 2     B   2  20  
## 3     C   8  40
```

Data Frame - Eleman Seçimi

```
x[[1]] #data frame aynı zamanda listedir
```

```
## [1] "A" "B" "C"
```

```
x[["cocuk"]]
```

```
## [1] "A" "B" "C"
```

```
x$cocuk
```

```
## [1] "A" "B" "C"
```

```
x[1,] #1.satırını seçer
```

```
##   cocuk yas kilo
```

```
## 1     A  10  50
```


Data Frame - Eleman Seçimi

```
x[,1]    #1.sütunu seçer
```

```
## [1] "A" "B" "C"
```

```
x[2,3]
```

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

Data Frame - subset()

- Yalnızca yaşların listesi

```
subset(x, select = yas)
```

```
##   yas
```

```
## 1  10
```

```
## 2   2
```

```
## 3   8
```

- Çocuk ve kiloların listesi

```
subset(x, select = c(cocuk,kilo))
```

Data Frame - subset()

```
##      çocuk kilo
## 1      A     50
## 2      B     20
## 3      C     40
```

- Yaşı 7'den büyük olan çocukların olduğu kayıtlar

```
subset(x, subset = (yas > 7))
```

```
##      çocuk yas kilo
## 1      A    10   50
## 3      C     8   40
```

- Kilosu 40'dan büyük olan çocukların isimleri ve yaşları

Data Frame - subset()

```
subset(x, select = c(cocuk,yas), subset = (kilo >= 40))
```

```
##      cocuk yas
## 1      A  10
## 3      C   8
```

Data Frame - Veri Ekleme

- Data.frame'e yeni veri ekleme, daha önce gördüğümüz `rbind()` fonksiyonu ile yapılmaktadır.

```
yenikayit <- data.frame(cocuk = "D", yas = 25, kilo = 80)
x <- rbind(x, yenikayit)
x
```

```
##   cocuk yas kilo
## 1     A  10  50
## 2     B   2  20
## 3     C   8  40
## 4     D  25  80
```

Data Frame - Veri Ekleme

- Eğer yeni bir değişken daha eklenmek istenirse daha önce gördüğümüz `cbind()` fonksiyonu kullanılmaktadır.

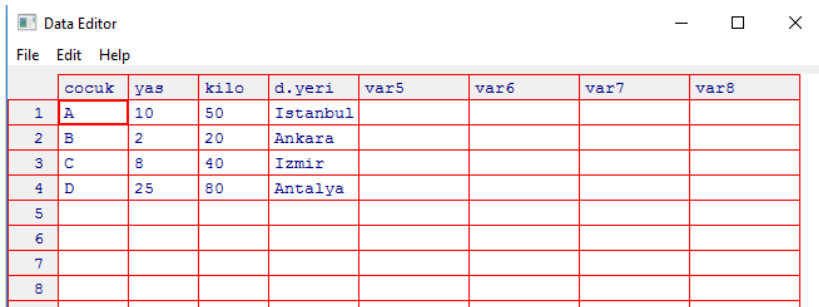
```
x <- cbind(x, data.frame(d.yeri=c("İstanbul", "Ankara",  
                                "İzmir", "Antalya")))
```

x

```
##   cocuk yas kilo   d.yeri
## 1     A  10   50 Istanbul
## 2     B   2   20   Ankara
## 3     C   8   40    Izmir
## 4     D  25   80  Antalya
```

Data Frame - edit()

- Data.frame'e RStudio üzerinden basit bir data editör ile düzenlenebilir. `edit(x)` fonksiyonu ile yapılmaktadır.



The screenshot shows the RStudio Data Editor window. The title bar says "Data Editor". The menu bar has "File", "Edit", and "Help". The data frame has 9 columns: "cocuk", "yas", "kilo", "d.yeri", "var5", "var6", "var7", and "var8". The first row is highlighted with a red border.

	cocuk	yas	kilo	d.yeri	var5	var6	var7	var8
1	A	10	50	Istanbul				
2	B	2	20	Ankara				
3	C	8	40	Izmir				
4	D	25	80	Antalya				
5								
6								
7								
8								

Data Frame - Eksik Verileri Temizleme

- Bir data.frame'de eksik **NA** veriler mevcut ise bunları temizlemek için `na.omit()` veya `na.exclude()` fonksiyonları kullanılabilir.
- Bu fonksiyonlara finansal zaman serileri kullanılarak yapılan analizlerde sıklıkla başvurulmaktadır.

Data Frame - merge()

- Farklı iki data.frame'i ortak sütun kullanarak birleştirmek için `merge(df1, df2, by = "ortaksütun")` fonksiyonu kullanılmaktadır.

```
y <- data.frame(cocuk = c("A", "B", "C", "D"),  
                boy = c(170, 100, 150, 175))
```

```
merge(x, y, by = "cocuk")
```

```
##   cocuk yas kilo   d.yeri boy  
## 1     A  10   50 Istanbul 170  
## 2     B   2   20   Ankara 100  
## 3     C   8   40    Izmir 150  
## 4     D  25   80  Antalya 175
```

Metin İşlemleri - paste()

- Farklı metinleri birleştirerek tek bir metin oluşturmak için `paste()` fonksiyonu kullanılır.

```
paste("R", "ve", "Temel", "İstatistik", sep="-")
```

```
## [1] "R-ve-Temel-Istatistik"
```

- `paste0()` fonksiyonu metinler arasında bir boşluk bırakmadan birleştirir.

```
paste0("Risk", "Active")
```

```
## [1] "RiskActive"
```

Metin İşlemleri - nchar()

- `nchar()` fonksiyonu bir metnin uzunluğunu bulmak için kullanılır.

```
nchar("R'a Giriş ve Temel İstatistik")
```

```
## [1] 29
```

Metin İşlemleri - strsplit()

- Metni belirli bir karakter ya da metne göre parçalara bölmek için `strsplit()` fonksiyonu kullanılır.

```
strsplit("İstanbul-Ankara-İzmir-Antalya", split = "-")
```

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

```
## [1] "Istanbul" "Ankara"    "Izmir"     "Antalya"
```

Metin İşlemleri - regexpr()

- Bir metin içerisinde aranan bir alt metnin yerini bulmak için `regexpr()` fonksiyonu kullanılır. `ignore.case` ile büyük harf duyarlılığı iptal edilebilir.

```
k <- regexpr(pattern = "mi", text = "İzmir")
```

```
## [1] 3  
## attr(,"match.length")  
## [1] 2  
## attr(,"index.type")  
## [1] "chars"  
## attr(,"useBytes")  
## [1] TRUE
```

Metin İşlemleri - regexpr()

```
k[1]
```

```
## [1] 3
```

```
attributes(k)$match.length
```

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

Metin İşlemleri - `grep()` ve `grepl()`

- Bir metin içerisinde aranan bir ifadenin geçip geçmediğini bulmak için `grep()` veya `grepl()` fonksiyonları kullanılır. `ignore.case = TRUE` ile büyük harf duyarlılığı iptal edilebilir.
- `grep()` fonksiyonu, eşleşen koşulların sıra numarasını döndürmektedir.
- `grepl()` fonksiyonu, tüm koşulların sınaama sonuçlarını döndürmektedir.

Metin İşlemleri - grep() ve grepl()

```
k <- c("R Programlama",  
      "Programlama",  
      "R",  
      "r ile istatistiksel analiz")
```

```
grep("R", k)
```

```
## [1] 1 3
```

```
grepl("r", k, ignore.case = TRUE)
```

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


Metin İşlemleri - Metin Kombinasyonları Oluşturma

- İki farklı metin vektörünü kullanarak çaprazlama tüm olası kombinasyonları `outer()` fonksiyonu ile yapılmaktadır.

```
metin1 <- c("kurumsal","ticari")
```

```
metin2 <- c("kk","kredi","kmh")
```

```
outer(metin1, metin2, paste, sep = "/")
```

```
##           [,1]           [,2]           [,3]
## [1,] "kurumsal/kk" "kurumsal/kredi" "kurumsal/kmh"
## [2,] "ticari/kk"   "ticari/kredi"   "ticari/kmh"
```

Programlama Dili Olarak R

- RStudio içerisinde sağ üst köşedeki panel içerisinde Environment/Import Dataset menüsü kullanılarak aşağıdaki formatlardaki dataset'ler içeri alınabilmektedir.
 - Metin dosyaları (csv, txt)
 - Excel dosyaları (xls, xlsx)
 - SPSS
 - SAS
 - Stata
- Metin dosyaları (**From Text**) seçeneğinin iki farklı kütüphane ile içeri alındığına dikkat ediniz.
 - **base**
 - **readr**

Veri Alışverişi - Base

Import Dataset

Name

capm

Encoding

Automatic

Heading

☒ Yes ☐ No

Row names

Automatic

Separator

Comma

Decimal

Period

Quote

Double quote (")

Comment

None

na.strings

NA

☒ Strings as factors

Input File

Date, IBM, MARKET, RKFREE
Jan-78,-0.029,-0.045,0.00487
Feb-78,-0.043,0.01,0.00494
Mar-78,-0.063,0.05,0.00526
Apr-78,0.13,0.063,0.00491
May-78,-0.018,0.067,0.00513
Jun-78,-0.004,0.007,0.00527
Jul-78,0.092,0.071,0.00528
Aug-78,0.049,0.079,0.00607
Sep-78,-0.051,0.002,0.00645
Oct-78,-0.046,-0.189,0.00685
Nov-78,0.031,0.084,0.00719
Dec-78,0.108,0.015,0.00690
Jan-79,0.034,0.058,0.00761
Feb-79,-0.017,0.011,0.00761
Mar-79,0.052,0.123,0.00769
Apr-79,-0.004,0.026,0.00764
May-79,-0.022,0.014,0.00772
Jun-79,-0.025,0.075,0.00715

Data Frame

Date	IBM	MARKET	RKFREE
Jan-78	-0.029	-0.045	0.00487
Feb-78	-0.043	0.010	0.00494
Mar-78	-0.063	0.050	0.00526
Apr-78	0.130	0.063	0.00491
May-78	-0.018	0.067	0.00513
Jun-78	-0.004	0.007	0.00527
Jul-78	0.092	0.071	0.00528
Aug-78	0.049	0.079	0.00607
Sep-78	-0.051	0.002	0.00645
Oct-78	-0.046	-0.189	0.00685
Nov-78	0.031	0.084	0.00719
Dec-78	0.108	0.015	0.00690
Jan-79	0.034	0.058	0.00761
Feb-79	-0.017	0.011	0.00761
Mar-79	0.052	0.123	0.00769
Apr-79	-0.004	0.026	0.00764
May-79	-0.022	0.014	0.00772
Jun-79	-0.025	0.075	0.00715

Import

Cancel

Veri Alışverişi - Base

```
path <- "D:/RiskActive/RiskActive_Learning/  
        TBB/introduction_to_R/data/capm.csv"
```

```
capm <- read.csv(path, header = TRUE)
```

```
str(capm)
```

```
## 'data.frame':    120 obs. of  4 variables:  
## $ Date   : Factor w/ 120 levels "Apr-78","Apr-79",...: 41  
## $ IBM    : num  -0.029 -0.043 -0.063 0.13 -0.018 -0.004  
## $ MARKET: num  -0.045 0.01 0.05 0.063 0.067 0.007 0.071  
## $ RKFREE: num   0.00487 0.00494 0.00526 0.00491 0.00513
```

Veri Alışverişi - Base

- İçeri alınan data özellikleri incelendiğinde **Date** kolonunun **factor** olduğu görülmektedir.
- Ancak bu kolonda metin bilgisi saklandığı için veri tipinin **character** olması bekleniyordu.
- **Base** kütüphane kullanılarak içeri alınan verilerde **stringsAsFactors = TRUE** ön tanımlı ayardır. Bu nedenle içeri alınan tüm metinler **factor** değişkenine dönüştürülmektedir.

```
capm <- read.csv(path,  
                  header = TRUE,  
                  stringsAsFactors = FALSE)
```

Veri Alışverişi - Readr

Import Text Data

File/URL
D:\RiskActive\RiskActive_Learning\TBB\introduction_to_R\data\caps.csv Browse...

Data Preview:

Date (datevector)	IBM (double)	MARKET (double)	RNFRFEE (double)
Jan-78	-0.029	-0.048	0.00487
Feb-78	-0.043	0.010	0.00494
Mar-78	-0.063	0.050	0.00526
Apr-78	0.130	0.063	0.00491
May-78	-0.018	0.067	0.00513
Jun-78	-0.004	0.007	0.00527
Jul-78	0.092	0.071	0.00528
Aug-78	0.049	0.079	0.00607
Sep-78	-0.051	0.002	0.00645
Oct-78	-0.046	-0.189	0.00685
Nov-78	0.031	0.084	0.00719
Dec-78	0.108	0.015	0.00690
Jan-79	0.034	0.058	0.00761
Feb-79	-0.017	0.011	0.00761
Mar-79	0.052	0.123	0.00769
Apr-79	-0.004	0.026	0.00764
May-79	-0.022	0.014	0.00772
Jun-79	-0.035	0.075	0.00715
Jul-79	-0.049	-0.013	0.00728
Aug-79	0.016	0.095	0.00789
Sep-79	-0.032	0.039	0.00802
Oct-79	-0.079	-0.087	0.00913

Processing first 50 entries.

Import Options

Name: caps ☐ First Row as Names
Skip: 0 ☒ Trim Spaces
☒ Open Data Viewer

Delimiter: Comma
Quotes: Default
Locale: Configure...

Escape: None
Comment: Default
NA: Default

Code Preview

```
library(readr)  
caps <- read_csv("D:/RiskActive/RiskActive_Learning/TBB/introduction_to_R/data/caps.csv")  
view(caps)
```

Reading rectangular data using readr Import Cancel

Veri Alışverişi - Readr

```
capm <- readr::read_csv(path,  
                          col_names = T)  
  
str(capm)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    120 obs. of  
## $ Date   : chr  "Jan-78" "Feb-78" "Mar-78" "Apr-78" ...  
## $ IBM    : num  -0.029 -0.043 -0.063 0.13 -0.018 -0.004  
## $ MARKET: num  -0.045 0.01 0.05 0.063 0.067 0.007 0.071  
## $ RKFREE: num   0.00487 0.00494 0.00526 0.00491 0.00513  
## - attr(*, "spec")=List of 2  
## ..$ cols   :List of 4  
## .. ..$ Date : list()  
## .. ..- attr(*, "class")= chr  "collector_character"
```



```
## .. ..$ IBM : list()
## .. .. ..- attr(*, "class")= chr "collector_double" "c
## .. ..$ MARKET: list()
## .. .. ..- attr(*, "class")= chr "collector_double" "c
## .. ..$ RKFREE: list()
## .. .. ..- attr(*, "class")= chr "collector_double" "c
## .. ..$ default: list()
## .. .. ..- attr(*, "class")= chr "collector_guess" "colle
## .. ..- attr(*, "class")= chr "col_spec"
```

- İçeri alınacak dosyanın bilgisayarda kayıtlı olma şartı bulunmamaktadır.

```
url <- "https://raw.githubusercontent.com/toygur/  
      REgitimMerkezi/master/dataset/mushrooms.csv"  
  
mushrooms <- readr::read_csv(url)
```

Veri Alışverişi - read.table()

- `read.table()` ile hem numeric hem de character değerler alınabilmektedir. Read.table sonucu data.frame yapısı oluşmaktadır.

```
h1 <- read.table("data/deneme1.txt", sep = ",",  
                 dec = ".", na.strings = "NA",  
                 header = T)
```

h1

```
##   age  name wage  
## 1  12 bobby 24.5  
## 2  24  kate 36.3  
## 3  NA james 42.0  
## 4  35 david 18.5
```

Veri Alışverişi - read.table()

```
str(h1)
```

```
## 'data.frame':    4 obs. of  3 variables:
```

```
## $ age : int  12 24 NA 35
```

```
## $ name: Factor w/ 4 levels "bobby","david",...: 1 4 3 2
```

```
## $ wage: num  24.5 36.3 42 18.5
```

- Character değişkenleri Factor değişkeni olarak almaması için `as.is = T` argümanı kullanılmalıdır.

```
h2 <- read.table("data/deneme1.txt", sep = "",  
                dec = ".", na.strings = "NA",  
                as.is = T, header = T)
```

```
str(h2)
```

Veri Alışverişi - read.table()

```
## 'data.frame':    4 obs. of  3 variables:  
##  $ age : int  12 24 NA 35  
##  $ name: chr  "bobby" "kate" "james" "david"  
##  $ wage: num  24.5 36.3 42 18.5
```

Veri Alışverişi - write.table()

- Data.frame yapısını .txt veya .csv uzantılı dosyalara yazmak için `write.table()` fonksiyonu kullanılır.

```
write.table(h2,  
            file="data/create_csv.csv",  
            append=FALSE,  
            sep="," ,  
            dec = ".",  
            na = "NA",  
            col.names=TRUE,  
            row.names = FALSE)
```

Veri Alışverişi - `install.packages("foreign")`

Fonksiyon	Açıklama
<code>read.dta</code>	Read Stata binary file
<code>read.spss</code>	Read an SPSS data file
<code>read.xport</code>	Read a SAS XPORT format library
<code>lookup.xport</code>	Lookup information on a SAS Export format library
<code>write.dbf</code>	Write a DBF file
<code>write.dta</code>	Write files in Stata binary format
<code>read.dbf</code>	Read a DBF file
<code>data.restore</code>	Read an S3 binary file
<code>read.octave</code>	Read Octave text data files
<code>write.foreign</code>	Write text files and code to read them

Veri Alışverişi - `install.packages("quantmod")`

- `quantmod` library'si ile Yahoo Finance veya Google Finance'dan tarihsel veri alınabilmektedir.
- Gelen tarihsel veri `xts` formatındadır.
- `head(data)` fonksiyonu ile tarihsel verinin başlangıcından `tail(data)` fonksiyonu ile de tarihsel verinin sonundan örnek görülebilmektedir.
- `getSymbols()` fonksiyonu ile hisse senedi, index ... vb. dataların açılış, kapanış, en yüksek, en düşük ve hacim gibi değerleri zaman serisi olarak gelmektedir.

Veri Alışverişi - `install.packages("quantmod")`

```
library(quantmod)
aapl <- getSymbols("AAPL",
                  src="yahoo",
                  auto.assign = FALSE,
                  from = "2015-01-01",
                  to = "2016-01-01")

str(aapl)
```

```
## An 'xts' object on 2015-01-02/2015-12-31 containing:
##   Data: num [1:252, 1:6] 111 108 107 107 109 ...
##   - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "AAPL.Open" "AAPL.High" "AAPL.Low" "AAPL.Close"
```

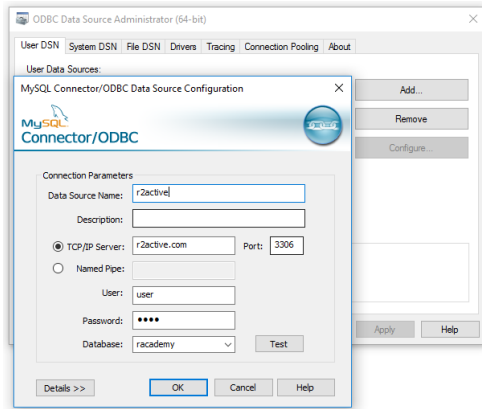
Veri Alışverişi - `install.packages("quantmod")`

```
## Indexed by objects of class: [Date] TZ: UTC
## xts Attributes:
## List of 2
## $ src      : chr "yahoo"
## $ updated: POSIXct[1:1], format: "2018-07-20 08:36:48"
```

- Yahoo Finance için bazı örnek tickerlar: `"^GSPC"` - SP500, `"^FCHI"` - CAC40, `"^DJI"` - Dow Jones, `"^IXIC"` - NASDAQ Composite, `"^N225"` - Nikkei 225

- `odbc`, `RODBC` veya `RMYSQL` library'leri ile MySQL veri tabanı bağlantısı yapılabilmektedir.
- ODBC aracılığı ile yapılan bağlantılar için ODBC Connector gereklidir. Ücretsiz olan MySQL ODBC Connector <https://dev.mysql.com/downloads/connector/odbc/> adresinden indirilebilir.

Veri Alışverişi - MySQL



- Veritabanı bağlantısı için **odbc** library'si içindeki **dbConnect()** fonksiyonu kullanılmaktadır.

Veri Alışverişi - MySQL

```
library(odbc)
```

```
conn <- dbConnect(odbc::odbc(), "r2active")
```

- Veritabanındaki tablolar `dbListTables()` fonksiyonu ile listelenmektedir.

```
dbListTables(conn)
```

```
## [1] "currency" "interest" "parity"
```

- Belirli bir harf ile başlayan tablolar şu şekilde listelenmektedir.

```
dbListTables(conn, table_name = "c%")
```

```
## [1] "currency"
```

- Bir tabloda geçerli alanları listelemek için `dbListFields()` fonksiyonu kullanılmaktadır.

```
dbListFields(conn, "currency")
```

```
## [1] "id"                "asset_code"        "currency_code"
```

- `dbSendQuery()` fonksiyonu ile istenilen SQL sorgusu çalıştırılabilmektedir. Veritabanından dönen sonucu `data.frame` formatına çevirmek için `dbFetch()` fonksiyonu kullanılmaktadır.

Veri Alışverişi - MySQL

```
h4_result <- dbSendQuery(conn,
                          paste0("SELECT * FROM parity
                                  WHERE asset_code ='USD/TRY'");")
h4 <- dbFetch(h4_result)
head(h4)
```

##	id	data_date	asset_code	open	high	low	close
## 1	1	2015-12-31	USD/TRY	2.9188	2.9308	2.9081	2.9160
## 2	6	2015-12-30	USD/TRY	2.9064	2.9294	2.9050	2.9262
## 3	9	2015-12-29	USD/TRY	2.9061	2.9149	2.9041	2.9056
## 4	10	2015-12-28	USD/TRY	2.9220	2.9259	2.9066	2.9069
## 5	13	2015-12-25	USD/TRY	2.9187	2.9409	2.8978	2.9088
## 6	17	2015-12-24	USD/TRY	2.9170	2.9272	2.9156	2.9169

Soru:

Veritabanından 2015-02-01 ile 2015-03-01 tarihleri arasında EUR/USD paritesinin kapanış fiyatını çekelim ve paritenin currency kodunu parite datasına ekleyelim.

Veri Alışverişi - MySQL Veri Birleştirme

```
h5_res <- dbSendQuery(conn,
  paste0("SELECT * FROM parity
        WHERE asset_code = 'EUR/USD' and
        data_date >= '2015-02-01' and
        data_date <= '2015-03-01';"))
h5 <- dbFetch(h5_res)

h6_res <- dbSendQuery(conn,
  paste0("SELECT * FROM currency
        WHERE asset_code = 'EUR/USD'")
)
h6 <- dbFetch(h6_res)

h7 <- merge(h5, h6, by = "asset_code")
```

Kontrol Yapıları / If-Else

- Bir koşulun test edilmesi için kullanılır.

```
if ( koşul ) {  
    ## koşul doğru ise işlem yap  
} else {  
    ## koşul yanlış ise işlem yap  
}
```

- Birden fazla koşulun test edileceği durumlarda aşağıdaki yapı kullanılır.

Kontrol Yapıları / If-Else

```
if ( koşul_1 ) {  
    ## koşul_1 doğru ise işlem yap  
} else if ( koşul_2 ) {  
    ## koşul_2 doğru ise işlem yap  
} else {  
    ## diğer durumlarda işlem yap  
}
```

Örnek:

x , 10'dan büyük ise 2^x , küçük ise $2 * x$ ve eşit ise x değerini veren bir if-else bloğu oluşturalım ve $x = 7$ için test edelim.

Kontrol Yapıları / If-Else

```
x <- 7
if (x > 10) {
  sonuc <- 2^x
} else if (x == 10) {
  sonuc <- x
} else {
  sonuc <- 2*x
}
sonuc
```

```
## [1] 14
```

Kontrol Yapıları / For

- “for” döngüleri bir iç değişkene yine bir iç vektör ya da dizideki değerlere sırasıyla atayarak tekrarlanan komutlardan oluşur.

```
x <- c(1:3)
for (i in 1:length(x)){
  print(x[i])
}
```

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

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

```
## [1] 3
```

Kontrol Yapıları / For

```
gun<- c("pzt","sal","çarş","pers", "cum")  
for (i in seq_along(gun)){  
  print(gun[i])  
}
```

```
## [1] "pzt"  
## [1] "sal"  
## [1] "çarş"  
## [1] "pers"  
## [1] "cum"
```

Kontrol Yapıları / For

```
gun<- c("pzt","sal","çarş","perş", "cum")  
for (i in gun){  
  print(i)  
}
```

```
## [1] "pzt"  
## [1] "sal"  
## [1] "çarş"  
## [1] "perş"  
## [1] "cum"
```

Örnek:

$n = 100$ adet gözlemden oluşan ($\mu = 0$, $\sigma = 1$) 5 adet rasgele seri oluşturarak, serilerin birbirleri ile olan korelasyonunu hesaplayıp matris formatında ekrana yazdıran bir yapı oluşturalım.

- Çözüm metodolojimiz şu şekildedir:
 - Öncelikle **data** adında 100x5 matris oluşturulacaktır.
 - For döngüsü ile matrisin her sütununa **rnorm()** kullanılarak data oluşturulacaktır.
 - Korelasyon sonuçlarını yazdırmak için 5x5 boyutunda **sonuc** adında bir matris oluşturulacaktır.
 - İç içe oluşturulacak iki adet for yapısı ile korelasyon hesaplanacak ve **sonuc** matrisinde ilgili hücreye yazdırılacaktır.

Kontrol Yapıları / For

```
n <- 100          #gözlem sayısı
datanum <- 5       #oluşturulacak seri sayısı
data <- matrix(c(0:0), nrow=n, ncol=datanum)
for (i in 1:datanum) {
  data[,i] <- rnorm(n=n, mean=0, sd=1)
}
sonuc <- matrix(c(0:0), nrow=datanum, ncol=datanum)
for (i in 1:datanum) {
  for (j in 1:datanum) {
    sonuc[i,j] <- cor(data[,i],data[,j])
  }
}
```

Kontrol Yapıları / For

```
options(digits = 2)
sonuc
```

##		[,1]	[,2]	[,3]	[,4]	[,5]
##	[1,]	1.00000	-0.00062	0.075	0.034	0.094
##	[2,]	-0.00062	1.00000	0.031	-0.096	0.038
##	[3,]	0.07480	0.03125	1.000	-0.171	0.031
##	[4,]	0.03414	-0.09626	-0.171	1.000	-0.092
##	[5,]	0.09356	0.03768	0.031	-0.092	1.000

Kontrol Yapıları / While

- “while” döngüleri koşul doğru olduğu sürece devam etmektedir.

```
x <- c(1:3)
i <- 1
while ( i <= length(x)){
  print(x[i])
  i <- i + 1
}
```

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

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

```
## [1] 3
```

Kontrol Yapıları / While

Örnek:

x , 10'dan büyük ise 2^x , küçük ise $2 * x$ ve eşit ise x değerini veren if-else bloğunu daha önce oluşturmuştuk. If bloğunu, while döngüsü ile 8 ile 12 arasındaki sayılar için test edelim.

```
x <- c(8:12)
sonuc <- vector(mode="numeric", length = length(x))
i <- 1
```

Kontrol Yapıları / While

```
while (i <= length(x)) {  
  if (x[i] > 10) {  
    sonuc[i] <- 2^x[i]  
  } else if (x[i] == 10) {  
    sonuc[i] <- x[i]  
  } else {  
    sonuc[i] <- 2*x[i]  
  }  
  i <- i + 1  
}  
sonuc
```

```
## [1]    16    18    10 2048 4096
```

Kontrol Yapıları / Repeat

- “repeat” sonsuz döngüdür, “break” ile döngüden çıkılır.

```
a <- 0
repeat {
  a <- a + 1
  print(a)
  if(a == 3) break()
}
```

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

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

```
## [1] 3
```

Fonksiyonlar

- Fonksiyonlar, sık tekrarlanan işlemler için gerekli kodları her defasında tekrar yazmak yerine oluşturulan bir yapı ile zamandan ve iş gücünden tasarruf etmemizi sağlayan yapılardır.

```
f <- function (argümanlar ) {  
  ## İşlemler  
}
```

Örnek:

Argüman olarak bir vektör alıp ortalama hesaplayan fonksiyon oluşturalım.

Fonksiyonlar

```
UserMean <- function (x) {  
  result <- sum(x) / length(x)  
  return(result)  
}
```

```
x <- rnorm(1000)  
UserMean(x); mean(x)
```

```
## [1] 0.0037
```

```
## [1] 0.0037
```


Soru:

Argüman olarak bir vektör alıp tanımlayıcı istatistiklerini (uzunluk, minimum, maksimum, medyan, ortalama, standart sapma, varyans, skewness, kurtosis) hesaplayan, sonuçları data.frame'e yazdıran bir fonksiyon oluşturunuz.

Fonksiyonlar

```
UserStat <- function (x) {  
  
  result <- data.frame(length = length(x),  
                        median = median(x),  
                        min = min(x),  
                        max = max(x),  
                        mean = mean(x),  
                        sd = sd(x),  
                        var = var(x),  
                        skewness = PerformanceAnalytics::skewness(x),  
                        kurtosis = PerformanceAnalytics::kurtosis(x))  
  return(result)  
}
```

```
UserStat(x)
```

```
##      length median   min max    mean    sd   var skewness kurtosis
## 1      1000 0.0056 -2.8 3.4 0.0037 0.99 0.98 -0.0093 0.0000
```

Fonksiyonlar

Soru:

Kullanıcın, tickerList, başlangıç, bitiş tarihleri ("YYYY-MM-DD") ve fiyat tipini ("Open", "High", "Low", "Close" veya "AdjClose") seçtiği; ve bu kıstaslara göre Yahoo servisine bağlanarak (quantmod) dataların çekilip ve ortak tarihlere göre datayı merge ettiği bir fonksiyon oluşturunuz.

```
tickerList <- c("^GSPC", "MSFT",  
               "SBUX", "XOM", "AMZN")  
data <- getYahoo(tickerList,  
                 from = "2015-01-01",  
                 end = "2016-01-01",  
                 pricetype="AdjClose")
```

```
head(data)
```

##		^GSPC	MSFT	SBUX	XOM	AMZN
##	2015-01-02	2058	43	38	82	309
##	2015-01-05	2021	43	38	80	302
##	2015-01-06	2003	42	37	79	295
##	2015-01-07	2026	42	38	80	298
##	2015-01-08	2062	44	39	81	300
##	2015-01-09	2045	43	38	81	297

apply()

- `apply()` fonksiyonu, bir matris ya da `data.frame`'in tüm satır veya sütunlarına aynı fonksiyonu uygulamak için kullanılır.

```
x <- data.frame(a = rnorm(6),  
                b = rnorm(6))
```

#her bir satıra sum fonksiyonunu uygular.

```
apply(x, 1, sum)
```

```
## [1] 0.58 -0.55 0.10 -0.48 -0.29 1.40
```

```
data <- readr::read_csv("data/capm.csv")
```

apply()

```
## Parsed with column specification:  
## cols(  
##   Date = col_character(),  
##   IBM = col_double(),  
##   MARKET = col_double(),  
##   RKFREE = col_double()  
## )  
  
head(data)
```

apply()

```
## # A tibble: 6 x 4
##   Date      IBM MARKET  RKFREE
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 Jan-78 -0.029 -0.045  0.00487
## 2 Feb-78 -0.043  0.01   0.00494
## 3 Mar-78 -0.063  0.05   0.00526
## 4 Apr-78  0.13   0.063  0.00491
## 5 May-78 -0.018  0.067  0.00513
## 6 Jun-78 -0.004  0.007  0.00527
```

```
apply(data[-1], 2, mean)
```

```
##      IBM MARKET RKFREE
## 0.0096 0.0140 0.0068
```


apply()

```
apply(data[-1], 2, sd)
```

```
##      IBM MARKET RKFREE
```

```
## 0.0590 0.0684 0.0022
```

Fonksiyonlar & apply()

Soru:

Data.frame formatında bir değişkenin tüm sütunları için tanımlayıcı istatistikleri (uzunluk, minimum, maksimum, medyan, ortalama, standart sapma, varyans, skewness, kurtosis) hesap edip sonucu bir data.frame dosyasına yazdıran bir fonksiyon oluşturunuz ve capm.csv dosyası için test ediniz.

Fonksiyonlar & apply()

```
UserAllstats <- function(data) {  
  
  testlist <- c("Uzunluk", "Minimum", "Maksimum",  
               "Medyan", "Ortalama", "Std.S.",  
               "Varyans", "Skewness", "Kurtosis")  
  
  sonuc <- data.frame(matrix(NA,  
                             nrow = length(testlist),  
                             ncol = ncol(data)))  
  
  colnames(sonuc) <- colnames(data)  
  rownames(sonuc) <- testlist
```

Fonksiyonlar & apply()

```
sonuc[1,] <- apply(data, 2, length)
sonuc[2,] <- apply(data, 2, min)
sonuc[3,] <- apply(data, 2, max)
sonuc[4,] <- apply(data, 2, median)
sonuc[5,] <- apply(data, 2, mean)
sonuc[6,] <- apply(data, 2, sd)
sonuc[7,] <- apply(data, 2, var)
sonuc[8,] <- apply(data, 2,
                    PerformanceAnalytics::skewness)
sonuc[9,] <- apply(data, 2,
                    PerformanceAnalytics::kurtosis)
return(sonuc)
}
```

Fonksiyonlar & apply()

```
UserAllstats(data[, -1])
```

##	IBM	MARKET	RKFREE
## Uzunluk	120.0000	120.0000	1.2e+02
## Minimum	-0.1870	-0.2600	2.1e-03
## Maksimum	0.1500	0.1480	1.3e-02
## Medyan	0.0020	0.0120	6.6e-03
## Ortalama	0.0096	0.0140	6.8e-03
## Std.S.	0.0590	0.0684	2.2e-03
## Varyans	0.0035	0.0047	4.8e-06
## Skewness	-0.0366	-1.1092	5.4e-01
## Kurtosis	0.1529	3.0022	-1.6e-01

Kategorik Değişkenler (Faktörler) ile İşlemler

- Kategorik değişkenler R'da bir vektör uzantısı olan faktör (factor) olarak saklanmaktadır.
- Faktör vektörleri `factor()` fonksiyonu ile üretilir ve dummy olarak tanımlı vektörlerdir.
- Tipik olarak faktörler ekonometrik uygulamalarda karşımıza, cinsiyet, ülke, kur olarak çıkmaktadır.

```
g <- rep(0:1, c(2,4))  
g <- factor(g, levels = 0:1,  
            labels = c("male", "female"))
```

```
## [1] male   male   female female female female  
## Levels: male female
```

tidyverse: Data Manipülasyonu

- **tidyverse**, içerisinde çeşitli paketler bulunduran ve bu paketleri tek bir seferde indirip çalışma ortamımızda kullanabilmek için tasarlanmış bir dış pakettir.
- Bu alt paketlerin tümü **install.packages("tidyverse")** komutu çalıştırılarak kurulmaktadır.
- **tidyverse** paketinin aktif hale getirilmesiyle bazı paketler otomatik olarak aktif hale getirilir ve bazı fonksiyonlarda çakışmalar meydana gelebilir.


```
library(tidyverse)
```

```
## -- Attaching packages -----
```

```
## v ggplot2 3.0.0      v purrr    0.2.5
```

```
## v tibble  1.4.2      v dplyr    0.7.6
```

```
## v tidyr   0.8.1      v stringr  1.3.1
```

```
## v readr   1.1.1      v forcats  0.3.0
```

```
## -- Conflicts -----
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::first()  masks xts::first()
```

```
## x dplyr::lag()    masks stats::lag()
```

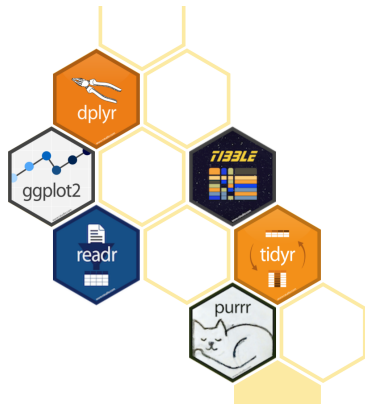
```
## x dplyr::last()   masks xts::last()
```

tidyverse - Pipeline

- **tidyverse** paketlerini incelemeye başlamadan önce, sol taraftaki işlevi sağ tarafa aktararak işlevler arası bağlantı kurmayı sağlayan **%>%** operatörünü incelemeliyiz.
- **Pipeline (%>%)** operatörünün **tidyverse** paket fonksiyonları ile kullanılması, ilişki kurmak için yazılacak ara satırlardan tasarruf edilmesini ve fonksiyon bütünlüğünün korunmasını sağlar.
- Bu operatör **magrittr** paketinin içinde yer alır.

tidyverse Paketleri

tidyverse dış
kütüphanesi(paketi), güçlü veri
manipülasyonu için geliştirilmiş
çeşitli paketler içerir.



tidyverse Paketleri

- **dplyr** paketi, veri manipülasyonu için ileri düzey çözümler sunar. Veri setleri ile doğrudan çalışabilmek üzere, işlevleri paralel olan **plyr** paketinden esinlenilerek yaratılmıştır. Hesaplamanın büyük bir kısmını C++'a taşıdığından zaman sorununa çözüm getirmiştir.
- **tidyr** paketi, veriyi düzenleyerek analize uygun şekilde anlamlandırmaya yardımcı olur. Verinin istenilen düzenli forma getirilmesiyle veriyi manipüle etme, görselleştirme ve modelleme süreci kolaylaşır.
- **readr** paketi, csv, tsv, fwf gibi dosya tiplerindeki veriyi çözümleyerek, uygun sütun ve satırlara yerleşmiş veri seti halinde okur.

tidyverse Paketleri

- **purrr** paketi, fonksiyonlar ile vektörlerin birlikte çalışması için bir takım araçlar sağlayarak, R'ın işlevsel(fonksiyonel) programlama araç setini geliştirir.
- **tibble** paketi, geleneksel veri seti yapısından daha iyi kontrol ve baskı özellikleri olan **tbl_df** tipinde veri setleriyle çalışmak için tasarlanmıştır.
- **ggplot2** paketi, grafiksel dilbilgisi tabanlı olup hem tek değişkenli hem de çok değişkenli sayısal ve kategorik verileri basit bir şekilde temsil eden grafikler oluşturulmasını sağlar.
- **stringr** paketi, karakter(string) veri tipleri üzerinde çeşitli manipülasyonlar yapmayı sağlayan bir dizi işlev sunar.

- **forecast** paketi, kategorik verileri temsil edebilen faktör değişkeni ile çözümler geliştiren bir takım fonksiyonlar sunar.

İçe Aktarım Paketleri

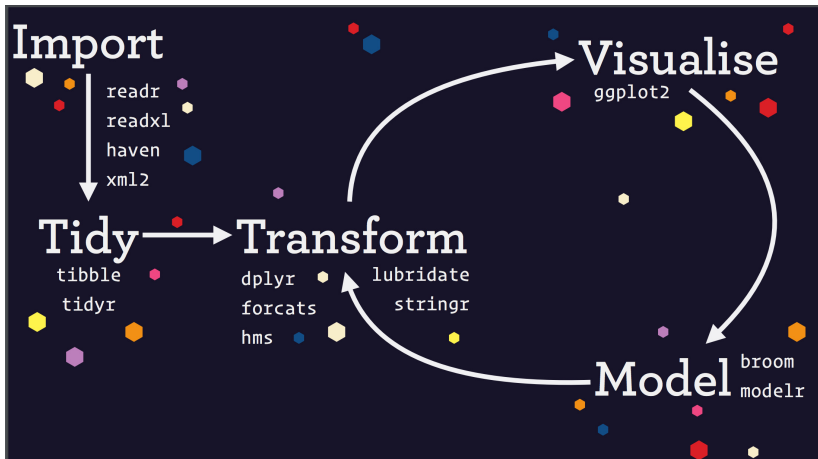
readr paketi gibi çeşitli dosya tipleri için özelleşmiş dosya okuma paketleri aşağıda listelenmiştir.

- .xls ve .xlsx dosyaları - **readxl**
- SPSS, Stata, ve SAS verileri - **haven**
- JSON - **jsonlite**
- XML - **xml2**
- web APIs - **httr**
- web scraping - **rvest**
- ilişkili veri tabanları - **DBI**

Veri Tipleri Özelinde Çalışma

- **lubridate** paketi, tarih-zaman verilerini ayrıştırıp düzenlemek için gelişmiş işlevler sunar.
- **hms** paketi, günlük zaman değerlerini saklamak ve ss:dd:ss formatında görüntüleme işlevleri için kullanılır.
- **blob** paketi, binary tipteki verileri saklamak için kullanılan işlevleri kapsar.

tidyverse Akış Şeması



dplyr'a Giriş

- **dplyr** kütüphanesi, özel bir data.frame yapısı olup data manipülasyonun grammar'ini olarak görülmektedir.
- **dplyr**, 5 ana fiilden oluşmaktadır.
 - **mutate()**: Varolan değişkenlerin fonksiyonları olan yeni değişkenler ekler
 - **select()**: Değişkenleri adlarına göre seçer.
 - **filter()**: Değişkenlerin koşullara göre sorgulanmasını sağlar.
 - **summarise()**: Birden çok değeri tek bir özete indirger.
 - **arrange()**: Satır sıralamasını değiştirir.

dplyr'a Giriş

dplyr Function	Description	Equivalent SQL
select()	Selecting columns (variables)	SELECT
filter()	Filter (subset) rows.	WHERE
group_by()	Group the data	GROUP BY
summarise()	Summarise (or aggregate) data	-
arrange()	Sort the data	ORDER BY
join()	Joining data frames (tables)	JOIN
mutate()	Creating New Variables	COLUMN ALIAS

data frame'den dplyr'a Geçiş

```
library(dplyr)
```

```
x <- as_tibble(iris)
```

```
## # A tibble: 3 x 5
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

```
##           <dbl>         <dbl>         <dbl>         <dbl> <fct>
```

```
## 1           5.1           3.5           1.4           0.2 seto
```

```
## 2           4.9           3           1.4           0.2 seto
```

```
## 3           4.7           3.2           1.3           0.2 seto
```

- **filter()** fonksiyonu, bir veri setindeki belirtilen koşulu veya koşulları sağlayan satırları filtreleyerek veri setini alt bir veri setine indirger.

```
Fonksiyon Yapısı : filter(.data, ...)  
                  data %>% filter(...)
```

```
Argümanlar      : .data : veri setinin adı  
                  ...   : filtreleme koşulu
```

filter

```
starwars %>% filter(species == "Droid")
```

```
## # A tibble: 5 x 13
```

```
##   name    height    mass hair_color skin_color eye_color birth_year
```

```
##   <chr>    <int> <dbl> <chr>      <chr>      <chr>
```

```
## 1 C-3PO      167     75 <NA>      gold        yellow
```

```
## 2 R2-D2       96     32 <NA>      white, bl~ red
```

```
## 3 R5-D4       97     32 <NA>      white, red red
```

```
## 4 IG-88      200    140 none      metal        red
```

```
## 5 BB8        NA      NA none      none         black
```

```
## # ... with 5 more variables: homeworld <chr>, species <chr>
```

```
## #   vehicles <list>, starships <list>
```

select

- **select()** fonksiyonu, veri setini belirli sütunlarda incelemek amacıyla kullanılmaktadır. Bazı özel fonksiyonlar yardımıyla seçme işlemi özelleştirilebilir.

```
Fonksiyon Yapısı : select(.data, ...)
                  data %>% select(...)
```

```
Argümanlar      : .data : veri setinin adı
                  ...    : sütun adı ya da
                  özel fonksiyonlar
```

select

```
starts_with() : ... ile başlayan sütunlar
ends_with()   : ... ile biten sütunlar
contains()    : ... içeren sütunlar
matches()     : belirtilen ifadelerle eşleşen sütunlar
num_range()   : x01,x02, x03 gibi sıradaki sütunlar
one_of()      : sütunlardan oluşan vektördeki sütunlar
everything()  : tüm sütunlar
```


select

```
starwars %>%  
  select(name, ends_with("color")) %>%  
  head(.,4)
```

```
## # A tibble: 4 x 4  
##   name          hair_color skin_color eye_color  
##   <chr>         <chr>         <chr>    <chr>  
## 1 Luke Skywalker blond         fair      blue  
## 2 C-3PO         <NA>          gold      yellow  
## 3 R2-D2         <NA>          white, blue red  
## 4 Darth Vader   none          white     yellow
```

- **mutate()** fonksiyonu, yeni bir sütun eklemek ya da var olan sütunu değiştirmek amacıyla kullanılmaktadır.

```
Fonksiyon Yapısı : mutate(.data, ...)  
                  data %>% mutate(...)
```

```
Argümanlar      : .data : veri setinin adı  
                  ...    : oluşacak sütun için ifadeler
```

mutate

```
starwars %>%  
  mutate(bmi = mass/((height/100)^2)) %>%  
  select(name:mass, bmi) %>%  
  head(.,4)
```

```
## # A tibble: 4 x 4  
##   name          height  mass   bmi  
##   <chr>         <int> <dbl> <dbl>  
## 1 Luke Skywalker    172    77  26.0  
## 2 C-3P0             167    75  26.9  
## 3 R2-D2              96    32  34.7  
## 4 Darth Vader       202   136  33.3
```

mutate & if else

- Bir önceki adımda vücut kütle endeksi hesaplamış ve **bmi** olarak yeni kolon yaratmıştık.
- **bmi** değerinin 30'dan büyük olması durumunda **TRUE**, aksi durumlarda **FALSE** değerini alan yeni bir kolon ekleyelim.

```
starwars %>%  
  mutate(bmi = mass/((height/100)^2),  
         bmi_control = if_else(bmi>=30, TRUE, FALSE)) %>%  
  select(name:mass, contains("bmi")) %>%  
  head
```

mutate & if else

```
## # A tibble: 6 x 5
##   name          height  mass   bmi bmi_control
##   <chr>         <int> <dbl> <dbl> <lgl>
## 1 Luke Skywalker   172    77  26.0 FALSE
## 2 C-3P0            167    75  26.9 FALSE
## 3 R2-D2            96     32  34.7 TRUE
## 4 Darth Vader      202   136  33.3 TRUE
## 5 Leia Organa      150    49  21.8 FALSE
## 6 Owen Lars        178   120  37.9 TRUE
```

mutate & case when

```
starwars %>%  
  select(name:mass, gender, species) %>%  
  mutate(  
    type = case_when(  
      height > 200 | mass > 200 ~ "large",  
      species == "Droid" ~ "robot",  
      TRUE ~ "other"  
    )  
  ) %>%  
  head
```

mutate & case when

```
## # A tibble: 6 x 6
##   name          height  mass gender species type
##   <chr>         <int> <dbl> <chr>  <chr>  <chr>
## 1 Luke Skywalker   172    77 male   Human  other
## 2 C-3P0            167    75 <NA>   Droid  robot
## 3 R2-D2            96     32 <NA>   Droid  robot
## 4 Darth Vader      202   136 male   Human  large
## 5 Leia Organa      150    49 female Human  other
## 6 Owen Lars        178   120 male   Human  other
```

mutate & case when - Örnek

Soru:

Uluslararası standartlarda bmi kategorileri şu şekildedir: Bmi değeri 18.5'dan küçükse Underweight, 18.5 - 24.9 arasındaysa Normal, 25 - 29.9 arasındaysa Overweight, 30 ve üzerindeyse Obesity.

Bu veriler ışığında daha önce hesaplanan bmi değeri için mutate ve case when bloğu kullanarak bmi kategorilerini belirleyiniz.

mutate & case when - Örnek

```
starwars %>%  
  mutate(bmi = mass/((height/100)^2),  
         bmi_category = case_when(  
           bmi < 18.5 ~ "Underweight",  
           bmi >= 18.5 & bmi < 25 ~ "Normal",  
           bmi >= 25 & bmi < 30 ~ "Overweight",  
           TRUE ~ "Obesity"  
         )) %>%  
  select(name:mass, contains("bmi")) %>%  
  head
```

mutate & case when - Örnek

```
## # A tibble: 6 x 5
##   name          height  mass   bmi bmi_category
##   <chr>         <int> <dbl> <dbl> <chr>
## 1 Luke Skywalker   172    77  26.0 Overweight
## 2 C-3P0            167    75  26.9 Overweight
## 3 R2-D2             96    32  34.7 Obesity
## 4 Darth Vader      202   136  33.3 Obesity
## 5 Leia Organa      150    49  21.8 Normal
## 6 Owen Lars        178   120  37.9 Obesity
```

arrange

- **arrange()** fonksiyonu, veriyi belirli sütunlara göre sıralı düzende görmek için kullanılır. Sütunlar büyükten küçüğe ya da küçükten büyüğe sıralanır.

```
Fonksiyon Yapısı : arrange(.data, ...)  
                  data %>% arrange(...)
```

```
Argümanlar      : .data : veri setinin adı  
                  ...    : sırala yapılacak sütunlar  
  
                  **desc()** argümanı kullanılarak  
                  büyükten küçüğe sıralama yapılabilir.
```

arrange

```
starwars %>% arrange(desc(mass)) %>% head(.,7)
```

```
## # A tibble: 7 x 13
##   name    height    mass hair_color skin_color eye_color birth_year
##   <chr>    <int> <dbl> <chr>         <chr>         <chr>
## 1 Jabb~      175   1358 <NA>         green-tan~ orange
## 2 Grie~      216   159 none         brown, wh~ green, y~
## 3 IG-88      200   140 none         metal         red
## 4 Dart~      202   136 none         white         yellow
## 5 Tarf~      234   136 brown        brown         blue
## 6 Owen~      178   120 brown, gr~ light         blue
## 7 Bossk      190   113 none         green         red
## # ... with 5 more variables: homeworld <chr>, species <chr>,
## #   vehicles <list>, starships <list>
```

group by

- Bir çok veri işlemi grup bazında yapılmaktadır.
- **group_by()** fonksiyonu, veri seti sütunlarını gruplamak için kullanılır.
- Grup bazında veri analizleri gerçekleştirdikten sonra grubu dağıtmak için **ungroup()** fonksiyonuna ihtiyaç duyulur.

```
Fonksiyon Yapısı : group_by(.data, ...)  
                  data %>% group_by(...)
```

```
Argümanlar      : .data : veri setinin adı  
                  ...   : gruptama yapılacak sütunlar
```

summarise

- **summarise()** fonksiyonu, özet istatistikleri hesaplamak için kullanılır ve genellikle gruplanmış veri üzerine uygulanır.
- Çıktıda her grup için özet satır görülür.

```
Fonksiyon Yapısı : summarise(.data, ...)  
                  data %>% summarise(...)
```

```
Argümanlar       : .data : veri setinin adı  
                  ...    : min(), mean(), max()
```

group by & summarise

```
starwars %>%  
  group_by(species) %>%  
  summarise(  
    n = n(),  
    mass = mean(mass, na.rm = TRUE)  
  ) %>%  
  filter(n > 1) %>%  
  head(.,5)
```

```
## # A tibble: 5 x 3  
##   species      n  mass  
##   <chr>    <int> <dbl>  
## 1 Droid         5  69.8  
## 2 Gungan        3  74  
## 3 Human       35  82.8  
## 4 Kaminoan     2  88  
## 5 Wookiee       6  90
```

- `dplyr` paketi içerisinde 4 farklı join fonksiyonu bulunmaktadır.

a		b	
x1	x2	x1	x3
A	1	A	T
B	2	B	F
C	3	D	T

+ =

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

`dplyr::left_join(a, b, by = "x1")`

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

`dplyr::right_join(a, b, by = "x1")`

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

`dplyr::inner_join(a, b, by = "x1")`

Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

`dplyr::full_join(a, b, by = "x1")`

Join data. Retain all values, all rows.

- `dplyr` paketi içerisindeki **band_members** ve **band_instruments** datalarını göz önüne alalım.

```
band_members
```

```
## # A tibble: 3 x 2
##   name band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```

```
band_instruments
```

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

join - left_join()

```
band_members %>% left_join(band_instruments, by="name")
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
```

join - right_join()

```
band_members %>% right_join(band_instruments, by="name")
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
## 3 Keith <NA>    guitar
```

join - inner_join()

```
band_members %>% inner_join(band_instruments, by="name")
```

```
## # A tibble: 2 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles  bass
```

join - full_join()

```
band_members %>% full_join(band_instruments, by="name")
```

```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

- **tidyr** paketi içerisindeki **gather()** fonksiyonu ile sütunlara dağılmış bir veri seti anahtar bir sütun ile toplanabilir.

Data Düzenleme - gather()

```
Fonksiyon Yapısı : gather(data, key, value, ...,  
                           na.rm = FALSE)  
data %>% gather(key, value, ...,  
                na.rm = FALSE)
```

```
Argümanlar : .data      : veri setinin adı  
             key        : sütunların satır değişkeni olarak  
                        toplanacağı yeni sütunun adı  
             value      : sütun değerlerinin toplanacağı  
                        yeni sütunun adı  
             ...        : birleştirilecek sütunların adları  
             na.rm      : NA değerlerini kaldırır.
```


Data Düzenleme - gather()

```
tidyr::smiths
```

```
## # A tibble: 2 x 5
##   subject      time    age weight height
##   <chr>      <dbl> <dbl>  <dbl>  <dbl>
## 1 John Smith      1     33     90    1.87
## 2 Mary Smith      1    NA     NA    1.54
```

smiths veri setinde subject sütunu dışındaki sütunları Key adında tek bir sütuna toplayalım.

```
smiths %>%
  gather(key = Key, value = Values, -subject ,
         na.rm = FALSE, convert = TRUE)
```

Data Düzenleme - gather()

```
## # A tibble: 8 x 3
##   subject      Key      Values
##   <chr>        <chr>    <dbl>
## 1 John Smith time      1
## 2 Mary Smith time      1
## 3 John Smith age      33
## 4 Mary Smith age      NA
## 5 John Smith weight    90
## 6 Mary Smith weight    NA
## 7 John Smith height    1.87
## 8 Mary Smith height    1.54
```

Data Düzenleme - gather()

```
library(readr)
explanatory <- read_csv("data/explanatory.csv",
                        col_types = cols(
                          Date = col_date(
                            format = "%Y-%m-%d")
                        )
                      )

head(explanatory, 5)
```

Data Düzenleme - gather()

```
## # A tibble: 5 x 14
##   Date          EURUSD USDJPY SP500 FTSE100    DAX    MCX    S
##   <date>        <dbl>  <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 2015-08-03    1.10   124. 2098.  6689. 11444. 1664. 36
## 2 2015-08-04    1.09   124. 2093.  6687. 11456. 1674. 37
## 3 2015-08-05    1.09   125. 2100.  6752. 11636. 1693. 36
## 4 2015-08-06    1.09   125. 2084.  6747. 11585. 1676. 36
## 5 2015-08-07    1.10   124. 2078.  6718. 11491. 1690. 37
## # ... with 4 more variables: WTI <dbl>, AAPL <dbl>, FB <dbl>
```

Data Düzenleme - gather()

```
library(tidyr)
explanatory %>% gather("asset", "price", -Date)
```

```
## # A tibble: 9,061 x 3
##   Date      asset price
##   <date>    <chr> <dbl>
## 1 2015-08-03 EURUSD  1.10
## 2 2015-08-04 EURUSD  1.09
## 3 2015-08-05 EURUSD  1.09
## 4 2015-08-06 EURUSD  1.09
## 5 2015-08-07 EURUSD  1.10
## 6 2015-08-10 EURUSD  1.10
## 7 2015-08-11 EURUSD  1.10
## 8 2015-08-12 EURUSD  1.12
## 9 2015-08-13 EURUSD  1.11
```

Data Düzenleme - spread()

- **tidyr** paketi içerisindeki **spread()** fonksiyonu ile anahtar bir sütunda toplanmış olan datalar sütunlara dağıtılabilmektedir.

Fonksiyon Yapısı : `spread(data, key, value, fill = NA)`
`data %>% spread(key, value, fill = NA)`

Argümanlar :

<code>.data</code>	:	veri setinin adı
<code>key</code>	:	birden çok sütuna dağıtılacak değerlerin bulunduğu sütunun adı
<code>value</code>	:	dağıtılacak değerlerin bulunduğu sütunun adı
<code>fill</code>	:	boş gelen satırların doldurulacağı değer

Data Düzenleme - spread()

```
df_res <- dbSendQuery(conn,
  paste0("SELECT data_date,asset_code, close
        FROM parity WHERE data_date between
        '2015-02-01' and '2015-03-01';"))

df <- dbFetch(df_res)
df <- as_tibble(df)
df %>% spread(key=asset_code, value=close, fill=NA)
```

A tibble: 20 x 4

##	data_date	`EUR/TRY`	`EUR/USD`	`USD/TRY`
##	<dtm>	<dbl>	<dbl>	<dbl>
##	1 2015-02-02 00:00:00	2.76	1.14	2.43
##	2 2015-02-03 00:00:00	2.75	1.14	2.40

Data Düzenleme - spread()

##	3	2015-02-04	00:00:00	2.79	1.14	2.44
##	4	2015-02-05	00:00:00	2.79	1.14	2.44
##	5	2015-02-06	00:00:00	2.80	1.13	2.47
##	6	2015-02-09	00:00:00	2.81	1.13	2.48
##	7	2015-02-10	00:00:00	2.83	1.13	2.50
##	8	2015-02-11	00:00:00	2.83	1.13	2.50
##	9	2015-02-12	00:00:00	2.82	1.14	2.47
##	10	2015-02-13	00:00:00	2.81	1.14	2.46
##	11	2015-02-16	00:00:00	2.80	1.14	2.45
##	12	2015-02-17	00:00:00	2.80	1.14	2.45
##	13	2015-02-18	00:00:00	2.79	1.14	2.45
##	14	2015-02-19	00:00:00	2.79	1.14	2.45
##	15	2015-02-20	00:00:00	2.79	1.13	2.46

Data Düzenleme - spread()

##	16	2015-02-23	00:00:00	2.81	1.13	2.47
##	17	2015-02-24	00:00:00	2.80	1.13	2.47
##	18	2015-02-25	00:00:00	2.82	1.14	2.48
##	19	2015-02-26	00:00:00	2.80	1.12	2.49
##	20	2015-02-27	00:00:00	2.82	1.12	2.52

Data Düzenleme - unite

- `tidyr` paketindeki `unite()` fonksiyonu, istenen sütunları birleştirerek tek bir sütunda toplar.

```
Fonksiyon Yapısı : unite(data, col, ..., sep = "",  
                          remove = TRUE)  
data %>% unite(col, ..., sep = "",  
               remove = TRUE)
```

```
Argümanlar : .data      : veri setinin adı  
             col        : yeni sütun adı  
             ...        : birleştirilecek sütunlar  
             sep        : ayırıcı  
             remove     : TRUE birleştirilen sütunlar  
                           kaldırılır
```

Data Düzenleme - unite

```
starwars %>%  
  unite(New, c("species", "gender"), sep="_") %>%  
  select(name, New)
```

```
## # A tibble: 87 x 2
```

```
##   name                New  
##   <chr>              <chr>  
## 1 Luke Skywalker    Human_male  
## 2 C-3P0              Droid_NA  
## 3 R2-D2              Droid_NA  
## 4 Darth Vader        Human_male  
## 5 Leia Organa        Human_female  
## 6 Owen Lars           Human_male  
## 7 Beru Whitesun lars Human_female  
## 8 R5-D4               Droid_NA
```

dplyr & odbc ile Veritabanı Bağlantısı

- Daha önceki bölümlerde **odbc** kütüphanesi ile veritabanı bağlantılarımızı kurmuş ve sorgularımızı çalıştırmıştık.
- Bu bölümde **odbc** ve **pool** kütüphaneleri ile veritabanı bağlantısı kurup, sorgularımızı **dplyr** fiilleri ile yapacağız.

```
install.packages(c("pool", "odbc"))
```

```
library(odbc)
```

```
con <- dbConnect(odbc::odbc(), "r2active")
```

- Daha önce olduğu gibi parite tablosundan **asset_code** değeri USD/TRY olan fiyatları alalım.

dplyr & odbc ile Veritabanı Bağlantısı

- Bunun için bağlantı kurduğumuz **con** değişkeni ve **tbl** fonksiyonunu kullanarak tabloya erişiyoruz.

```
tbl(con, "parity") %>% filter(asset_code == "USD/TRY")
```

```
## # Source:   lazy query [?? x 7]
## # Database: MySQL 5.7.21-0ubuntu0.16.04.1[user@r2active
## #   TCP/IP/racademy]
##   id                data_date                asset_code  open
##   <S3: integer64> <dtm>                <chr>       <dbl>
## 1 " 1"              2015-12-31 00:00:00 USD/TRY      2.92
## 2 " 6"              2015-12-30 00:00:00 USD/TRY      2.91
## 3 " 9"              2015-12-29 00:00:00 USD/TRY      2.91
## 4 10                2015-12-28 00:00:00 USD/TRY      2.92
```

dplyr & odbc ile Veritabanı Bağlantısı

```
## 5 13          2015-12-25 00:00:00 USD/TRY      2.92
## 6 17          2015-12-24 00:00:00 USD/TRY      2.92
## 7 21          2015-12-23 00:00:00 USD/TRY      2.93
## 8 22          2015-12-22 00:00:00 USD/TRY      2.92
## 9 25          2015-12-21 00:00:00 USD/TRY      2.91
## 10 28         2015-12-18 00:00:00 USD/TRY      2.93
## # ... with more rows
```

- Sonuç incelendiğinde tablonun tamamına değil ilk 10 satıra erişebildiğimize dikkat edin.
- Tablonun tamamının R ortamına alınması için `collect()` fonksiyonu kullanılacaktır.

dplyr & odbc ile Veritabanı Bağlantısı

```
tbl(con, "parity") %>%  
  filter(asset_code == "USD/TRY") %>%  
  collect()
```

```
## # A tibble: 252 x 7
```

##	id	data_date	asset_code	open
##	<S3: integer64>	<dtm>	<chr>	<dbl>
##	1 " 1"	2015-12-31 00:00:00	USD/TRY	2.92
##	2 " 6"	2015-12-30 00:00:00	USD/TRY	2.91
##	3 " 9"	2015-12-29 00:00:00	USD/TRY	2.91
##	4 10	2015-12-28 00:00:00	USD/TRY	2.92
##	5 13	2015-12-25 00:00:00	USD/TRY	2.92
##	6 17	2015-12-24 00:00:00	USD/TRY	2.92
##	7 21	2015-12-23 00:00:00	USD/TRY	2.93

dplyr & odbc ile Veritabanı Bağlantısı

```
##      8 22      2015-12-22 00:00:00 USD/TRY      2.92
##      9 25      2015-12-21 00:00:00 USD/TRY      2.91
##     10 28      2015-12-18 00:00:00 USD/TRY      2.93
## # ... with 242 more rows
```


- Daha önce yaptığımız örneği, **dplyr** bakış açısı ile tekrar yapalım.

Soru:

Veritabanından 2015-02-01 ile 2015-03-01 tarihleri arasında EUR/USD paritesinin kapanış fiyatını çekelim ve paritenin currency kodunu parite datasına ekleyelim.

Örnek

```
parity <- tbl(con, "parity") %>%  
  filter(asset_code == "EUR/USD",  
         data_date >= "2015-02-01",  
         data_date <= "2015-03-01") %>%  
  select(data_date, asset_code, close) %>%  
  collect()
```

```
currency <- tbl(con, "currency") %>%  
  filter(asset_code == "EUR/USD") %>%  
  select(asset_code, currency_code) %>%  
  collect()
```

```
df <- left_join(parity, currency, by="asset_code")
```

```
head(df)
```

```
## # A tibble: 6 x 4
```

##	data_date	asset_code	close	currency_code
##	<dtm>	<chr>	<dbl>	<chr>
## 1	2015-02-27 00:00:00	EUR/USD	1.12	USD
## 2	2015-02-26 00:00:00	EUR/USD	1.12	USD
## 3	2015-02-25 00:00:00	EUR/USD	1.14	USD
## 4	2015-02-24 00:00:00	EUR/USD	1.13	USD
## 5	2015-02-23 00:00:00	EUR/USD	1.13	USD
## 6	2015-02-20 00:00:00	EUR/USD	1.13	USD

tidyverse: Seçilmiş Örnekler

Örnek 1: rename

- **rename()** fonksiyonu, sütun değişkenlerini yeniden adlandırmak için kullanılır.

```
starwars %>% select(name,height) %>% rename(boy = height)
```

```
## # A tibble: 87 x 2
```

```
##   name                boy
```

```
##   <chr>              <int>
```

```
## 1 Luke Skywalker    172
```

```
## 2 C-3P0              167
```

```
## 3 R2-D2              96
```

```
## 4 Darth Vader       202
```

```
## 5 Leia Organa       150
```

```
## 6 Owen Lars         178
```

Örnek 1: rename

```
## 7 Beru Whitesun lars 165
## 8 R5-D4 97
## 9 Biggs Darklighter 183
## 10 Obi-Wan Kenobi 182
## # ... with 77 more rows
```

Örnek 2: pull

- **pull()** fonksiyonu, veri setinin sütun değerlerini vektör elemanına çevirir.

```
starwars %>% head(.,10) %>% pull(name)
```

```
## [1] "Luke Skywalker"      "C-3P0"                "R2-D2"  
## [4] "Darth Vader"         "Leia Organa"          "Owen Lars"  
## [7] "Beru Whitesun lars"  "R5-D4"                "Biggs Darklighter"  
## [10] "Obi-Wan Kenobi"
```

Örnek 3: distinct

- **distinct()** fonksiyonu, belirli sütunlara göre tekrarlı satırları ortadan kaldırır.

```
starwars %>% distinct(homeworld,species)
```

```
## # A tibble: 58 x 2
##   homeworld species
##   <chr>      <chr>
## 1 Tatooine   Human
## 2 Tatooine   Droid
## 3 Naboo      Droid
## 4 Alderaan   Human
## 5 Stewjon    Human
## 6 Eriadu     Human
```


Örnek 3: distinct

```
## 7 Kashyyyk Wookiee
## 8 Corellia Human
## 9 Rodia Rodian
## 10 Nal Hutta Hutt
## # ... with 48 more rows
```

Örnek 4: filter, %in%

```
starwars %>% filter(species %in% c("Human", "Droid"))
```

```
## # A tibble: 40 x 13
```

```
##   name      height  mass hair_color skin_color eye_color blood_color
```

```
##   <chr>    <int> <dbl> <chr>      <chr>      <chr>      <chr>
```

```
## 1 Luke~    172    77 blond      fair        blue
```

```
## 2 C-3P0    167    75 <NA>      gold        yellow
```

```
## 3 R2-D2     96    32 <NA>      white, bl~ red
```

```
## 4 Dart~    202   136 none      white       yellow
```

```
## 5 Leia~    150    49 brown     light       brown
```

```
## 6 Owen~    178   120 brown, gr~ light       blue
```

```
## 7 Beru~    165    75 brown     light       blue
```

```
## 8 R5-D4     97    32 <NA>      white, red red
```

Örnek 4: filter, %in%

```
## 9 Bigg~      183      84 black      light      brown
## 10 Obi-~     182      77 auburn, w~ fair      blue-gray
## # ... with 30 more rows, and 5 more variables: homeworld
## #   species <chr>, films <list>, vehicles <list>, starsh
```

Örnek 5: filter, grepl

```
starwars %>% filter(grepl("H", name))
```

```
## # A tibble: 2 x 13
##   name height mass hair_color skin_color eye_color birth_year
##   <chr>   <int> <dbl> <chr>         <chr>         <chr>
## 1 Han ~    180    80 brown         fair          brown
## 2 San ~    191   NA none          grey          gold
## # ... with 5 more variables: homeworld <chr>, species <chr>,
## #   vehicles <list>, starships <list>
```

Örnek 6: add_row

- **add_row** fonksiyonu, veri setine yeni bir satır eklemek için kullanılır.

```
tibble(cocuk = c("Ali", "Ayşe"),
       yas = c(5, 7),
       kilo = c(18, 23)) %>%
  add_row(cocuk = "Fatih", yas = 6, kilo = 20)
```

```
## # A tibble: 3 x 3
##   cocuk   yas kilo
##   <chr> <dbl> <dbl>
## 1 Ali      5    18
## 2 Ayse     7    23
## 3 Fatih    6    20
```

Örnek 7: lag/lead

- **lag()** fonksiyonu satır elemanlarını gecikmeli olarak kullanır.
- **lead()** fonksiyonu ise kaydırma işlemi, bir önceki satıra doğrudur.

```
starwars %>% head(.,10) %>%  
  select(name) %>%  
  mutate(nameLag = lag(name),  
         nameLead = lead(name))
```

```
## # A tibble: 10 x 3
```

##	name	nameLag	nameLead
##	<chr>	<chr>	<chr>
##	1 Luke Skywalker	<NA>	C-3P0
##	2 C-3P0	Luke Skywalker	R2-D2

Örnek 7: lag/lead

##	3	R2-D2	C-3P0	Darth Vader
##	4	Darth Vader	R2-D2	Leia Organa
##	5	Leia Organa	Darth Vader	Owen Lars
##	6	Owen Lars	Leia Organa	Beru Whitesun lars
##	7	Beru Whitesun lars	Owen Lars	R5-D4
##	8	R5-D4	Beru Whitesun lars	Biggs Darklighter
##	9	Biggs Darklighter	R5-D4	Obi-Wan Kenobi
##	10	Obi-Wan Kenobi	Biggs Darklighter	<NA>

Örnek 8: do

do fonksiyonu, yapılması istenen işlemi grup bazında uygular.

```
starwars %>%  
  select(homeworld,name,height) %>%  
  group_by(homeworld) %>%  
  arrange(desc(height)) %>%  
  do(head(., 1))
```

```
## # A tibble: 49 x 3  
## # Groups:   homeworld [49]  
##   homeworld      name      height  
##   <chr>         <chr>    <int>  
## 1 Alderaan     Bail Prestor Organa    191  
## 2 Aleen Minor  Ratts Tyerell         79
```


Örnek 8: do

##	3	Bespin	Lobot	175
##	4	Bestine IV	Jek Tono Porkins	180
##	5	Cato Neimoidia	Nute Gunray	191
##	6	Cerea	Ki-Adi-Mundi	198
##	7	Champala	Mas Amedda	196
##	8	Chandrila	Mon Mothma	150
##	9	Concord Dawn	Jango Fett	183
##	10	Corellia	Han Solo	180
##	#	... with 39 more rows		

Örnek 9: intersect

- **intersect()** fonksiyonu, iki datasette ortak olan unique (eşsiz) satırları belirler.

```
df_1 <- mtcars[1:7,] %>% as_tibble
df_2 <- mtcars[5:10,] %>% as_tibble

intersect(df_1, df_2)
```

```
## # A tibble: 3 x 11
##   mpg    cyl  disp    hp  drat    wt   qsec    vs    am
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  18.7     8   360   175  3.15  3.44  17.0     0     0
## 2  18.1     6   225   105  2.76  3.46  20.2     1     0
## 3  14.3     8   360   245  3.21  3.57  15.8     0     0
```

Örnek 10: union

- **union()** fonksiyonu, her iki veri setindeki tüm satırları tekrarlı olanları kaldırarak verileri dikey olarak bir araya getirir.

```
union(df_1, df_2)
```

```
## # A tibble: 10 x 11
```

```
##      mpg    cyl  disp    hp  drat    wt    qsec    vs    am  
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1  21      6   160    110  3.9    2.62  16.5     0     1  
## 2  18.1     6   225    105  2.76   3.46  20.2     1     0  
## 3  19.2     6   168.    123  3.92   3.44  18.3     1     0  
## 4  14.3     8   360    245  3.21   3.57  15.8     0     0  
## 5  18.7     8   360    175  3.15   3.44  17.0     0     0  
## 6  22.8     4   141.     95  3.92   3.15  22.9     1     0
```

Örnek 10: union

##	7	24.4	4	147.	62	3.69	3.19	20	1	0
##	8	21	6	160	110	3.9	2.88	17.0	0	1
##	9	22.8	4	108	93	3.85	2.32	18.6	1	1
##	10	21.4	6	258	110	3.08	3.22	19.4	1	0

Örnek 11: setdiff

- **setdiff()** fonksiyonu, ilk veri setinde olup ikincisinde olmayan satırları listeler.

```
setdiff(df_1, df_2)
```

```
## # A tibble: 4 x 11
```

##	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
##	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	21	6	160	110	3.9	2.62	16.5	0	1
## 2	21	6	160	110	3.9	2.88	17.0	0	1
## 3	22.8	4	108	93	3.85	2.32	18.6	1	1
## 4	21.4	6	258	110	3.08	3.22	19.4	1	0

Data İşleme

Örnek:

explanatory.csv dosyasındaki datalar için öncelikle hafta sonlarını temizleyip, ardından eksik dataları bulup bir önceki gün datasını kopyalayalım.

Missing Data

```
library(readr)
explanatory <- read_csv("data/explanatory.csv",
                        col_types = cols(
                          Date = col_date(
                            format = "%Y-%m-%d")
                        )
                      )

head(explanatory, 5)
```


Missing Data

```
## # A tibble: 5 x 14
##   Date          EURUSD USDJPY SP500 FTSE100    DAX    MCX    S
##   <date>        <dbl>  <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 2015-08-03    1.10   124. 2098.  6689. 11444. 1664. 36
## 2 2015-08-04    1.09   124. 2093.  6687. 11456. 1674. 37
## 3 2015-08-05    1.09   125. 2100.  6752. 11636. 1693. 36
## 4 2015-08-06    1.09   125. 2084.  6747. 11585. 1676. 36
## 5 2015-08-07    1.10   124. 2078.  6718. 11491. 1690. 37
## # ... with 4 more variables: WTI <dbl>, AAPL <dbl>, FB <dbl>
```

- EURUSD sütununda herhangi bir eksik data olup olmadığını kontrol edelim.

```
any(is.na(explanatory[, "EURUSD"]))
```

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

Missing Data

- Tüm sütunlarda eksik data olup olmadığını kontrol edelim.

```
apply(explanatory[,-1], 2, function(x) any(is.na(x)) )
```

##	EURUSD	USDJPY	SP500	FTSE100	DAX	MCX	SSEC
##	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
##	WTI	AAPL	FB	GOOGL			
##	TRUE	TRUE	TRUE	TRUE			

Missing Data

- Tüm sütunlarda eksik olan dataları belirleyelim.

```
explanatory[!complete.cases(explanatory),]
```

```
## # A tibble: 295 x 14
```

##	Date	EURUSD	USDJPY	SP500	FTSE100	DAX	MCX
##	<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
##	1 2015-08-15	NA	NA	NA	NA	NA	NA
##	2 2015-08-16	NA	NA	NA	NA	NA	NA
##	3 2015-08-22	NA	NA	NA	NA	NA	NA
##	4 2015-08-23	NA	NA	NA	NA	NA	NA
##	5 2015-08-29	NA	NA	NA	NA	NA	NA
##	6 2015-08-30	NA	NA	NA	NA	NA	NA
##	7 2015-08-31	1.12	121.	1972.	NA	10259.	1733.
##	8 2015-09-03	1.11	120.	1951.	6194.	10318.	1711.
##	9 2015-09-04	1.11	119	1921.	6043.	10038.	1698.

Missing Data - Haftasonlarını temizleme

- Datadan hafta sonlarını temizleyelim.

```
calcSeq <- seq(explanatory$Date[1],  
              explanatory$Date[nrow(explanatory)],  
              by = 1)  
  
calcSeq <- as.Date(setdiff(calcSeq,  
                          calcSeq[which(  
                            chron::is.weekend(calcSeq)==TRUE)]  
                          ),origin = "1970-01-01")  
  
explanatory <- explanatory[explanatory$Date  
                          %in% calcSeq,]
```

Missing Data

- Hafta sonlarını temizledikten sonra kalan data sayısını kontrol edelim.

```
## [1] 501
```

- Hafta sonlarını temizlediğimiz data setinde eksik datalar için bir önceki günün kayıtlarını alalım.

```
explanatory <- cbind(explanatory[, "Date"],  
                      zoo::na.locf(explanatory[, -1],  
                                   fromLast = TRUE)  
                      )
```

- Son değişimler sonrası sütunlarda eksik data olup olmadığını kontrol edelim.

Missing Data

```
apply(explanatory[, -1], 2, function(x) any(is.na(x)) )
```

##	EURUSD	USDJPY	SP500	FTSE100	DAX	MCX	SSEC
##	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
##	WTI	AAPL	FB	GOOGL			
##	FALSE	FALSE	FALSE	FALSE			

Missing Data - dplyr Uygulaması

- Bir önceki bölümde bir dataset içerisinden hafta sonlarını temizlemiştik. Aynı uygulamayı **dplyr** kütüphanesi ile yapalım.
- Öncelikle bir önceki uygulamada temizlediğimiz datayı `explanatory` ismi ile kayıt ettiğimiz için, datayı tekrar yüklememiz gereklidir.

```
explanatory <- read_csv("data/explanatory.csv",  
                        col_types = cols(  
                          Date = col_date(  
                            format = "%Y-%m-%d")  
                          )  
                        )
```


Missing Data - dplyr Uygulaması

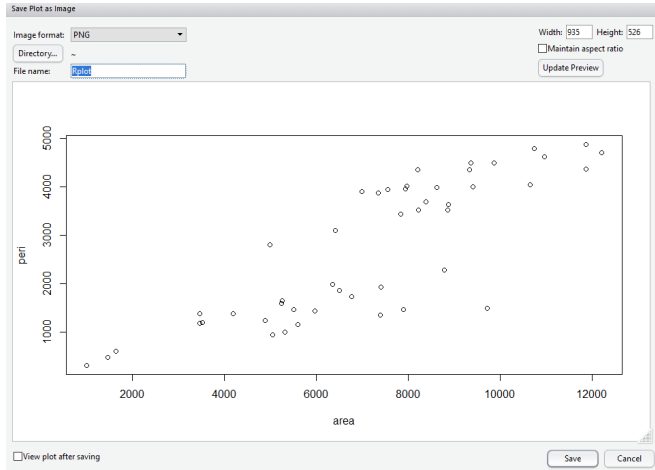
```
explanatory <- explanatory %>%  
  mutate(flag=chron::is.weekend(Date)) %>%  
  filter(flag == FALSE) %>% select(-flag) %>%  
  tidyr::fill(everything(), .direction="down")  
  
apply(explanatory[,-1], 2, function(x) any(is.na(x)))
```

##	EURUSD	USDJPY	SP500	FTSE100	DAX	MCX	SSEC
##	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
##	WTI	AAPL	FB	GOOGL			
##	FALSE	FALSE	FALSE	FALSE			

“R Base” Türü Grafik Çizimi

- `base`, `lattice` ve `ggplot2` en çok kullanılan çizim paketleridir.
- `plotly` library desteği ile `ggplot2` en gelişmiş çizim sistemidir.
- Pratikte en fazla kullanılan sistemler `base` ve `ggplot2` sistemleridir.

Çizim Gereçleri



Nokta (Scatter) Grafik

- **mtcars**, 1974 Motor Trend US dergisinde 73-74 model 32 aracın 10 farklı açıdan karşılaştırılmalarının yapıldığı bir veri setidir.
 - mpg, miles/gallon
 - cyl, silindir
 - disp, silindir hacmi
 - hp, toplam beygir gücü
 - drat, arka aks oranı
 - qsec, 1/4 mil süresi
 - greb, vites sayısı
 - wt, ağırlık (1000 lbs)
 - am, vites, (0 = otomatik, 1 = düz)
 - carb, karbüratör sayısı

Nokta (Scatter) Grafik

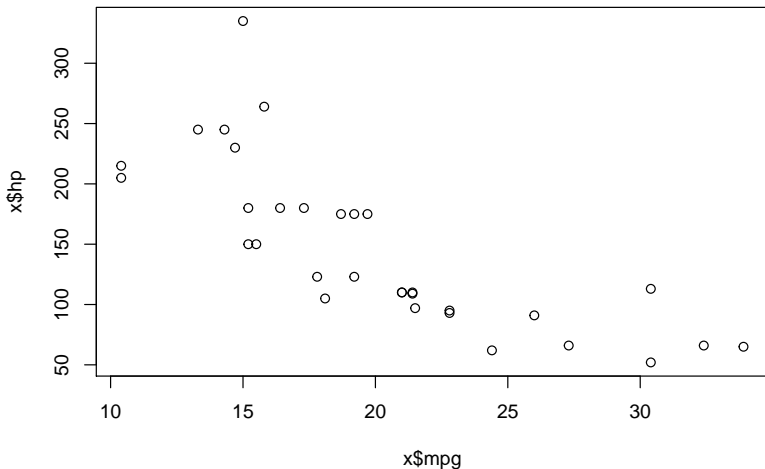
```
x <- datasets::mtcars  
head(x)
```

##	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	g
## Mazda RX4	21	6	160	110	3.9	2.6	16	0	1	
## Mazda RX4 Wag	21	6	160	110	3.9	2.9	17	0	1	
## Datsun 710	23	4	108	93	3.8	2.3	19	1	1	
## Hornet 4 Drive	21	6	258	110	3.1	3.2	19	1	0	
## Hornet Sportabout	19	8	360	175	3.1	3.4	17	0	0	
## Valiant	18	6	225	105	2.8	3.5	20	1	0	

Nokta (Scatter) Grafik

```
plot(x$mpg, x$hp)
```

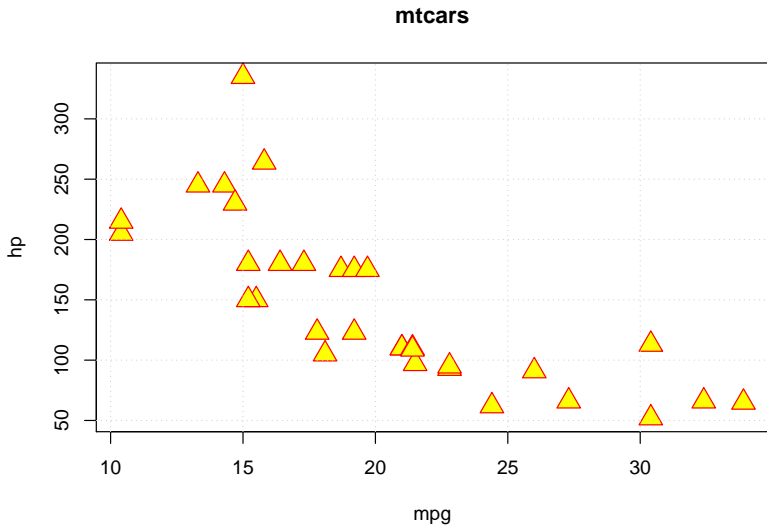
Nokta (Scatter) Grafik



Nokta (Scatter) Grafik - grid()

```
plot(x$mpg, x$hp,  
     main = "mtcars",  
     xlab = "mpg",  
     ylab = "hp",  
     pch = 24,  
     col = "red",  
     bg = "yellow",  
     cex = 2)  
grid()
```

Nokta (Scatter) Grafik - grid()

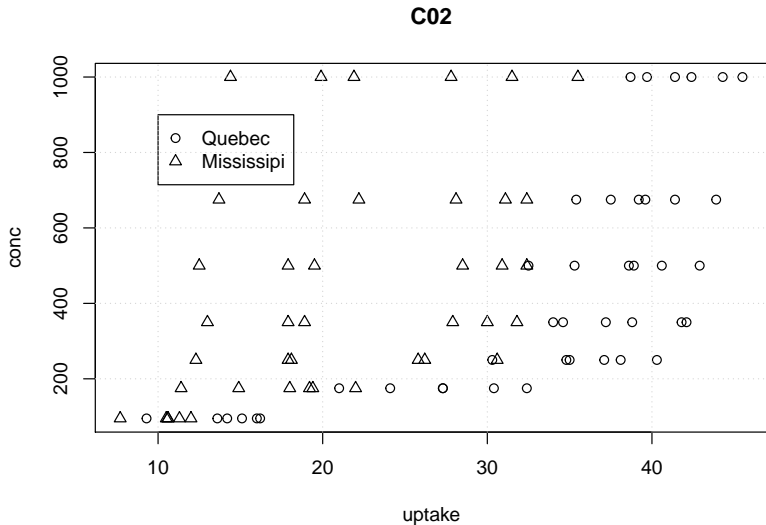


Nokta (Scatter) Grafik - legend()

- Farklı türde olan dataları ayırtırmak için `pch` argümanına faktörel bir değişken atanabilir.
- Eklenen üçgen ve/veya çember şekillerinin hangi datayı simgelediğini `legend()` fonksiyonu ile gösterilebilir.

```
x <- datasets::C02
plot(x$uptake, x$conc,
     main = "C02", xlab = "uptake", ylab = "conc",
     pch = as.integer(x$Type))
legend(x = 10, y = 900, c("Quebec", "Mississippi"),
      pch = 1:2)
grid()
```

Nokta (Scatter) Grafik - legend()

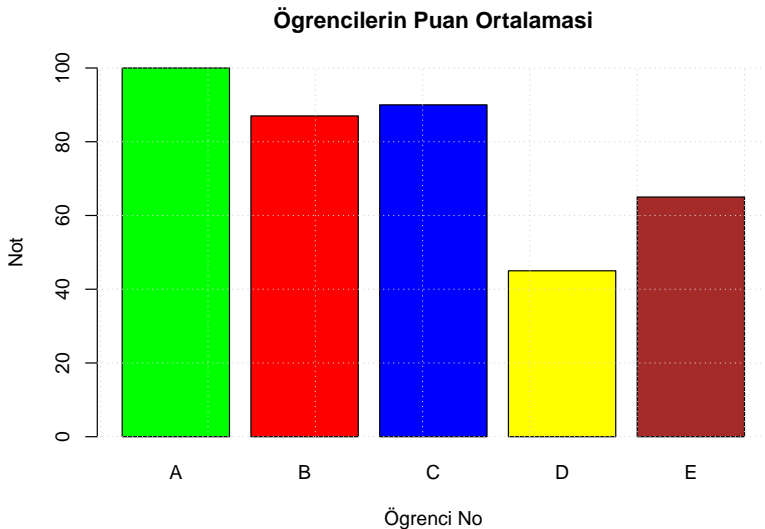


Sütun (Bar) Grafik

```
x <- c(100, 87, 90, 45, 65)
names(x) <- c("A","B","C","D","E")

barplot(x,
        main = "Öğrencilerin Puan Ortalaması",
        ylim = c(0,100),
        xlab = "Öğrenci No",
        ylab = "Not",
        col = c("green","red","blue","yellow","brown"))
grid()
```

Sütun (Bar) Grafik



Çizgi (Line) Grafik

- `plot()` fonksiyonu **type** argümanı olarak "l" değeri aldığı anda çizgi grafik oluşturacaktır.
- Type argümanının alabileceği değerler şunlardır.

```
# l : çizgi  
# p : noktalar  
# o : noktalı çizgi (overplotted)  
# b : noktalı çizgi (çizgiler noktalara dokunmaz)  
# c : noktalı çizgi (noktalar boş kalır)  
# s : merdiven basamakları  
# h : histogram çizgileri
```

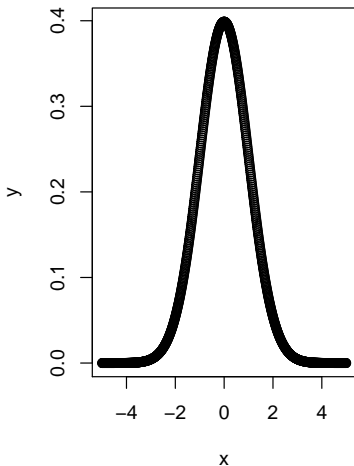
Çizgi (Line) Grafik

```
x <- seq(from = -5, to = 5, by = 0.01)
y <- dnorm(x)

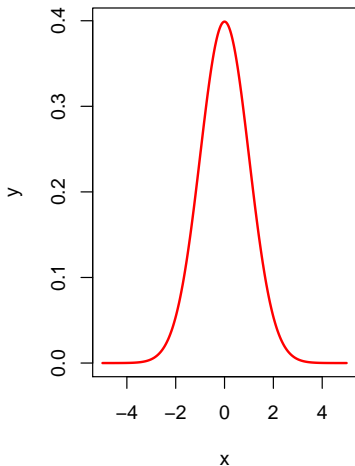
par(mfrow = c(1,2))
plot(x, y, main = "Nokta Grafik")
plot(x, y, type = "l", main = "Çizgi Grafik",
      lty = 1,
      lwd = 2,
      col = "red")
```


Çizgi (Line) Grafik

Nokta Grafik



Çizgi Grafik



```
h1_res <- dbSendQuery(conn,
  paste0("SELECT data_date, close FROM parity
        WHERE asset_code ='EUR/TRY' and
        data_date between '2015-02-01'
        and '2015-03-01';"))

h1 <- dbFetch(h1_res)

h2_res <- dbSendQuery(conn,
  paste0("SELECT data_date, close FROM parity
        WHERE asset_code ='USD/TRY' and
        data_date between '2015-02-01'
        and '2015-03-01';"))
```

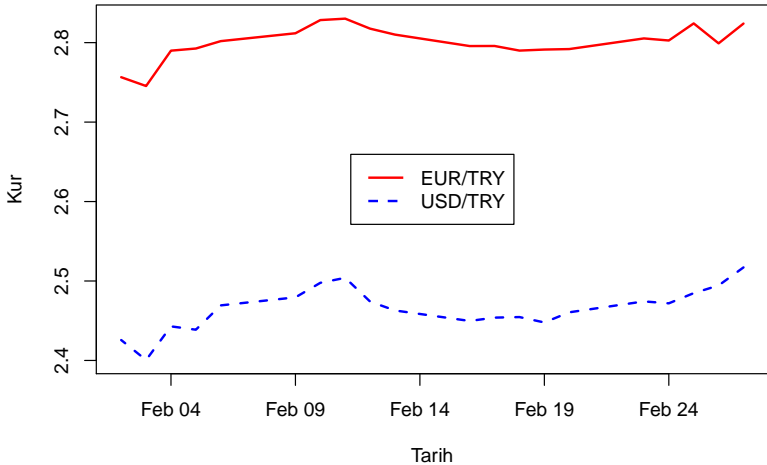
```
## Warning in new_result(connection@ptr, statement): Cancel
```

```
h2 <- dbFetch(h2_res)
```

Çoklu Grafik

```
plot(h1$data_date, h1$close,
     type = "l", lty = 1, lwd = 2, col = "red",
     ylim = c(min(h1$close,h2$close),
               max(h1$close,h2$close)
             ),
     xlab = "Tarih",
     ylab = "Kur")
lines(h2$data_date, h2$close,
      type = "l", lty = 2, lwd = 2, col = "blue")
legend("center", c("EUR/TRY","USD/TRY"),
      col = c("red","blue"), lwd = c(2,2), lty = c(1,2))
```

Çoklu Grafik



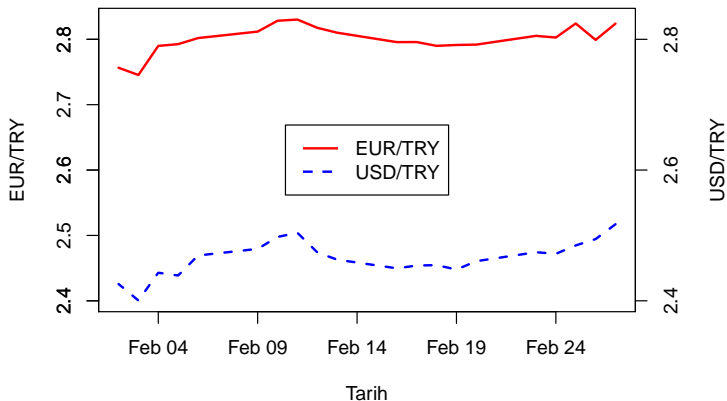
- `par(mar = c(x,y,z,q))` fonksiyonu ile grafik çerçevesi etrafındaki boşluklar ayarlanmaktadır.

```
par(mar = c(6,6,6,6))
plot(h1$data_date, h1$close,
     type = "l", lty = 1, lwd = 2, col = "red",
     ylim = c(min(h1$close,h2$close),
               max(h1$close,h2$close)),
     xlab = "Tarih", ylab = "")
axis(2, pretty(c(0.8*min(h1$close, na.rm = TRUE),
                  1.1*max(h1$close, na.rm = TRUE))))
mtext(2, text = "EUR/TRY", line = 3)
```

```
lines(h2$data_date, h2$close,
      type = "l", lty = 2, lwd = 2, col = "blue")
axis(4, pretty(c(0.8*min(h2$close, na.rm = TRUE),
                  1.1*max(h2$close, na.rm = TRUE))))
mtext(4, text = "USD/TRY", line = 3)

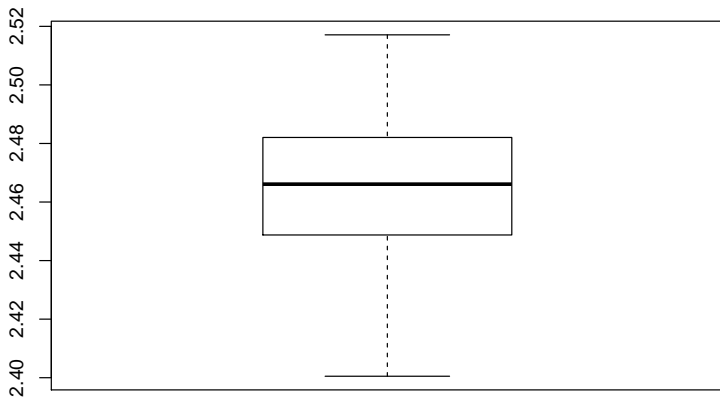
legend("center", c("EUR/TRY", "USD/TRY"),
      col = c("red", "blue"), lwd = c(2,2), lty = c(1,2))
```

Çoklu Grafik



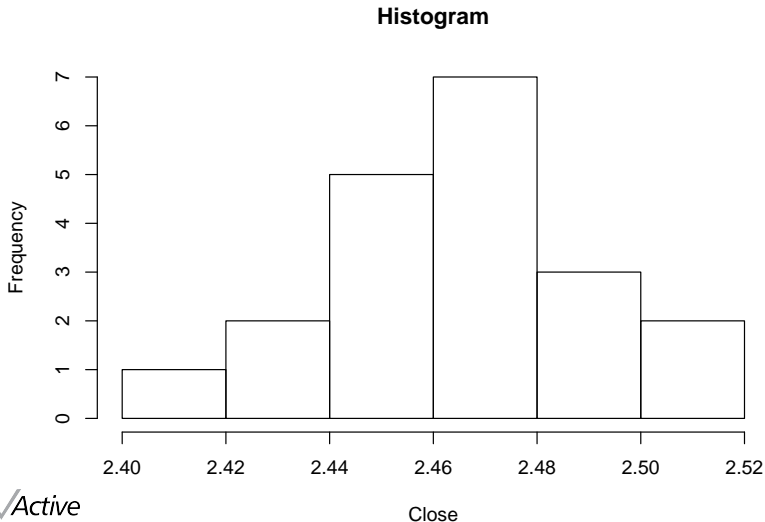
Kutu (Box) Grafik

```
boxplot(h2$close)
```



Histogram ve Dağılım Yoğunluk Çizimleri

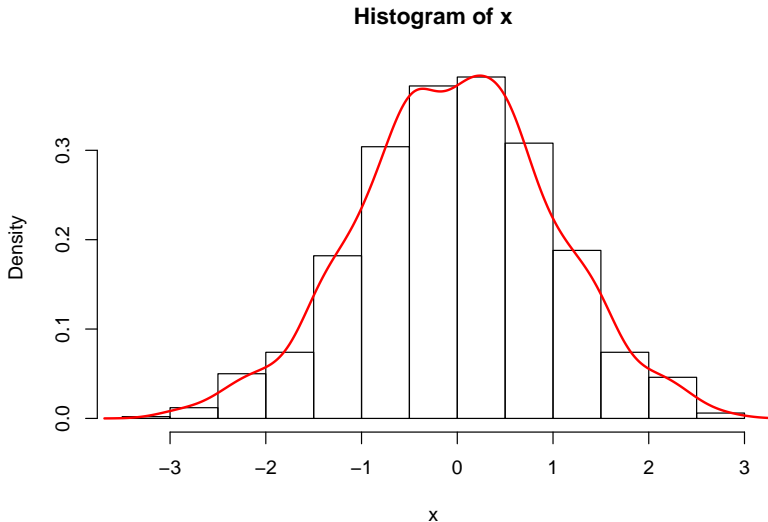
```
hist(h2$close, 5, xlab = "Close", main = "Histogram")
```



Histogram ve Dağılım Yoğunluk Çizimleri

```
x <- rnorm(1000)
hist(x, probability = TRUE)
lines(density(x), col = "red", lwd=2)
```

Histogram ve Dağılım Yoğunluk Çizimleri



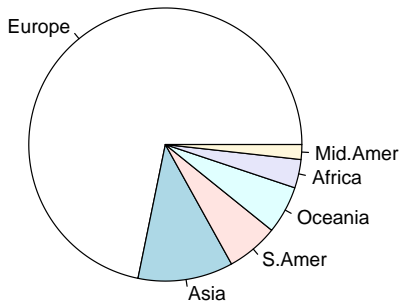
Pasta (Pie) Grafik

```
x <- datasets::WorldPhones  
head(x)
```

##		N.Amer	Europe	Asia	S.Amer	Oceania	Africa	Mid.Amer
##	1951	45939	21574	2876	1815	1646	89	555
##	1956	60423	29990	4708	2568	2366	1411	733
##	1957	64721	32510	5230	2695	2526	1546	773
##	1958	68484	35218	6662	2845	2691	1663	836
##	1959	71799	37598	6856	3000	2868	1769	911
##	1960	76036	40341	8220	3145	3054	1905	1008

Pasta (Pie) Grafik

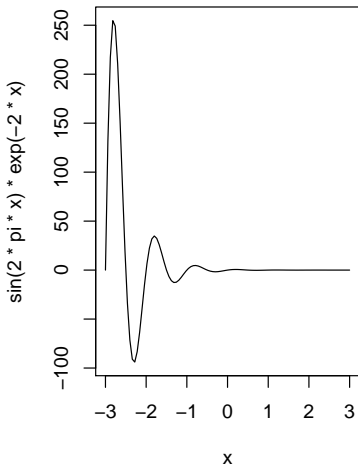
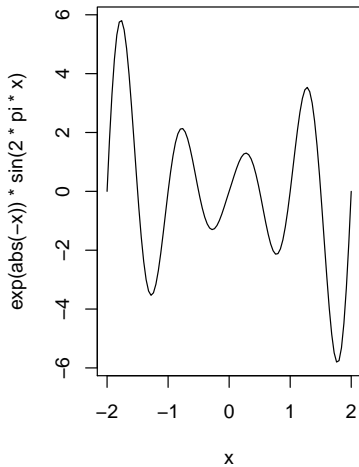
```
pie(x[2,2:7])
```



Fonksiyon Çizimi

```
par(mfrow = c(1,2))  
curve(exp(abs(-x)) * sin(2 * pi * x), -2, 2)  
curve(sin(2*pi*x)*exp(-2*x), -3, 3)
```

Fonksiyon Çizimi



“ggplot2” ile Grafik Çizimi

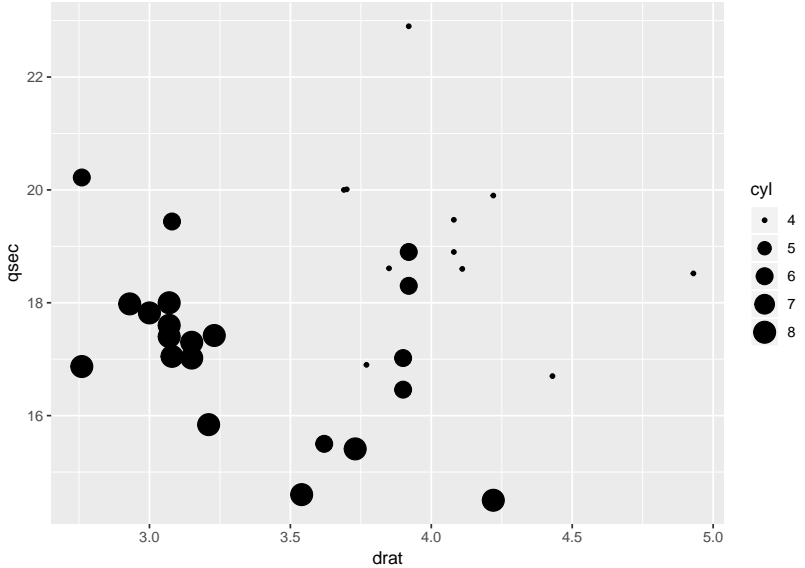
- ggplot2, istatistiksel veya veri grafikleri üretmek için geliştirilmiştir.
- Kendisine ait bir grameri bulunmaktadır. `geom` adı verilen katmanlar eklenerek oluşmaktadır.
- Geom, nokta, çizgi veya sütun grafik; eksen ismi, başlık gibi estetik özellikler olabilir.

```
library(ggplot2)
```

Nokta (Scatter) Grafik

```
ggplot(mtcars, aes(x = drat, y = qsec,  
                  size = cyl)) +  
  geom_point()
```

Nokta (Scatter) Grafik

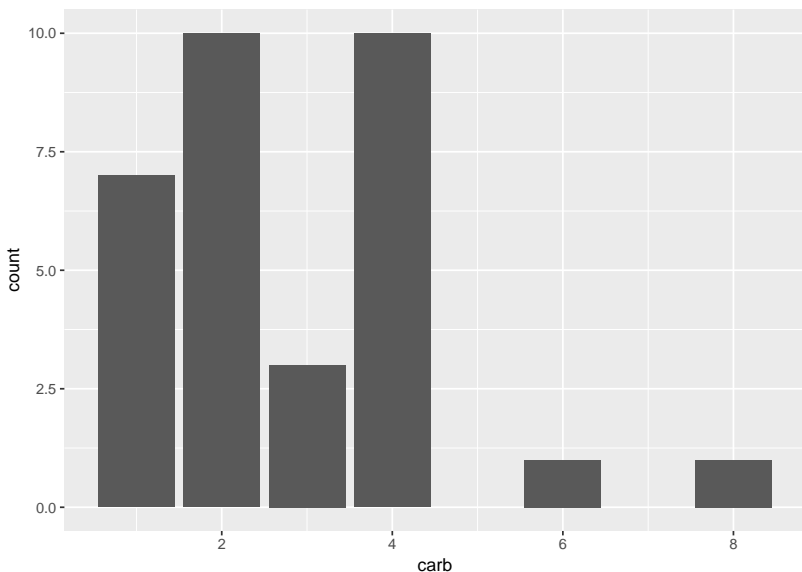


Sütun (Bar) Grafik

- carb değişkeni numerik bir değişkendir.

```
ggplot(mtcars, aes(x = carb)) +  
  geom_bar()
```

Sütun (Bar) Grafik

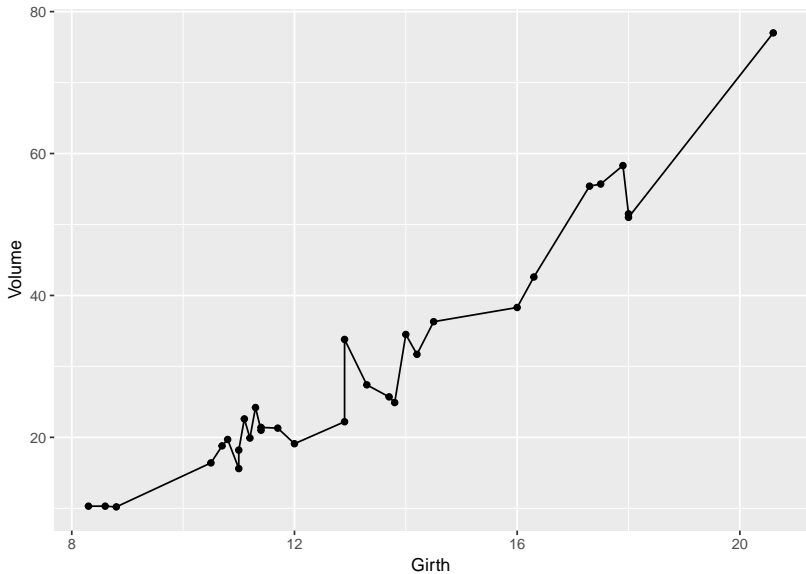


Çizgi (Line) Grafik

- **trees** veri seti 31 adet kesilmiş vişne ağaçlarının çapı, yüksekliği ve hacmini göz önüne almaktadır.

```
ggplot(trees, aes(x = Girth, y = Volume)) +  
  geom_line() +  
  geom_point()
```

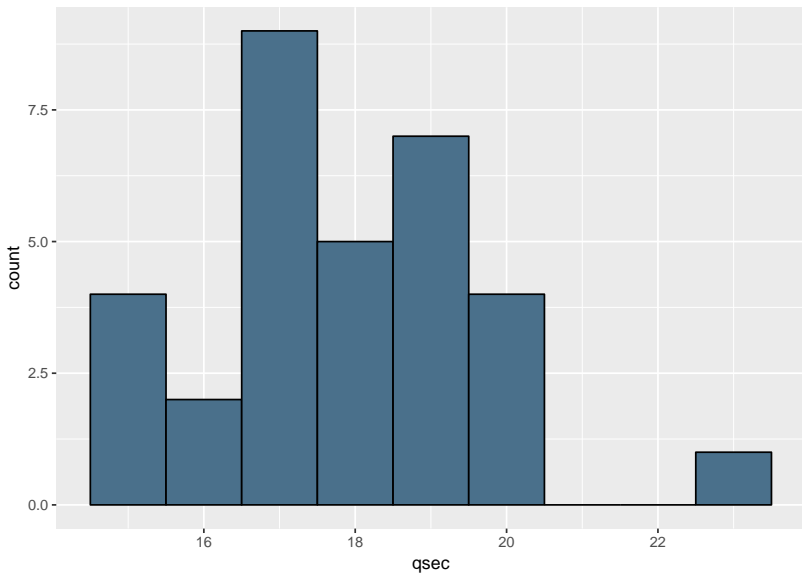
Çizgi (Line) Grafik



Histogram

```
ggplot(mtcars, aes(x = qsec)) +  
  geom_histogram(binwidth = 1, #sütun genişliği  
                 fill = "skyblue4", #iç dolgu rengi  
                 col = "black") #dış çizgi rengi
```

Histogram



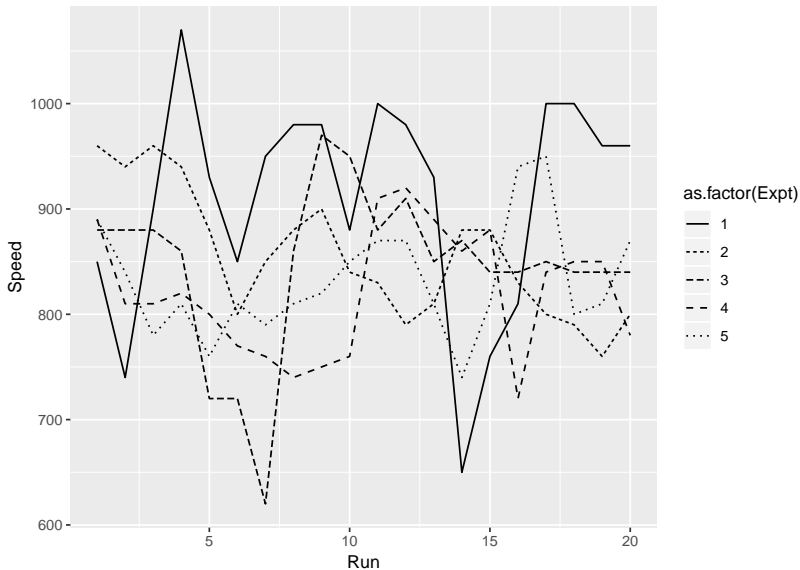
- Farklı deneylerde ölçülmüş ışık hızları ile ilgili örnek **morley** datasını göz önüne alalım.

```
head(morley)
```

##		Expt	Run	Speed
##	001	1	1	850
##	002	1	2	740
##	003	1	3	900
##	004	1	4	1070
##	005	1	5	930
##	006	1	6	850

```
ggplot(morley, aes(x = Run, y = Speed)) +  
  geom_line(aes(linetype = as.factor(Expt)))
```

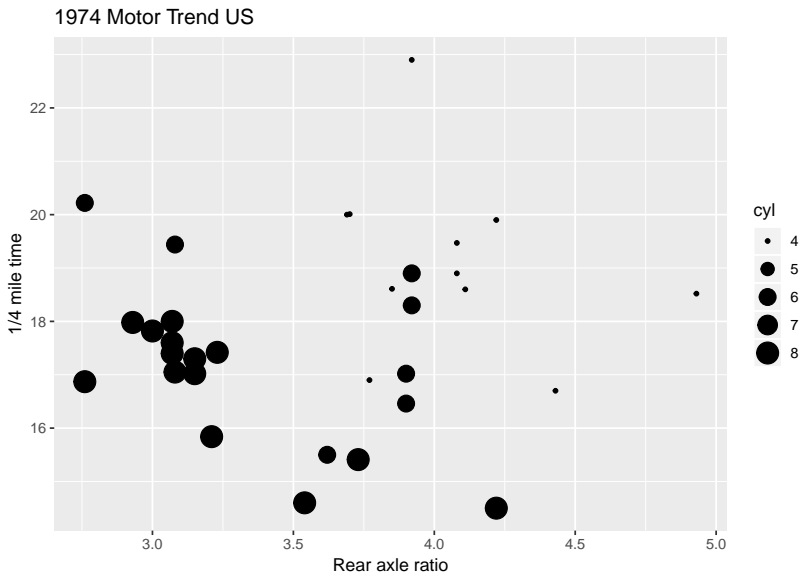
Çoklu Çizim



Başlık ve Eksen İşlemleri

```
p <- ggplot(mtcars, aes(x = drat, y = qsec,  
                        size = cyl))  
  
p <- p + geom_point() +  
  ggtitle("1974 Motor Trend US") +  
  xlab("Rear axle ratio") +  
  ylab("1/4 mile time")
```

Başlık ve Eksen İşlemleri

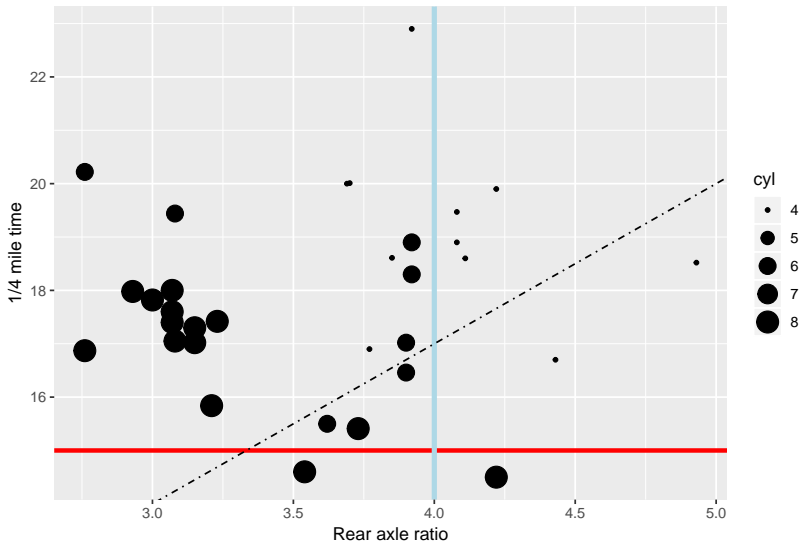


Grafiğe Doğru Ekleme

```
p + geom_hline(yintercept = 15,  
               colour="red", size = 1.3) +  
  geom_vline(xintercept = 4,  
             color = "lightblue", size = 1.5) +  
  geom_abline(intercept = 5,  
             slope = 3, linetype = "dotdash")
```


Grafiğe Doğru Ekleme

1974 Motor Trend US



Teşekkürler 😊

- tunc.oygur@riskactive.com
- [linkedin.com/in/TuncOygur](https://www.linkedin.com/in/TuncOygur)