

Data Import : : HATIRLATICI NOT



R'nin **tidyverse** paketi **tidy data**'da saklanan, ve data frame'in gelişmiş bir hali diyebileceğimiz, **tibbles**'lar üzerine kurulmuştur.



Bu notun ilk sayfası metin dosyalarının **readr** paketiyle nasıl okunduğunu göstermektedir.



Notun ikinci sayfası ise, **tibble** ile nasıl tibble oluşturulacağını ve **tidyr** ile düzenli verinin nasıl tertipleneceğini göstermektedir.

Diğer Veri Tipleri

Aşağıdaki paketleri kullanarak farklı tipteki dosyaları import edebilirsiniz

- **haven** - SPSS, Stata, ve SAS dosyaları
- **readxl** - Excel dosyaları (.xls ve .xlsx)
- **DBI** - veri tabanları
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Veriyi Kaydetmek

x'i R objesi olarak dosya **dizine** (path) kaydetmek:

Virgüle ayrılmış dosya (CSV)

write_csv(x, path, na = "NA", append = FALSE, col_names = !append)

İsteğe bağlı ayrılmış dosya

write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = !append)

Excel için CSV

write_excel_csv(x, path, na = "NA", append = FALSE, col_names = !append)

Dosyaya string (dizi) olarak

write_file(x, path, append = FALSE)

Dosyaya string vector olarak, satır başına tek eleman

write_lines(x, path, na = "NA", append = FALSE)

RDS dosyası olarak

write_rds(x, path, compress = c("none", "gz", "bz2", "xz"), ...)

Tab ile ayrılmış dosya (TSV)

write_tsv(x, path, na = "NA", append = FALSE, col_names = !append)

Tabular Veriyi Okumak

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())

a,b,c
1,2,3
4,5,NA

| A | B | C |
|---|---|----|
| 1 | 2 | 3 |
| 4 | 5 | NA |

Virgüle ayrılmış dosya (CSV)

read_csv("file.csv")

file.csv dosyasını çalıştırmak için:

write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

a;b;c
1;2;3
4;5;NA

| A | B | C |
|---|---|----|
| 1 | 2 | 3 |
| 4 | 5 | NA |

Noktalı virgüle ayrılmış dosya

read_csv2("file2.csv")

write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

a|b|c
1|2|3
4|5|NA

| A | B | C |
|---|---|----|
| 1 | 2 | 3 |
| 4 | 5 | NA |

Herhangi bir ayırıcı ile ayrılmış dosya

read_delim("file.txt", delim = "|")

write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")

a b c
1 2 3
4 5 NA

| A | B | C |
|---|---|----|
| 1 | 2 | 3 |
| 4 | 5 | NA |

Sabit Genişlikli dosyalar

read_fwf("file.fwf", col_positions = c(1, 3, 5))

write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab ile ayrılmış dosyalar (TSV)

read_tsv("file.tsv") Ayrıca **read_table**()

write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

BAZİ YARARLI FONKSİYON ARGUMANLARI

a,b,c
1,2,3
4,5,NA

Örnek Dosya

write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

| 1 | 2 | 3 |
|---|---|----|
| 4 | 5 | NA |

Satırları atla

read_csv(f, **skip** = 1)

| A | B | C |
|---|---|----|
| 1 | 2 | 3 |
| 4 | 5 | NA |

Başlıksız

read_csv(f, **col_names** = FALSE)

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |

Altküme içinde oku

read_csv(f, **n_max** = 1)

| x | y | z |
|---|---|----|
| A | B | C |
| 1 | 2 | 3 |
| 4 | 5 | NA |

Başlık ekle

read_csv(f, **col_names** = c("x", "y", "z"))

| A | B | C |
|----|---|----|
| NA | 2 | 3 |
| 4 | 5 | NA |

Kayıp değerler

read_csv(f, **na** = c("1", ""))

Tabular Olmayan Veriyi Okuma

Dosyayı tek bir string olarak okur

read_file(file, locale = default_locale())

Herbir satırı kendi stringi içinde okur

read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive())

Apache tipindeki log dosyalarını okur

read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive())

Dosyayı işlenmemiş bir vektör olarak okur

read_file_raw(file)

Herbir satırı işlenmemiş vektör içinde okur

read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive())

Veri türleri

readr fonksiyonları her bir sütunun tipini tahmin eder ve gerektiğinde türlerini değiştirir (ancak otomatik olarak string (dizi) değerlerini faktör olarak DEĞİŞTİRMEZ.)

Aşağıdaki gibi bir mesaj sütun tanımlamalarını verecektir.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

yaş (age)
tamsayı

cinsiyet(sex)
karakter

earn ise double (nümerik)

1. Problemleri tanılamak için **problems()** kullanın
x <- read_csv("file.csv"); problems(x)

2. Ayırıştırmak (parsing) için col_ fonk. kullanın

- **col_guess()** - varsayılan
- **col_character()**
- **col_double()**, **col_euro_double()**
- **col_datetime**(format = "") Ayrıca **col_date**(format = ""), **col_time**(format = "")
- **col_factor**(levels, ordered = FALSE)
- **col_integer()**
- **col_logical()**
- **col_number()**, **col_numeric()**
- **col_skip()**

```
x <- read_csv("file.csv", col_types = cols(
  A = col_double(),
  B = col_logical(),
  C = col_factor()))
```

3. Veyahut, karakter vektörü olarak okuyun, sonrasında **parse_fonksiyonuyla** ayırıştırın.

- **parse_guess()**
- **parse_character()**
- **parse_datetime()** Ayrıca **parse_date()** ve **parse_time()**
- **parse_double()**
- **parse_factor()**
- **parse_integer()**
- **parse_logical()**
- **parse_number()**
- **x\$A <- parse_number(x\$A)**



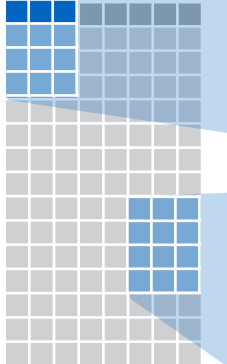
Tibble - geliştirilmiş data frame

Tibble paketi tabular datayı depolamak için yeni bir S3 sınıflandırması temin eder. Tibblelar data frame sınıflandırmasını devralmanın yanı sıra, ondan üç farklı davranışla ayrılır:

- **Subsetting** (*altkümleme*) - [her zaman yeni bir tibble yaratır, [[ve \$ ise yeni bir vektör yaratır.

- **Kısmi eşleme olmaması** - Subsetting yaparken her zaman tam sütun isimleri kullanılmalıdır.

- **Gösterim** - Bir tibble'ı çalıştırdığınızda, R size, verinin tek bir ekrana sığdığını gösteren kısa bir özet verir.



Örnek büyük bir tablo üzerinden

```
# A tibble: 234 x 6
  manufacturer <chr>   model <chr>   displ <dbl>
1 audi         a4         1.8
2 audi         a4         2.0
3 audi         a4         2.0
4 audi         a4         2.8
5 audi         a4         3.1
6 audi         a4 quattro 1.8
7 audi         a4 quattro 1.8
8 audi         a4 quattro 2.0
... with 224 more rows, and
3 more variables: year <int>,
  cyl <int>, trans <chr>
```

tibble gösterimi

```
156 1999 6 auto(l4)
157 1999 6 auto(l4)
158 2008 6 auto(l4)
159 2008 8 auto(s4)
160 1999 4 manual(m5)
161 1999 4 auto(l4)
162 2008 4 manual(m5)
163 2008 4 manual(m5)
164 2008 4 auto(l4)
165 2008 4 auto(l4)
166 1999 4 auto(l4)
[ reached getOption("max.print")
  -- omitted 68 rows ]
```

data frame gösterimi

- Varsayılan görünüm options ile değiştirilebilir:

options(tibble.print_max = n,
tibble.print_min = m, tibble.width = Inf)

- Tam veri setini **View()** ya da **glimpse()** ile görebilirsiniz.

- Data frame'e çevirmek için: **as.data.frame()**

İKİ ŞEKİLDE TIBBLE OLUŞTURMAK

tibble(...)
Sütunlar üzerine inşa,
`tibble(x = 1:3, y = c("a", "b", "c"))`

tribble(...)
Satırlar üzerine inşa,
`tribble(~x, ~y, 1, "a", 2, "b", 3, "c")`

İkisi birlikte bu tibble'ı oluştur

```
A tibble: 3 x 2
  x     y
<int> <dbl>
1     1  a
2     2  b
3     3  c
```

as_tibble(x, ...) Data frame'i tibble'a dönüştürür.

enframe(x, name = "name", value = "value")
İsmlendirilmiş vektörü tibble'a dönüştürür.

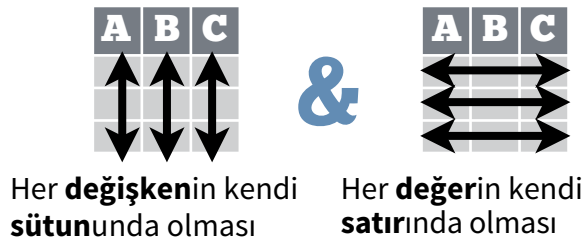
is_tibble(x) x'in bir tibble olup olmadığını test eder.



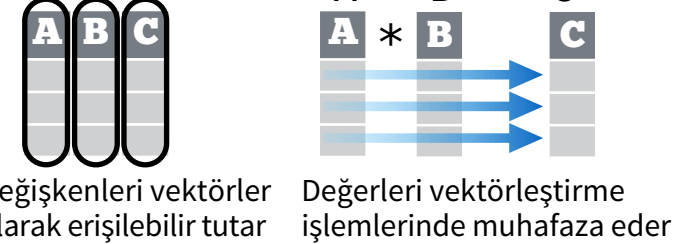
Tidyr ile Tidy Data (Düzenli Veri)

Tidy data tabular veriyi organize etmenin bir şeklidir. Paketler arası birbiriyle tutarlı veri yapıları sağlar.

Bir tablonun tidy olabilmesi için:



Tidy data:

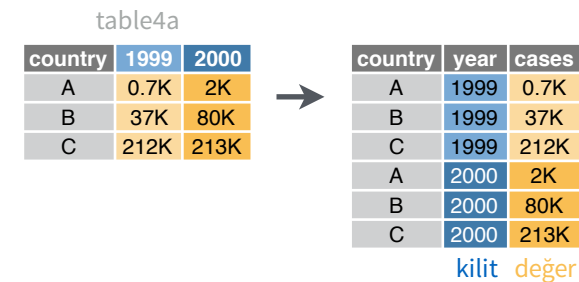


Veriyi Yeniden Şekillendirmek - tablodaki değer düzenlerini değiştirme

Tablodaki değerleri yeniden düzenlemek için **gather()** ve **spread()** kullanın.

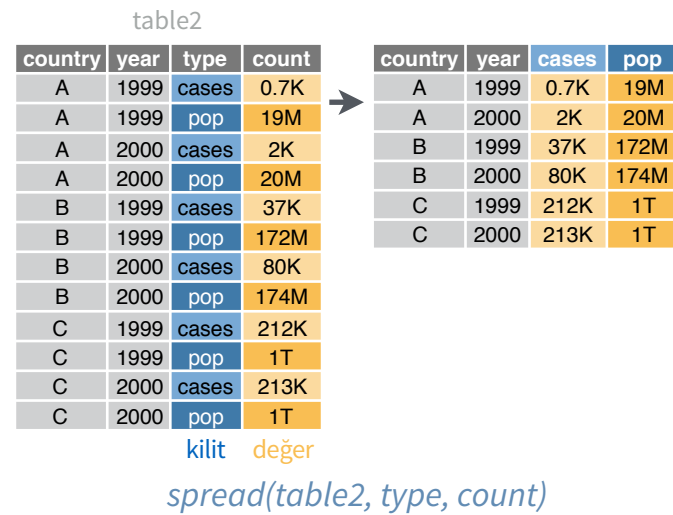
gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE)

Gather sütun isimlerini **kilit** sütuna taşır, yani tüm sütun değerlerini tek bir **değer** sütununda toplar.



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

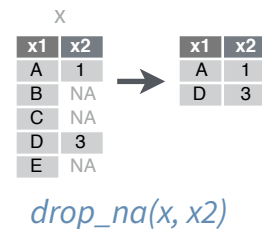
Spread **kilit** sütunun eşsiz (*unique*) değerlerini sütun isimlerine dağıtır, yani bir değer sütunundaki hücre **değerleri** yeni sütunlara neşrolacaktır.



Kayıp Değerleri (NA) Yönetmek

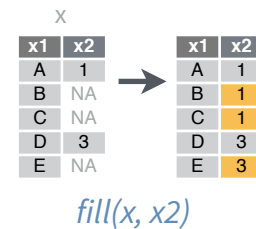
drop_na(data, ...)

... sütunlarında NA değerlerini içeren satırları eksiltir.



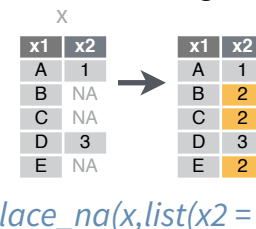
fill(data, ..., .direction = c("down", "up"))

... sütunlarında NA'ları en son NA olmayan değerlerle doldurur.



replace_na(data, replace = list(), ...)

NA'ları sütuna göre değiştirir.



Tabloları Genişletmek - değer kombinasyonlarıyla çabuk tablolar yaratma

complete(data, ..., fill = list())

... listelenen değişkenlerin değerlerinin kayıp kombinasyonlarını veriye ekler.

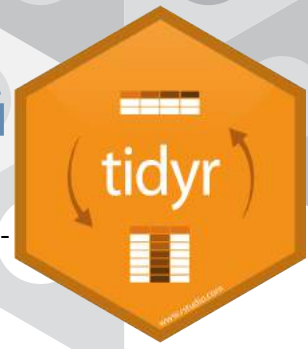
complete(mtcars, cyl, gear, carb)

expand(data, ...)

... listelenen değişkenlerin değerlerinin mümkün olan tüm kombinasyonlarla tibble yaratır.

expand(mtcars, cyl, gear, carb)

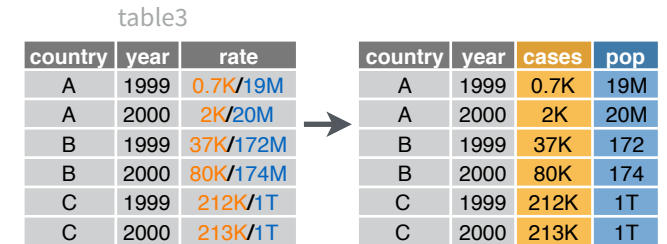
Hücre Bölümü



Hücreleri tekil izole değerlere bölmek ya da birleştirmek için buradaki fonksiyonları kullanın.

separate(data, col, into, sep = "[^:alnum:]", +, remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...)

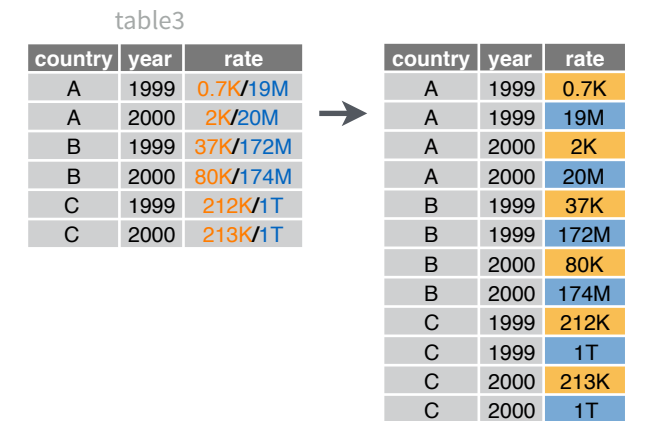
Bir sütundaki hücreleri sınıfına göre ayrı sütunlara ayırmak.



separate(table3, rate, into = c("cases", "pop"))

separate_rows(data, ..., sep = "[^:alnum:]", +, convert = FALSE)

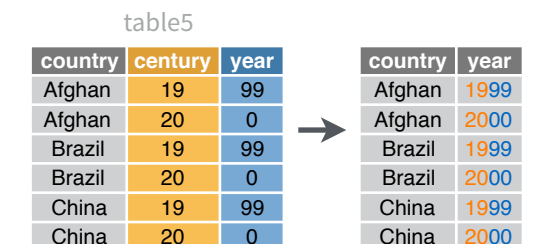
Bir sütundaki her hücreyi ayrı satırlara ayırmak. Ayrıca **separate_rows_()**.



separate_rows(table3, rate)

unite(data, col, ..., sep = "_", remove = TRUE)

Hücreleri tek sütunda toplamak için ayrı sütunları birleştirmek.



unite(table5, century, year, col = "year", sep = "")