dplyr ile Veri Manipülasyonu: : REFERANS KAĞIDI

*Bu dökümanda **pipe**, **zincir** olarak çevrilmiştir. Çevirmenleriniz "boru" operatörü yerine zincir operatörü yazmayı daha mantıklı bulmaktadırlar.

dplyr fonksiyonları zincirler ile çalışır ve **düzenli veri** ister. Düzenli veride:

Her **değişken**in kendi **sütunu** vardır

&

Her **gözlem** veya **vaka** kendi **satır**ında bulunur.

pipes* (zincir)

x %>% f(y) sonucu **f(x, y)**

Vaka Özetleme (summarise)

Bu fonksiyonlar ilgili sütunların aynı bir pivot tablosu gibi işlemekte ve özet tablolar çıkarmaktadır. Özet fonksiyonları vektörleri girdi olarak alıp tek değer sonucu verirler (diğer sayfaya bakın).

özet (summary) fonksiyonu

summarise(.data, …)
Özet tablosunu oluştur. Ayrıca bakınız **summarise_**().
summarise(mtcars, avg = mean(mpg))

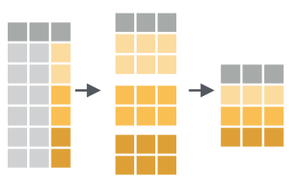
count(x, ..., wt = NULL, sort = FALSE)
... ile belirlenen değişkenlerden oluşturulan her gruptaki satır sayısını hesaplar. Ayrıca bkz
tally().
count(iris, Species)

VARIATIONS

summarise_all() - Fonksiyonları her sütuna uygula.
summarise_at() - Fonksiyonları belirtilen sütunlara uygula.
summarise_if() - Fonksiyonları ilgili veri tipindeki sütunlara uygula.

Vakaları Gruplama

Bir tablonun "gruplanmış" halini oluşturmak için **group_by()** kullanın.
dplyr fonksiyonları her "grup"ta ayrı ayrı işlem yapıp sonuçları birleştirir.



```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

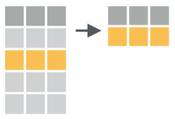
group_by(.data, ..., add = FALSE**)**
aynı tablonun ... değişkenleriyle kopyalanmış grubunu verir.
g_iris <- group_by(iris, Species)

ungroup(x, …**)**
tablonun grupsuz halini verir.
ungroup(g_iris)

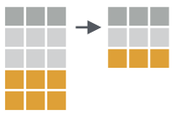
Vaka Manipülasyonu

VAKA ÇIKARMA

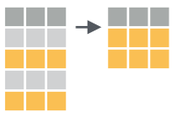
Satır fonksiyonları, tablonun istenen bir alt kümesini verir. "_" ile biten versiyonlarında *standart olmayan değerlendirme* (NSE) yöntemleri rahatça kullanılabilir.



filter(.data, …**)** Kural bazlı komutlara göre uygun satırları getirir. Ayrıca **filter_()**. *filter(iris, Sepal.Length > 7)*



distinct(.data, ..., .keep_all = FALSE**)** Tıpa tıp aynı değerleri olan satırları teke indirir. Ayrıca **distinct_()**. *distinct(iris, Species)*



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)** Satırların belli bir yüzdesini rastgele seçer.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Belli sayıda rastgele satır seçer. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …**)** Sıra numarasına göre satır seçer. Ayrıca **slice_()**. *slice(iris, 10:15)*

top_n(x, n, wt**)** En üst n satırı seçer ve sıralar. (gruplu veride gruba göre). *top_n(iris, 5, Sepal.Width)*

filter() ile kullanılabilecek mantık ve bool işlemleri

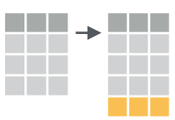
<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

Konsola **?base::logic** ve **?Comparison** yazarak yardım alabilirsiniz.

VAKALARI SIRALAMA

arrange(.data, …**)** Satırları, seçili sütun(lar)daki değerlere göre sırala (küçükten büyüğe), **desc()** kullanarak büyükten küçüğe sırala.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

VAKA EKLEME



add_row(.data, ..., .before = NULL, .after = NULL**)**
Bir tabloya bir veya birden fazla satır ekle.
add_row(faithful, eruptions = 1, waiting = 1)

SÜTUN SEÇME

Bir tablonun istenilen sütunlarını seçip yeni bir tablo oluşturma. "_" ile biten versiyonlarında *standart olmayan değerlendirme* (NSE) yöntemleri rahatça kullanılabilir.

select(.data, …**)**
İsimlerine göre sütun seçme.
Ayrıca **select_if()**
select(iris, Sepal.Length, Species)

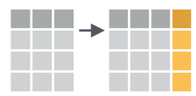
select() içinde bu fonksiyonları ve operatörleri kullanabilirsiniz:
ör. select(iris, starts_with("Sepal"))

contains(match**)**	**num_range(**prefix, range**)**	**:**, ör. mpg:cyl
ends_with(match**)**	**one_of(**…**)**	**-**, ör., -Species
matches(match**)**	**starts_with(**match**)**	

YENİ DEĞİŞKENLER (SÜTUNLAR) OLUŞTURMA

Bu komutlar, sütunlara **vektörize fonksiyonlar** uygularlar. Vektörize fonksiyonlar vektörleri girdi olarak alıp aynı uzunlukta vektörleri çıktı olarak verirler. (bkz. arka sayfa)

vektörize fonksiyonlar



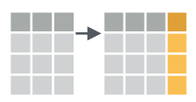
mutate(.data, …**)**
Yeni sütun(lar) hesapla.
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …**)**
Sadece yeni hesaplanan sütunları bırak.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …**)** Fonksiyonları her sütuna uygula. **funs()** ile birlikte kullan.
mutate_all(faithful, funs(log(.), log2(.)))

mutate_at(.tbl, .cols, .funs, …**)** Fonksiyonları belli sütunlara uygula. **funs()**, **vars()** ve select() yardımcı fonksiyonları ile kullan.
mutate_at(iris, vars(-Species), funs(log(.)))

mutate_if(.tbl, .predicate, .funs, …**)**
Fonksiyonları belli tipteki sütunlara uygula.
funs() ile birlikte kullan.
mutate_if(iris, is.numeric, funs(log(.)))



add_column(.data, ..., .before = NULL, .after = NULL**)** Yeni sütun(lar) ekle.
add_column(mtcars, new = 1:32)

rename(.data, …**)** Sütunları yeniden adlandır.
rename(iris, Length = Sepal.Length)

Vektörize Fonksiyonlar

MUTATE() İLE BİRLİKTE KULLANMAK İÇİN

mutate() ve **transmute()** yeni sütunlar oluşturmak için vektörize fonksiyonlar uygularlar. Vektörize fonksiyonlar vektörleri girdi olarak alır ve aynı uzunlukta vektörleri çıktı olarak verirler.

vektörize fonksiyon

KAYDIRMALAR

dplyr::**lag()** - Değerleri 1 satır kaydırır
dplyr::**lead()** - Değerleri -1 satır kaydırır

TOPLU KÜMÜLATİF İŞLEMLER

dplyr::**cumall()** - Kümülatif all()
dplyr::**cumany()** - Kümülatif any()
cummax() - Kümülatif max()
dplyr::**cummean()** - Kümülatif mean()
cummin() - Kümülatif min()
cumprod() - Kümülatif prod()
cumsum() - Kümülatif sum()

SIRALAMALAR

dplyr::**cume_dist()** -Bütün değerlerin oranları <=
dplyr::**dense_rank()** - sıralama, beraberlikte = min, boşluk olmadan
dplyr::**min_rank()** - sıralama, beraberlikte = min
dplyr::**ntile()** - n kadar gruba gruplar
dplyr::**percent_rank()** - min_rank [0,1] ölçeği
dplyr::**row_number()** - sıralama, berab. = "ilk"

MATEMATİK İŞLEMLERİ

+, - , *, /, ^, %/%, %% - aritmetik
log(), log2(), log10() - logaritma
<, <=, >, >=, !=, == - mantıksal işlemler

DİĞER

dplyr::**between()** - x >= sol & x <= sağ
dplyr::**case_when()** - if_else() in çoklu hali
dplyr::**coalesce()** - bir grup vektörde ilk NA olmayan değerleri getir
dplyr::**if_else()** - eleman eleman if() + else()
dplyr::**na_if()** - spesifik değerleri NA ile değiştir
pmax() - eleman eleman max()
pmin() - eleman eleman min()
dplyr::**recode()** - vektörize switch()
dplyr::**recode_factor()** - factor veri tipi için vektörize switch()

Özet Fonksiyonları

SUMMARISE() İLE BİRLİKTE KULLANMAK İÇİN

summarise() sütunlara özetleme fonksiyonları uygulayarak yeni bir tablo oluşturur. Özetleme fonksiyonları vektörleri girdi olarak alıp çıktı olarak tek değer verirler.

özet (summary) fonksiyonu

SAYIMLAR

dplyr::**n()** - değer/satır sayısı
dplyr::**n_distinct()** - tekil #
sum(!is.na()) - NA olmayan #

KONUM

mean() - ortalama, ayrıca **mean(!is.na())**
median() - ortanca (medyan)

MANTIKSALLAR

mean() - TRUE değerlerin ortalaması
sum() - TRUE değerlerin sayısı

KONUM/SIRALAMA

dplyr::**first()** - ilk değer
dplyr::last() - son değer
dplyr::nth() - vektörün **n**inci değeri

SIRA

quantile() - **n**inci kantil
min() - minimum değer
max() - maksimum değer

DAĞILIM

IQR() - Çeyreklik kantil aralığı
mad() - ortanca mutlak sapma
sd() - standart sapma
var() - varyans

Satır İsimleri

Düzenli veri sütun dışında değer taşıyan satır isimlerini kullanmaz. Satır isimlerini kullanmak için onları bir sütuna taşıyın.

rownames_to_column()
Satır isimlerini sütuna taşı.
a <- rownames_to_column(iris, var = "C")

column_to_rownames()
Sütunu satır ismine taşı.
column_to_rownames(a, var = "C")

Ayrıca **has_rownames()**, **remove_rownames()**

Tabloları Birleştirme

DEĞİŞKENLERİ BİRLEŞTİRME

bind_cols() kullanarak tabloları olduğu gibi yan yana yapıştır.

bind_cols(…) yan yana konmuş tabloları tek bir tablo gibi yapar.

SATIR SAYISININ AYNI OLDUĞUNA EMİN OLUN.

"**Mutating Join**" kullanarak bir tablonun sütunlarını, diğer tablonunkiyle her satırdaki uyuşan değeriyle birleştir. Her birleşim türü tablo değerlerinin farklı bir kombinasyonunu ifade eder.

left_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),…**)**
y'deki uyuşan değerleri x ile birleştir.

right_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),…**)**
x'teki uyuşan değerleri y ile birleştir.

inner_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),…**)**
Birleşimde sadece uyuşan satırları getir.

full_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),…**)**
Birleşimde bütün değerleri ve satırları getir.

by = c("col1", "col2") kullanarak birleşim sütunlarını belirt.
left_join(x, y, by = "A")

İsimli bir vektör, **by = c("col1" = "col2")**, kullanarak farklı isimdeki sütunları aynıymış gibi birleştir.
left_join(x, y, by = c("C" = "D"))

suffix kullanarak aynı sütun isimlerine ek değer vererek ayrıştır.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

VAKALARI BİRLEŞTİRME

bind_rows() iki tabloyu olduğu gibi alt alta eklemeye yarar.

bind_rows(…, .id = NULL**)**
Tabloları tek bir tablo olarak getirir. *.id* 'yi bir kolon ismine atayın ve o satırların hangi dataframe'den geldiğini görün (örneğin, soldaki şekilde DF kolonu)

intersect(x, y, …)
Hem x hem y tablosunda beraber görünen değerler

setdiff(x, y, …)
x tablosunda olan ama y tablosudna olmayan değerler

union(x, y, …)
x ya da y tablosundaki satırlar (tekrarlı değerler çıkarılır)

union_all() tekrarlı değerleri tutar.

setequal() fonksiyonu ile iki tablo aynı satırları içeriyor mu kontrolü yapılır (Satırların sırası önemli değil).

SATIRLARI ÇIKARTMA

"**Filtering Join**" kullanarak bir tabloyu diğer tablonun satırlarına göre filtreleyin. NELERİN BİRLEŞTİRİLECEĞİNİ/BİRLEŞTİRİLMEYECEĞİNİ DENEYEREK GÖRÜN.

semi_join(x, y, by = NULL, …**)**
x tablosunun y tablosundaki kolonlarla eşleşen değerlerini getirir.

anti_join(x, y, by = NULL, …**)**
x tablosunun, y ile eşleşmeyen satırlarını getirir.