İstatistiksel Analiz Uygulamaları Final Ödevi

**02 Ocak 2022**

**Korona Verilerine İlişkin Veri Analizi**

**Onur Toka**

**Berca Akbayır: 21935577**

**Ramazan Erduran: 21821809**

**Mert Hasan Kılıç: 21936134**

**Talha Kurt: 21936191**

**İlkay Şafak Baytar: 2193571**

İçindekiler Tablosu

[Veri Manipülasyonu 1](#_Toc92017086)

[Veri Analizi 10](#_Toc92017087)

[Veri Görselleştirme 30](#_Toc92017088)

[ShinyR Web Reporting 47](#_Toc92017089)

[Kaynakça: 50](#_Toc92017090)

## Veri Manipülasyonu

**Tidyr Kütüphanesi(Düzenli Dağınık Veri):**

R’daki en önemli paketlerden biri tidyr paketidir. tidyr, ortak API’ler ve paylaşılan bir felsefe ile tasarlanmış bir paket ekosistemi olan tidyverse’nin bir parçasıdır. tidyr, reshape2 (2010-2014) ve reshape- (2005-2010) paketlerinin yerini almıştır.Tidyr, verilerinizi “düzenlemeyi” kolaylaştıran yani verilerimizi düzenli hale getirmemize yaran bir pakketir.Düzenli veriler, toplu evren boyunca mümkün olan her yerde kullanılan standart bir veri depolama yöntemini tanımlar. Düzenli verilerin en önemli iki özelliği şunlardır; Her sütun bir değişkendir, her satır bir gözlemdir,her hücre tek bir değerdir denebilir. Verilerinizi bu şekilde düzenlemek, değişkenlere (sütun adları olarak) ve gözlemlere (satır endeksleri olarak) atıfta bulunmak için tutarlı bir yolunuz olduğundan, Analiz yapmamız kolay bir hale gelir.

“tidyr”, “tidyverse” R paketleri koleksiyonun Düzenleme sağlayan “tibble” kütüphanesi gibi bir alt kütüphanesi olarak düşünülebilir.

Kurucu-Geliştirici-Destekçi : Hadley Wickham (Kurucu/Geliştirici-Tidyverse–RStudio)

R programının help menüsünden yararlanarak bu pakete baktığımızda ise; Her sütunun bir değişken, her satırın bir gözlem olduğu ve her hücrenin tek bir değer içerdiği düzenli veriler oluşturmaya yardımcı olacak araçlar. ‘tidyr’, bir veri kümesinin şeklini (pivoting) ve hiyerarşisini (yuvalama ve ‘yuvadan çıkarma’), derinlemesine iç içe geçmiş listeleri dikdörtgen veri çerçevelerine (‘dikdörtgenleme’) dönüştürmek ve dize sütunlarından değerleri çıkarmak için araçlar içerir. Ayrıca eksik değerlerle (hem örtük hem de açık) çalışmak için araçlar içerir yorumunu görmekteyiz.

**Kurulum**:

Bir paketi R programında kullanmak için önce paketin yüklenmesi gerekir. Tidyr paketinin yüklemesi aşağıdaki komut kullanılarak yapılabilir

**install.packages(“tidyr”)**

Daha sonra yukarıda bahsettiğimiz gibi Dağınık verileri düzenlemek için önce veri kümenizdeki değişkenleri tanımlarsınız, ardından bunları sütunlara taşımak için tidyr tarafından sağlanan araçları kullanırsınız.

**Tidyr işlevleri Beş Ana Kategoriye Ayrılır:**

-> Uzun ve geniş formlar arasında dönüşüm sağlar, bunu spread()ve gather() fonksiyonlarını kullanarak yapar. detyalı bilgi vermek gerekirse;

**gather() fonksiyonu**: Birden çok sütun alır ve bunları değer çiftlerinde toplar aslında; Geniş verileri daha uzun hale getirir. Bu işlev birden çok sütunu alır ve gerektiğinde diğer tüm sütunları çoğaltarak, anahtar-değer çiftleri içine atar.

**spread() fonksiyonu**: iki sütun (bir anahtar/değer çifti) alır ve bunları birden çok sütuna yayarak “uzun” verileri daha geniş hale getirir. Sütunlar yerine satırlar oluşturan değişkenleriniz olduğunda kullanılır.

Ayrıca spread()ve gather() fonksiyonları pivot\_longer()ve pivot\_wider() fonksiyonlarının yerini almıştır.

-> İç içe geçmiş listeleri düzenli tibble(data frame)’lara dönüştüren “Dikdörtgenleme”.bu işlemleri yaparken unnest\_longer(), unnest\_wider(), hoist() fonksiyonlarını kullanır.cDetaylı bilgi vermek gerekirse;

**unnest\_longer() fonksiyonu**: Bir liste sütununun her öğesini bir satıra dönüştürür.

**unnest\_wider() fonksiyonu**: Bir liste sütununun her öğesini bir sütuna dönüştürür.

**hoist() fonksiyonu**: Liste sütununun bileşenlerini seçerek kendi üst düzey sütunlarına çekmenize olanak tanır.

-> İç içe yerleştirme, gruplanmış verileri, her grubun iç içe geçmiş bir veri çerçevesi içeren tek bir satır haline geldiği bir forma dönüştürür ve verileri ayırmamızı sağlar. Bu işlemi yaparken nest(), unnest() fonksiyonlarını kullanır. Detaylı bilgi vermek gerekirse:

**nest() fonksiyonu**: Tüm iç içe değişkenleri içeren Data farme veri türnün bir listesini oluşturur. dolaylı olarak bir özetleme işlemidir. Bu, tüm veri kümeleriyle, özellikle de modellerle çalışan diğer özetlerle birlikte kullanışlıdır. Aslında nest() data frames şeklinden list-column’a dönüştürür.

**unnest() fonksiyonu**: Temel olarak nest() işlemini tersine çevirir. Listenin her elemanını kendi satırı yapar. Atomik vektörler, listeler veya data fream içeren (ancak farklı türlerin bir karışımını içermeyen) liste sütunlarını işleyebilir.

-> Karakter sütunlarını bölme ve birleştirme,birden çok sütunu tek bir karakter sütununda birleştirmek için ve tek bir karakter sütununu birden çok sütuna çekmek için separate(),extract() ve unite() fonksiyonlarını kullanır. Detaylı bilgi vermek gerekirse;

**separate() fonksiyonu**: Daha uzun verileri daha geniş bir biçime dönüştürür yani; Bir normal ifade veya bir karakterin konumlarının vektörü verildiğinde, separate(); tek bir karakter sütununu birden çok sütuna dönüştürür.

**extract() fonksiyonu**: Grupları yakalayan normal bir ifade verildiğinde, extract(), her grubu yeni bir sütuna dönüştürür. Gruplar eşleşmezse veya giriş NA ise, çıkış NA olur.

**unite() fonksiyonu**: İki sütunu bir sütunda birleştirir. Birden fazla değişken değerlerini yapıştırmak için bir kolaylık sağlayan bir fonksiyonudur. Kısaca tek bir gözlemin iki değişkenini tek bir değişkende birleştirir.

-> Örtük eksik değerleri açık hale getirir, örtük açık eksik değerler yapar ; eksik değerleri sonraki/önceki değerle veya bilinen bir değeri ile değiştirir. Bunlar için complete(), drop\_na(), fill() ve replace\_na() fonkiyonlarını kullanır. Detaylı bilgi vermek gerekirse;

**complete() fonksiyonu**: Örtülü eksik değerleri açık eksik değerlere dönüştürür.

**drop\_na() fonksiyonu**: Eksik değerleri içeren satırları veriden çıkartır.

**fill() fonksiyonu**: seçili sütunlardaki eksik değerleri doldurur. Bu, değerlerin tekrarlanmadığı ve yalnızca değiştiklerindekaydedilir.kaydedilen çıktı ise ortak çıktı biçiminde kullanışlı olur.

**replace\_na() fonksiyonu**: Eksik değerlerin içine atama yapmak için kullanılır.

tidyr, dağınık verilerinizi düzenlemek için üç ana işlev sunar: gather(), separate()ve spread(). Bu fonksiyonlar ve yukarıda bahsedilen fonksiyonlar haricinde diğer fonksiyonlar ağağıda açıklanmıştır.

**pivot\_longer() fonskiyonu**: pivot\_longer(), satır sayısını artırarak ve sütun sayısını azaltarak verileri uzatır.

**pivot\_wider() fonksiyonu**: pivot\_wider(), satır sayısını azaltarak ve sütun sayısını artırarak verileri kısaltır.

**starts\_with()fonksiyonu**: “Bu seçimin yardımcıları, değişkenleri belirli bir desene göre eşleştirir. Bu yardımcılar ise adlarındaki kalıpları eşleştirerek değişkenleri seçer:

1-starts\_with(): Baş kısmına eklerin .

2-ends\_with(): Sonuna ekler.

3-contains(): Gerçek bir dize içerir.

4-matches(): Normal bir ifadeyle eşleşir.

5-num\_range(): x01, x02, x03 gibi sayısal bir aralıkla eşleşir.

**crossing()fonksiyonu**: Girdilerini tekilleştiren ve sıralayan expand\_grid() çevresinde bir sarmalayıcıdır.

**all\_of()fonksiyonu**: Bir karakter vektöründeki değişken adlarını eşleştirir. Tüm adlar mevcut olmalıdır, aksi takdirde bir sınır dışı hatası verilir.

**any\_of()fonksiyonu**: all\_of() İle aynı, ancak var olmayan adlar için hiçbir hata atılmaz.

**full\_seq() fonksiyonu**: Temel olarak bir vektörde gözlemlenmesi gereken ancak gözlemlenmeyen eksik değerleri doldurur. Önemli bir nokta Vektör sayısal olmalıdır.

Ek olarak tidyr’de düzenli değerlendirmenin iki temel biçimi kullanılır:

**Tıdy Selection:**

Düzenli seçimi kullanan tüm işlevlerin altında tidyselect paketi bulunur. bu işlevi; fill(), drop\_na(), pivot\_longer(), pivot\_wider(), nest(), separate(), extract(), starts\_with() fon

**Data Masking:**

Veri değişkenlerini ortamdaki değişkenlermiş gibi kullanmamıza izin verir. Bu işlevi; as\_tibble(), nesting(), crossing(), expand() fonksiyonlarını kullanarak gerçekleştirir.

Yukarıda verdiğimiz bilgileri kısaca özetlemek gerekirse, tidyr verileri düzenlemeye, temizlememizi ve manipüle etmemizi kolaylaştıran ve buna olanak sağlayan bir kütüphanedir.

Yukarıda da bahsettiğmiz gibi tidyr, dağınık verilerinizi düzenlemek için üç ana işlev sunar: gather(), separate()ve spread() bu fonksiyonları verilerimizle örneklemek gerekirse;

**gather() fonsiyonu örneği:**

library(tidyr)

## Warning: package 'tidyr' was built under R version 4.1.2

library(readr)  
corona <- read\_csv("C:/Users/lenovo/Documents/GitHub/R-Final/Datasets/coronavirus.csv")

## Rows: 149028 Columns: 67

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (4): iso\_code, continent, location, tests\_units  
## dbl (62): total\_cases, new\_cases, new\_cases\_smoothed, total\_deaths, new\_dea...  
## date (1): date

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

head(corona, 3) #kitlemiz

## # A tibble: 3 x 67  
## iso\_code continent location date total\_cases new\_cases new\_cases\_smoot~  
## <chr> <chr> <chr> <date> <dbl> <dbl> <dbl>  
## 1 AFG Asia Afghanis~ 2020-02-24 5 5 NA  
## 2 AFG Asia Afghanis~ 2020-02-25 5 0 NA  
## 3 AFG Asia Afghanis~ 2020-02-26 5 0 NA  
## # ... with 60 more variables: total\_deaths <dbl>, new\_deaths <dbl>,  
## # new\_deaths\_smoothed <dbl>, total\_cases\_per\_million <dbl>,  
## # new\_cases\_per\_million <dbl>, new\_cases\_smoothed\_per\_million <dbl>,  
## # total\_deaths\_per\_million <dbl>, new\_deaths\_per\_million <dbl>,  
## # new\_deaths\_smoothed\_per\_million <dbl>, reproduction\_rate <dbl>,  
## # icu\_patients <dbl>, icu\_patients\_per\_million <dbl>, hosp\_patients <dbl>,  
## # hosp\_patients\_per\_million <dbl>, weekly\_icu\_admissions <dbl>, ...

set.seed(10) #kodu her çalıştırdığımızda örneklemimizin değişmemesi için kullandığımız bir fonksiyon  
data\_sample <- corona[sample(nrow(corona),10),replace=F]  
class(data\_sample)

## [1] "tbl\_df" "tbl" "data.frame"

sutun <- c("x", "y")  
new<-data.frame(sutun,data\_sample$new\_deaths,data\_sample$total\_cases\_per\_million,data\_sample$total\_cases)  
new\_new<- gather(new, referans, yeni\_deger, -sutun)  
print(new\_new)

## sutun referans yeni\_deger  
## 1 x data\_sample.new\_deaths 18.000  
## 2 y data\_sample.new\_deaths 0.000  
## 3 x data\_sample.new\_deaths 0.000  
## 4 y data\_sample.new\_deaths NA  
## 5 x data\_sample.new\_deaths 0.000  
## 6 y data\_sample.new\_deaths 15271.000  
## 7 x data\_sample.new\_deaths NA  
## 8 y data\_sample.new\_deaths 13.000  
## 9 x data\_sample.new\_deaths 2.000  
## 10 y data\_sample.new\_deaths NA  
## 11 x data\_sample.total\_cases\_per\_million 81.673  
## 12 y data\_sample.total\_cases\_per\_million 5.456  
## 13 x data\_sample.total\_cases\_per\_million 1534.298  
## 14 y data\_sample.total\_cases\_per\_million 8.602  
## 15 x data\_sample.total\_cases\_per\_million 458.253  
## 16 y data\_sample.total\_cases\_per\_million 11329.646  
## 17 x data\_sample.total\_cases\_per\_million 8.602  
## 18 y data\_sample.total\_cases\_per\_million 7832.209  
## 19 x data\_sample.total\_cases\_per\_million 1293.659  
## 20 y data\_sample.total\_cases\_per\_million 5544.966  
## 21 x data\_sample.total\_cases 11917.000  
## 22 y data\_sample.total\_cases 299.000  
## 23 x data\_sample.total\_cases 43616.000  
## 24 y data\_sample.total\_cases 1.000  
## 25 x data\_sample.total\_cases 6084.000  
## 26 y data\_sample.total\_cases 89220573.000  
## 27 x data\_sample.total\_cases 1.000  
## 28 y data\_sample.total\_cases 23247.000  
## 29 x data\_sample.total\_cases 14931.000  
## 30 y data\_sample.total\_cases 32.000

Bu kod bloğunda görüldüğü gibi kümme verilerimiz çok büyük olduğu için **set.seed** foknsiyonu ile her sütün için 10 gözleme indirerek devam ettik, verimizi data fream formatına dönüştürdük ve gather() fonksiyonu ile 3 sütunu birleştirerek tek bir sütun haline getirdik.

**spread() fonksiyonu örneği:**

library(tidyr)  
class(data\_sample)

## [1] "tbl\_df" "tbl" "data.frame"

sutun <- c("x", "y")  
yeni<-data.frame(sutun,data\_sample$location,data\_sample$total\_cases\_per\_million,data\_sample$date)  
yeni\_yeni <- separate(yeni, data\_sample.date, c("yil", "ay","gün"))  
print(yeni\_yeni)

## sutun data\_sample.location data\_sample.total\_cases\_per\_million yil ay  
## 1 x Russia 81.673 2020 04  
## 2 y Myanmar 5.456 2020 06  
## 3 x Madagascar 1534.298 2021 10  
## 4 y Micronesia (country) 8.602 2021 08  
## 5 x Rwanda 458.253 2020 12  
## 6 y World 11329.646 2021 01  
## 7 x Micronesia (country) 8.602 2021 09  
## 8 y Armenia 7832.209 2020 06  
## 9 x Haiti 1293.659 2021 06  
## 10 y Saint Pierre and Miquelon 5544.966 2021 10  
## gün  
## 1 10  
## 2 28  
## 3 15  
## 4 18  
## 5 05  
## 6 08  
## 7 09  
## 8 26  
## 9 03  
## 10 13

Bu kod bloğunda görüldüğü gibi tekrar “yeni” adında bir veri seti oluşturduk çünkü diğer örneklerde oluşturduğumuz veri setleri bu örnek için kullanılamaz. Bu örnekte separate() fonksiyonunu kullanarak yeni adlı veri setimizdeki date sütununu yıl, ay ve gün sütunu şeklinde 3 sütuna ayırdık.

**Verimizi Manipüle Etme ve Düzenleme Aşaması:**

library(readr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
corona <- read\_csv("C:/Users/lenovo/Documents/GitHub/R-Final/Datasets/coronavirus.csv")

## Rows: 149028 Columns: 67

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (4): iso\_code, continent, location, tests\_units  
## dbl (62): total\_cases, new\_cases, new\_cases\_smoothed, total\_deaths, new\_dea...  
## date (1): date

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

head(corona)

## # A tibble: 6 x 67  
## iso\_code continent location date total\_cases new\_cases new\_cases\_smoot~  
## <chr> <chr> <chr> <date> <dbl> <dbl> <dbl>  
## 1 AFG Asia Afghanis~ 2020-02-24 5 5 NA   
## 2 AFG Asia Afghanis~ 2020-02-25 5 0 NA   
## 3 AFG Asia Afghanis~ 2020-02-26 5 0 NA   
## 4 AFG Asia Afghanis~ 2020-02-27 5 0 NA   
## 5 AFG Asia Afghanis~ 2020-02-28 5 0 NA   
## 6 AFG Asia Afghanis~ 2020-02-29 5 0 0.714  
## # ... with 60 more variables: total\_deaths <dbl>, new\_deaths <dbl>,  
## # new\_deaths\_smoothed <dbl>, total\_cases\_per\_million <dbl>,  
## # new\_cases\_per\_million <dbl>, new\_cases\_smoothed\_per\_million <dbl>,  
## # total\_deaths\_per\_million <dbl>, new\_deaths\_per\_million <dbl>,  
## # new\_deaths\_smoothed\_per\_million <dbl>, reproduction\_rate <dbl>,  
## # icu\_patients <dbl>, icu\_patients\_per\_million <dbl>, hosp\_patients <dbl>,  
## # hosp\_patients\_per\_million <dbl>, weekly\_icu\_admissions <dbl>, ...

*Buradaki kod satırında;; “corona” adlı veri setimizi R’a aktarma işlemini yaptık, kullnacağımız kütüphanelerin girişlerini yaptık.*

library(readr)  
library(dplyr)  
library(tidyr)  
a <-corona %>%   
 filter(continent %in% c("Africa","Asia","Europe","North America","South America")) %>%   
 summarize(date=date,continent=continent, location=location, new\_cases=new\_cases, total\_cases=total\_cases, new\_cases\_per\_million=new\_cases\_per\_million,  
 new\_deaths=new\_deaths, total\_deaths=total\_deaths, new\_deaths\_per\_million=new\_deaths\_per\_million, population=population,cases\_density\_population=total\_cases/population)

*Buradaki kod satırında;; Corona verisinin içine girmemize yardımcı olan fonksiyon (%>%) ile yeni bir data seti oluşturmak üzere Corona verisinde yapılacak işlemleri a data setinin içine aktardık. “filter” fonksiyonunu kullanarak Corona verisinin içindeki Africa, Asia, Europe, North America ve South America kıtalarını a datasına atanmasını sağladık daha sonra, Verimizde kullanmak istediğimiz sütunlaru summarize() fonksiyonunu kullanarak çektik.*

a <- a%>%  
 filter(date > "2020-06-01" & date <"2020-06-30") %>%   
 drop\_na()

*Buradaki kod satırında;; Veri setimizdeki tarih verilerini filter() fonksiyonunu kullanarak “2020-06-01” ile “2020-06-30” tarihi arasındaki veriler olarak filtreledik ve drop\_na() fonksiyonun kullanarak eksik ve na değerlerini içeren kod satırlarını verimizden çıkardık.*

library(dplyr)  
africa <- a %>%  
 filter(continent == "Africa")  
Asia <- a %>%  
 filter(continent == "Asia")  
Europe <- a %>%  
 filter(continent == "Europe")  
North\_America <- a %>%  
 filter(continent == "North America")  
South\_America <- a %>%  
 filter(continent == "South America")

*Buradaki kod satırında; “a” veri setine atadığımız kıtaları filter() fonksiyonunu kullanarak her kıta için ayrı olarak oluşturduğumuz veri setinin içine atayarak filtreleme yaptık.*

x\_asia <- Asia[sample(nrow(Asia),20),]  
x\_africa <- africa[sample(nrow(africa),20),]  
x\_europe <- Europe[sample(nrow(Europe),20),]  
x\_sth\_america <- South\_America[sample(nrow(South\_America),20),]  
x\_nrth\_america <- North\_America[sample(nrow(North\_America),20),]

*Buradaki kod satırında; Kıtalara ait veri setlerimizi oluşturduktan sonra anlaşılabilir hale getirmek için bu veri setlerinin satırlarını 20 satıra indirgeyerek veri düzenlemesi işlemine devam ettik.*

x\_corona <- rbind(x\_asia,x\_africa,x\_europe,x\_sth\_america,x\_nrth\_america)

*Buradaki kod satırında;Veri setlerini 20 satıra indirdikten sonra hepsini tek bir veri setinde toplamak için rbind() fonksiyonunu kullanarak “x\_corona” data sına atama işlemini gerçekleştirdik.*

y\_corona <- x\_corona[,-1]

*Buradaki kod satırında; 1. sütünda bulunana tarih verilerini veri setimizden çıkardık.*  
**Bu işlemler sonucunda, Gozlemi yapilacak veri “y\_corona” verisi olarak elde edilmiştir.**

## Veri Analizi

**Kullanılan kütüphaneler:**

library(caret)

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.5

## Loading required package: lattice

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(readr)

## Warning: package 'readr' was built under R version 4.0.5

library(DescTools)

## Warning: package 'DescTools' was built under R version 4.0.5

##   
## Attaching package: 'DescTools'

## The following objects are masked from 'package:caret':  
##   
## MAE, RMSE

library(corrplot)

## corrplot 0.92 loaded

library(stringr)

## Warning: package 'stringr' was built under R version 4.0.5

**caret (Classification And REgression Training):** Caret paketi tahmine yönelik algoritma geliştirmeye olanak sağlayan bir pakettir. Veriyi bölme, önişleme, değişken seçimi, yeniden örnekleme metodunu kullanarak model doğrulama, değişken önem tahmini gibi ihtiyaca yönelik bir dizi özelliği içinde barındırır.

**DescTools:** DescTools paketi, verileri analiz etmek üzere doğru bir şekilde tanımlanması üzere içinde açıklayıcı ve ihtiyaca yönelik istatistiksel bilgiler barındıran bir pakettir.

**corrplot:** corrplot paketi, verisetindeki değişkenler arasındaki ilişkileri ve ilişki düzeylerini matris aracılığıyla görsel bir arayüzde kullanıcıya sunulmasını amaçlayan bir keşif paketidir.

**VERİSETİ**

corx<-read.csv("C:/Users/User/Desktop/y\_xorona.csv")  
head(corx)

## X continent location new\_cases total\_cases new\_cases\_per\_million new\_deaths  
## 1 1 Asia India 10667 343091 7.655 380  
## 2 2 Asia Palestine 41 555 7.850 0  
## 3 3 Asia Armenia 517 10009 174.184 19  
## 4 4 Asia Jordan 7 1111 0.682 0  
## 5 5 Asia Iran 2369 182525 27.861 75  
## 6 6 Asia Hong Kong 3 1131 0.397 0  
## total\_deaths new\_deaths\_per\_million population cases\_density\_population  
## 1 9900 0.273 1393409033 0.0002462242  
## 2 3 0.000 5222756 0.0001062657  
## 3 158 6.401 2968128 0.0033721592  
## 4 9 0.000 10269022 0.0001081895  
## 5 8659 0.882 85028760 0.0021466266  
## 6 5 0.000 7552800 0.0001497458

table(corx$continent)

##   
## Africa Asia Europe North America South America   
## 20 20 20 20 20

**TABAKALAMA**

Asia<-corx[corx$continent=="Asia",]  
Africa<-corx[corx$continent=="Africa",]  
Europe<-corx[corx$continent=="Europe",]  
N\_America<-corx[corx$continent=="North America",]  
S\_America<-corx[corx$continent=="South America",]

**ÖZET İSTATİSTİKLER**

summary(Asia)

## X continent location new\_cases   
## Min. : 1.00 Length:20 Length:20 Min. : 0.0   
## 1st Qu.: 5.75 Class :character Class :character 1st Qu.: 7.0   
## Median :10.50 Mode :character Mode :character Median : 61.0   
## Mean :10.50 Mean : 1109.5   
## 3rd Qu.:15.25 3rd Qu.: 865.8   
## Max. :20.00 Max. :10667.0   
## total\_cases new\_cases\_per\_million new\_deaths total\_deaths   
## Min. : 141 Min. : 0.0000 Min. : 0.0 Min. : 3.0   
## 1st Qu.: 1126 1st Qu.: 0.4135 1st Qu.: 0.0 1st Qu.: 7.5   
## Median : 7850 Median : 6.9560 Median : 0.5 Median : 71.0   
## Mean : 59707 Mean : 31.7194 Mean : 35.6 Mean :2212.1   
## 3rd Qu.: 55644 3rd Qu.: 31.5390 3rd Qu.: 19.0 3rd Qu.:2006.5   
## Max. :343091 Max. :174.1840 Max. :380.0 Max. :9996.0   
## new\_deaths\_per\_million population cases\_density\_population  
## Min. :0.0000 Min. :4.415e+05 Min. :9.685e-06   
## 1st Qu.:0.0000 1st Qu.:5.223e+06 1st Qu.:1.077e-04   
## Median :0.0135 Median :1.427e+07 Median :2.828e-04   
## Mean :0.6555 Mean :1.065e+08 Mean :1.384e-03   
## 3rd Qu.:0.4105 3rd Qu.:8.503e+07 3rd Qu.:2.340e-03   
## Max. :6.4010 Max. :1.393e+09 Max. :5.892e-03

summary(Africa)

## X continent location new\_cases   
## Min. :21.00 Length:20 Length:20 Min. : 0.00   
## 1st Qu.:25.75 Class :character Class :character 1st Qu.: 3.00   
## Median :30.50 Mode :character Mode :character Median : 15.50   
## Mean :30.50 Mean : 118.70   
## 3rd Qu.:35.25 3rd Qu.: 58.75   
## Max. :40.00 Max. :1455.00   
## total\_cases new\_cases\_per\_million new\_deaths total\_deaths   
## Min. : 25.0 Min. : 0.0000 Min. : 0.00 Min. : 1.00   
## 1st Qu.: 335.8 1st Qu.: 0.1935 1st Qu.: 0.00 1st Qu.: 5.00   
## Median : 733.0 Median : 1.2745 Median : 0.00 Median : 10.50   
## Mean : 3575.2 Mean : 3.3841 Mean : 2.95 Mean : 80.75   
## 3rd Qu.: 1944.0 3rd Qu.: 2.6715 3rd Qu.: 1.00 3rd Qu.: 74.25   
## Max. :35812.0 Max. :24.2330 Max. :50.00 Max. :755.00   
## new\_deaths\_per\_million population cases\_density\_population  
## Min. :0.00000 Min. : 1002197 Min. :2.534e-06   
## 1st Qu.:0.00000 1st Qu.: 7845642 1st Qu.:3.139e-05   
## Median :0.00000 Median :15092171 Median :4.907e-05   
## Mean :0.06170 Mean :24975042 Mean :3.335e-04   
## 3rd Qu.:0.05075 3rd Qu.:29361261 3rd Qu.:9.587e-05   
## Max. :0.83300 Max. :92377986 Max. :4.625e-03

summary(Europe)

## X continent location new\_cases   
## Min. :41.00 Length:20 Length:20 Min. : 0.0   
## 1st Qu.:45.75 Class :character Class :character 1st Qu.: 4.5   
## Median :50.50 Mode :character Mode :character Median : 20.0   
## Mean :50.50 Mean : 201.0   
## 3rd Qu.:55.25 3rd Qu.: 324.8   
## Max. :60.00 Max. :1218.0   
## total\_cases new\_cases\_per\_million new\_deaths total\_deaths   
## Min. : 324 Min. : 0.000 Min. : 0.00 Min. : 9.0   
## 1st Qu.: 2133 1st Qu.: 1.808 1st Qu.: 0.00 1st Qu.: 95.0   
## Median : 9098 Median : 4.073 Median : 2.50 Median : 341.5   
## Mean : 32241 Mean : 22.591 Mean : 5.00 Mean : 2779.8   
## 3rd Qu.: 50843 3rd Qu.: 30.168 3rd Qu.: 5.25 3rd Qu.: 1593.2   
## Max. :248469 Max. :119.880 Max. :48.00 Max. :28341.0   
## new\_deaths\_per\_million population cases\_density\_population  
## Min. :0.0000 Min. : 77354 Min. :0.0002807   
## 1st Qu.:0.0000 1st Qu.: 3263459 1st Qu.:0.0007460   
## Median :0.2430 Median : 5504544 Median :0.0023306   
## Mean :0.6302 Mean : 8417878 Mean :0.0030975   
## 3rd Qu.:0.6012 3rd Qu.: 9765661 3rd Qu.:0.0052619   
## Max. :4.7240 Max. :46745211 Max. :0.0110272

summary(N\_America)

## X continent location new\_cases   
## Min. : 81.00 Length:20 Length:20 Min. : 0.0   
## 1st Qu.: 85.75 Class :character Class :character 1st Qu.: 0.0   
## Median : 90.50 Mode :character Mode :character Median : 6.0   
## Mean : 90.50 Mean : 2251.2   
## 3rd Qu.: 95.25 3rd Qu.: 435.5   
## Max. :100.00 Max. :21723.0   
## total\_cases new\_cases\_per\_million new\_deaths total\_deaths   
## Min. : 22.0 Min. : 0.00 Min. : 0.0 Min. : 1.00   
## 1st Qu.: 117.8 1st Qu.: 0.00 1st Qu.: 0.0 1st Qu.: 7.75   
## Median : 2242.5 Median : 10.12 Median : 0.0 Median : 58.00   
## Mean : 207962.6 Mean : 27.89 Mean : 115.5 Mean : 12694.45   
## 3rd Qu.: 15718.8 3rd Qu.: 41.32 3rd Qu.: 17.5 3rd Qu.: 461.50   
## Max. :1960101.0 Max. :171.86 Max. :1006.0 Max. :113926.00   
## new\_deaths\_per\_million population cases\_density\_population  
## Min. :0.0000 Min. : 62092 Min. :8.765e-05   
## 1st Qu.:0.0000 1st Qu.: 256980 1st Qu.:2.705e-04   
## Median :0.0000 Median : 6610440 Median :6.428e-04   
## Mean :0.8215 Mean : 45611224 Mean :1.660e-03   
## 3rd Qu.:0.7823 3rd Qu.: 13218729 3rd Qu.:2.422e-03   
## Max. :5.6040 Max. :332915074 Max. :6.997e-03

summary(S\_America)

## X continent location new\_cases   
## Min. :61.00 Length:20 Length:20 Min. : 0.00   
## 1st Qu.:65.75 Class :character Class :character 1st Qu.: 51.25   
## Median :70.50 Mode :character Mode :character Median : 976.00   
## Mean :70.50 Mean : 3537.90   
## 3rd Qu.:75.25 3rd Qu.: 3890.25   
## Max. :80.00 Max. :39436.00   
## total\_cases new\_cases\_per\_million new\_deaths total\_deaths   
## Min. : 122 Min. : 0.00 Min. : 0.00 Min. : 1.0   
## 1st Qu.: 1786 1st Qu.: 15.41 1st Qu.: 0.00 1st Qu.: 21.5   
## Median : 39855 Median : 39.22 Median : 18.00 Median : 1454.5   
## Mean : 130214 Mean : 72.95 Mean : 145.80 Mean : 6869.6   
## 3rd Qu.: 198501 3rd Qu.: 93.36 3rd Qu.: 79.25 3rd Qu.: 3974.0   
## Max. :1145906 Max. :260.93 Max. :1374.00 Max. :52645.0   
## new\_deaths\_per\_million population cases\_density\_population  
## Min. : 0.0000 Min. : 591798 Min. :7.271e-05   
## 1st Qu.: 0.0000 1st Qu.: 9745990 1st Qu.:2.673e-04   
## Median : 0.8975 Median : 19212362 Median :1.074e-03   
## Mean : 2.9503 Mean : 32461021 Mean :3.339e-03   
## 3rd Qu.: 2.6410 3rd Qu.: 36421017 3rd Qu.:5.453e-03   
## Max. :17.5360 Max. :213993441 Max. :1.305e-02

**summary fonksiyonu:** Değişken hakkında özet istatistikler sunar.

Dünyadaki koronavirüs tablosunu daha iyi anlamak ve özetlemek için verilere kıta bazında bakmak gerekir. Asya’da her gün ortalama 1109 vaka, 35 ölüm gerçekleşmektedir. Afrika’da günlük ortalama 118 vaka, 2 ölüm; Avrupa’da günlük ortalama 201 vaka, 5 ölüm; Kuzey Ameriak’da günlük ortalama 2251 vaka, 115 ölüm; Güney Amerika’da 3537 vaka, 145 ölüm gerçekleşmektedir.

Dünyadaki günlük oranlarına farklı bir perspektiften bakmak için günlük vakaların nüfusa olan oranına bakıp pandemi sürecinin gidişatı hakkında daha sağlıklı yorum yapılmalıdır.

|  |  |
| --- | --- |
| Kıtalar | Vaka Oranı |
| Asya | 0.001384 |
| Afrika | 0.0003335 |
| Avrupa | 0.0030975 |
| Kuzey Amerika | 0.001660 |
| Güney Amerika | 0.003339 |

Kıtalara göre günlük vakaların nüfusa oranına bakıldığında en yüksek oranın Güney Amerika’da, en düşük oranın ise Afrika’da olduğu görülmüştür. Vaka yoğunluğu olarak Güney Amerika ve ona yakın bir değerde seyreden Avrupa’da vakaların oldukça fazla olduğu görülmüştür.

class(Asia)

## [1] "data.frame"

class(Africa)

## [1] "data.frame"

class(Europe)

## [1] "data.frame"

class(N\_America)

## [1] "data.frame"

class(S\_America)

## [1] "data.frame"

**class fonksiyonu:** Verinin türünü gösteren fonksiyondur.

dim(Asia)

## [1] 20 11

dim(Africa)

## [1] 20 11

dim(Europe)

## [1] 20 11

dim(N\_America)

## [1] 20 11

dim(S\_America)

## [1] 20 11

dim(corx)

## [1] 100 11

glimpse(Asia)

## Rows: 20  
## Columns: 11  
## $ X <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14~  
## $ continent <chr> "Asia", "Asia", "Asia", "Asia", "Asia", "Asia~  
## $ location <chr> "India", "Palestine", "Armenia", "Jordan", "I~  
## $ new\_cases <int> 10667, 41, 517, 7, 2369, 3, 26, 1396, 244, 31~  
## $ total\_cases <int> 343091, 555, 10009, 1111, 182525, 1131, 2120,~  
## $ new\_cases\_per\_million <dbl> 7.655, 7.850, 174.184, 0.682, 27.861, 0.397, ~  
## $ new\_deaths <int> 380, 0, 19, 0, 75, 0, 0, 19, 3, 70, 133, 1, 0~  
## $ total\_deaths <int> 9900, 3, 158, 9, 8659, 5, 8, 5065, 987, 8012,~  
## $ new\_deaths\_per\_million <dbl> 0.273, 0.000, 6.401, 0.000, 0.882, 0.000, 0.0~  
## $ population <int> 1393409033, 5222756, 2968128, 10269022, 85028~  
## $ cases\_density\_population <dbl> 2.462242e-04, 1.062657e-04, 3.372159e-03, 1.0~

glimpse(Africa)

## Rows: 20  
## Columns: 11  
## $ X <int> 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 3~  
## $ continent <chr> "Africa", "Africa", "Africa", "Africa", "Afri~  
## $ location <chr> "Madagascar", "Libya", "South Africa", "Mozam~  
## $ new\_cases <int> 18, 14, 1455, 28, 3, 35, 282, 11, 198, 3, 0, ~  
## $ total\_cases <int> 1290, 196, 35812, 816, 206, 1386, 11892, 431,~  
## $ new\_cases\_per\_million <dbl> 0.633, 2.012, 24.233, 0.871, 0.199, 1.678, 10~  
## $ new\_deaths <int> 0, 0, 50, 0, 0, 1, 2, 0, 0, 1, 0, 3, 0, 1, 0,~  
## $ total\_deaths <int> 10, 5, 755, 5, 4, 79, 303, 2, 142, 74, 4, 75,~  
## $ new\_deaths\_per\_million <dbl> 0.000, 0.000, 0.833, 0.000, 0.000, 0.048, 0.0~  
## $ population <int> 28427333, 6958538, 60041996, 32163045, 150921~  
## $ cases\_density\_population <dbl> 4.537886e-05, 2.816684e-05, 5.964492e-04, 2.5~

glimpse(Europe)

## Rows: 20  
## Columns: 11  
## $ X <int> 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 5~  
## $ continent <chr> "Europe", "Europe", "Europe", "Europe", "Euro~  
## $ location <chr> "Bosnia and Herzegovina", "Slovakia", "Andorr~  
## $ new\_cases <int> 11, 2, 1, 17, 3, 99, 5, 732, 1218, 564, 464, ~  
## $ total\_cases <int> 2535, 1533, 853, 7104, 2258, 61106, 25396, 53~  
## $ new\_cases\_per\_million <dbl> 3.371, 0.366, 12.928, 3.064, 0.735, 8.511, 1.~  
## $ new\_deaths <int> 3, 0, 0, 1, 0, 5, 6, 5, 48, 3, 6, 0, 6, 10, 0~  
## $ total\_deaths <int> 157, 28, 51, 326, 107, 9731, 1726, 308, 4939,~  
## $ new\_deaths\_per\_million <dbl> 0.919, 0.000, 0.000, 0.180, 0.000, 0.430, 1.2~  
## $ population <int> 3263459, 5460726, 77354, 5548361, 4081657, 11~  
## $ cases\_density\_population <dbl> 0.0007767832, 0.0002807319, 0.0110272255, 0.0~

glimpse(N\_America)

## Rows: 20  
## Columns: 11  
## $ X <int> 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 9~  
## $ continent <chr> "North America", "North America", "North Amer~  
## $ location <chr> "Guatemala", "Haiti", "Cayman Islands", "Cuba~  
## $ new\_cases <int> 434, 148, 6, 3, 21723, 0, 0, 327, 76, 0, 0, 4~  
## $ total\_cases <int> 10706, 3072, 193, 2315, 1878885, 102, 101, 10~  
## $ new\_cases\_per\_million <dbl> 23.781, 12.823, 90.228, 0.265, 65.251, 0.000,~  
## $ new\_deaths <int> 19, 0, 0, 0, 1006, 0, 0, 19, 1, 0, 0, 7, 0, 0~  
## $ total\_deaths <int> 418, 50, 1, 85, 111443, 11, 3, 8489, 52, 64, ~  
## $ new\_deaths\_per\_million <dbl> 1.041, 0.000, 0.000, 0.000, 3.022, 0.000, 0.0~  
## $ population <int> 18249868, 11541683, 66498, 11317498, 33291507~  
## $ cases\_density\_population <dbl> 5.866344e-04, 2.661657e-04, 2.902343e-03, 2.0~

glimpse(S\_America)

## Rows: 20  
## Columns: 11  
## $ X <int> 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 7~  
## $ continent <chr> "South America", "South America", "South Amer~  
## $ location <chr> "Chile", "Ecuador", "Suriname", "Peru", "Colo~  
## $ new\_cases <int> 3804, 1003, 22, 4358, 4149, 2115, 15, 578, 13~  
## $ total\_cases <int> 250767, 51643, 122, 191758, 88591, 57046, 230~  
## $ new\_cases\_per\_million <dbl> 197.998, 56.070, 37.175, 130.638, 80.931, 41.~  
## $ new\_deaths <int> 3, 51, 0, 585, 128, 63, 0, 46, 0, 0, 36, 14, ~  
## $ total\_deaths <int> 4505, 4274, 1, 24225, 2939, 1864, 12, 3874, 2~  
## $ new\_deaths\_per\_million <dbl> 0.156, 2.851, 0.000, 17.536, 2.497, 1.229, 0.~  
## $ population <int> 19212362, 17888474, 591798, 33359415, 5126584~  
## $ cases\_density\_population <dbl> 1.305238e-02, 2.886943e-03, 2.061514e-04, 5.7~

**glimpse fonksiyonu:** Girilen verisetinin bir çerçeve halinde görülmesini sağlar.

**KUKLA DEĞER BELİRLEME**

ratios<-data.frame(continents=c("Asia", "Africa", "Europe", "South America", "North America"),ratio=c(0.001384, 0.0003335, 0.0030975, 0.003339, 0.001660), criticality=c(1,0,1,1,1))  
dummies <- dummyVars(~., data = ratios)  
head(predict(dummies, newdata = ratios))

## continentsAfrica continentsAsia continentsEurope continentsNorth America  
## 1 0 1 0 0  
## 2 1 0 0 0  
## 3 0 0 1 0  
## 4 0 0 0 0  
## 5 0 0 0 1  
## continentsSouth America ratio criticality  
## 1 0 0.0013840 1  
## 2 0 0.0003335 0  
## 3 0 0.0030975 1  
## 4 1 0.0033390 1  
## 5 0 0.0016600 1

**dummyVars fonksiyonu:**

Veriseti içerisindeki kukla değişkenleri veren fonksiyondur.

**Stokastik yapıyı bozan değişkenler (sıfır ve sıfıra yakın varyanslılar):**

data.frame(table(corx$new\_deaths))

## Var1 Freq  
## 1 0 48  
## 2 1 9  
## 3 2 2  
## 4 3 6  
## 5 4 1  
## 6 5 2  
## 7 6 3  
## 8 7 1  
## 9 8 1  
## 10 10 1  
## 11 14 1  
## 12 15 1  
## 13 17 1  
## 14 19 4  
## 15 21 1  
## 16 36 1  
## 17 46 1  
## 18 48 1  
## 19 50 1  
## 20 51 1  
## 21 63 1  
## 22 70 1  
## 23 75 1  
## 24 128 1  
## 25 133 1  
## 26 226 1  
## 27 354 1  
## 28 380 1  
## 29 511 1  
## 30 585 1  
## 31 730 1  
## 32 1006 1  
## 33 1374 1

Veriler analiz edilmeden önce modelin kararsızlaşmasına neden olan ve stokastik süreci bozan değişkenlerin tespit edilmesi gerekir. Bu yapıyı bozan değişkenler sıfır ve sıfıra yakın varyanslılardır. Yapılan analiz sonucunda 48 değişkenin sıfır varyanslı olduğu gözlenmektedir. Ayrıca sıfıra yaklaşan değerlerin de sıklığı tabloda görülmektedir. Sıfır ve sıfıra yakın varyanslı değerlerin sıklığı önceden belirlenmiş bir eşik değeri aştığında verisetinden çıkarılarak analize devam edilir. Aksi halde modelin kararsızlaşmasına yol açar ve overfitting’e varan durumlara yol açarak sağlıklı tahminler yapılmasının önünüe geçer.

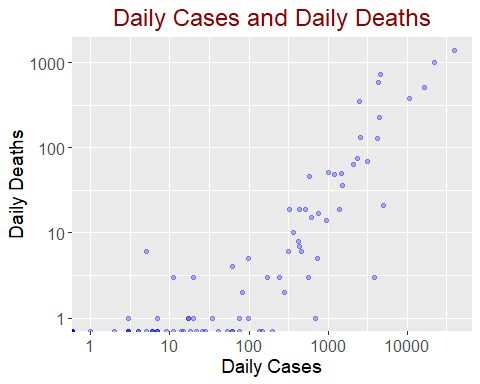
**LİNEER REGRESYON MODELİ**

**Regresyon Grafiği:**

corx%>% ggplot(aes(new\_cases, new\_deaths)) +   
 geom\_point(color="blue", alpha=0.3) +  
 ggtitle("Daily Cases and Daily Deaths") +  
 xlab("Daily Cases") +  
 ylab("Daily Deaths") +   
 theme(plot.title = element\_text(color = "darkred",  
 size = 18,  
 hjust = 0.5),  
 axis.text.y = element\_text(size=12),  
 axis.text.x = element\_text(size = 12, hjust = .5),  
 axis.title.x = element\_text(size = 14),  
 axis.title.y = element\_text(size = 14))+ scale\_x\_log10()+scale\_y\_log10()

## Warning: Transformation introduced infinite values in continuous x-axis

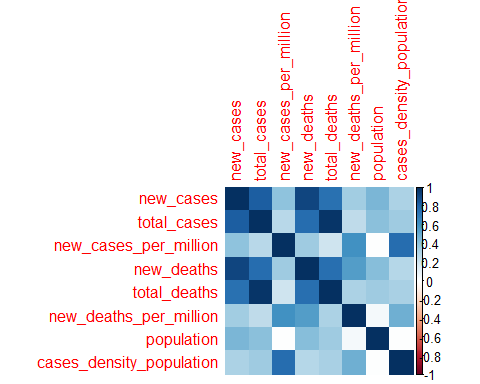
## Warning: Transformation introduced infinite values in continuous y-axis



5 kıtada gözlenen günlük vaka sayıları ile günlük ölüm sayıları arasındaki ilişkiyi gözlemlemek üzere ggplot fonksiyonu ile regresyon grafiği çizdirildi. Günlük vaka sayıları ile günlük ölüm sayıları arasında doğrusal bir ilişki olduğu görülmektedir. İlişkinin doğrusal ve yüksek ilişkili olduğunun saptanmasının ardından lineer regrsyon modeli kurulabileceğine karar verilmiştir. X ve Y eksenlerinin standartlaştırılması ve bu sayede ilişkinin grafik üzerinde daha görünür olması için scale\_x\_log10() scale\_y\_log10() fonksiyonları kullanılmıştır.

**Korelasyon Matrisi Isı Haritası:**

correlations = cor(corx[, 4:11])  
corrplot(correlations, method="color")



Analiz yapılacak veriler arasındaki daoğrusal ilişkiyi ve gücünü görmek amacıyla korelasyon hesaplanır. cor() fonksiyonuyla fonksiyonun içine girilen değerlerin matris olarak korelasyonları hesaplanır. corrplot fonksiyonu ile matris olarak hesaplanan korelasyon tablosu görselleştirilir. Güçlü pozitif ilişkiye sahip olanlar koyu mavi ile, ilişkisiz olanlar beyaz ile, güçlü negatif ilişkiye sahip olanlar ise koyu kırmızı ile gösterilir. Aradaki korelasyon değerler buna bağlı olarak renk skalasında karşılık gelen renk ile tanımlanır.

Korelasyon matrisine bakıldığında total\_cases (toplam vaka sayısı) değişkeni ile total\_deaths (toplam ölüm sayısı) değişkeni arasında ilişki gücü koyu mavi olarak gösterildiğinden dolayı güçlü olarak tanımlanır. Aynı yorumu kullanarak total\_cases ile new\_cases, total\_deaths ile new\_deaths değişkenleri arasında doğrusal ilişki olduğu söylenebilir.

**Verisetini bölme işlemi (Train set-Test data)**

set.seed(81)  
inTrain<-createDataPartition(y = corx$new\_deaths, p = 0.8, list = FALSE)  
training<-corx[inTrain,]  
testing<-corx[-inTrain, ]  
  
rbind("Training Set" = nrow(training)/nrow(corx),  
 "Test Data" = nrow(testing)/nrow(corx))%>%  
 round(2) #train=0.81, test=0.18

## [,1]  
## Training Set 0.81  
## Test Data 0.19

Lineer regresyon modelini kurmadan önce modelin eğitilmesi ve modelin doğrulanması için veriseti iki parçaya bölünmelidir. Yukarıdaki kod bloğunda createDataPartition() fonksiyonuyla tahmin yapılmak istenen değişkenin p parametresine girilen değer oranında bölünmesi sağlanır. Bu bölünen değerler modelimizin eğitim kısmında yer alacak olan değerlerdir. Her ne kadar verisetini p parametresine girilen değer kadar bölünmesi beklense de fonksiyon içinde bu değere yakın olacak şekilde optimize edilmiş biçimde bir bölünme sağlanır. Yukarıda da görüldüğü üzere Traning Set (eğitim seti) 0.81 oranında, Test Data (Test Verisi) 0.19 oranında bölünmüştür.

**Modeli kurma:**

linear\_model <- train(training[,c(4,5,8)], training[,7],  
 method = "lm",  
 preProcess = c("center","scale"))

Burada train() fonksiyonuyla model kurma işlemi gerçekleşmektedir. Burada tahmin edilecek değişken ayrı, tahmin etmek için kullanılacak değişkenler ayrı olarak girilir. ‘method’ parametresine lineer regresyon modeliyle tahmin yapmak istediğimiz için “lm” yazarak metodumuzu belirleriz.

**RMSE ve R-Squared değerlerini elde etme:**

linear\_model$results[c("RMSE","Rsquared")] %>%  
 round(2)

## RMSE Rsquared  
## 1 87.94 0.75

summary(linear\_model)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -260.45 -6.85 -3.75 -2.31 369.52   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 61.580 7.126 8.641 5.89e-13 \*\*\*  
## new\_cases 222.069 16.119 13.777 < 2e-16 \*\*\*  
## total\_cases -242.655 41.422 -5.858 1.09e-07 \*\*\*  
## total\_deaths 240.245 33.137 7.250 2.78e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 54.73 on 77 degrees of freedom  
## Multiple R-squared: 0.8325, Adjusted R-squared: 0.826   
## F-statistic: 127.6 on 3 and 77 DF, p-value: < 2.2e-16

Lineer regresyon modelinin oluşturulmasının ardından RMSE (Root Mean Squared Error - Kök Ortalama Kare Hata) ve R-squared değerleri hesaplanır. RMSE, tahminleyicinin tahmin ettiği değerler ile gerçek değerleri arasındaki uzaklığın bulunmasında sıklıkla kullanılan, hatanın büyüklğünü ölçen kuadratik bir metriktir. RMSE tahmin hatalarının (kalıntıların) standart sapmasıdır. Yani, kalıntılar, regresyon hattının veri noktalarından ne kadar uzakta olduğunun bir ölçüsüdür. Verilere en iyi uyan çizgi etrafında o verilerin ne kadar yoğun olduğunu söyler. R-squared, bir değişkenin varyansının ikinci değişkenin varyansını ne ölçüde açıkladığını açıklar. R-squared oranı ne kadar büyükse regresyon modelinin uyumu o kadar iyidir.

Yukarıdaki analize bakılarak RMSE değerinin 87.94, Rsquared’in 0.75 olduğu görülmüştür. Bu değerlere bakarak oluşturulan modelin uyumunun yüksek olduğu ve verilerin en iyi uyum çizgisi etrafında toplandığını söyleyebiliriz.

Lineer regresyon modelimiz hakkında özetleyici istatistiklere baktığımızda medyan 3.923 olarak görülmektedir. Katsayılar tablosuna bakıldığında tahminler, her değişkenin standart hatası ve t-testi değerleri görülmektedir.Artık sandart hata oranı 77 serbestlik derecesiyle 54.73, çoklu R kare değeri 0.8325, ve düzeltilmiş R kare değeri 0.826 olarak bulunmuştur. Böylece model uyumluluğu yüksek olduğu kanısına varılır. Ayrıca 3;77 serbestlik derecesiyle F-istatistiği 127.6 olarak bulunmuştur.

**Tahmin oluşturma:**

prediction<-predict(linear\_model, testing[,c(4,5,8)])

predict() fonksiyonuyla oluşturulan regresyon modelindeki tahmin edilecek değerlerin tahmin edilmesi sağlanır.

**SSE VE SST oluşturma:**

SSE=sum((testing[,7] - prediction)^2)  
SST=sum((testing[,7] - mean(training[,7]))^2)

SST, gözlemlenen bağımlı değişken ile ortalaması arasındaki farkların karesidir. Gözlemlenen değişkenlerin ortalama etrafındaki dağılımı olarak düşünülebilir. Bu modelde SST değeri SST 2720593 olarak çıkmıştır.

SSE, hata kareler toplamıdır. Hata, gözlenen değer ile tahmin edilen değer arasındaki farktır. Genellikle hatayı en aza indirmek modelin daha doğru sonuç vermesini sağlar. Bu modelde SSE değeri 828219.2’dir.

**R-squared**

R\_square=1-SSE/SST  
round(R\_square, 2)

## [1] 0.7

Düzelitmiş R kare (adjusted R squared), model karmaşıklığını ortadan kaldırmak, modeli anlaşılabilir ve yorumlanabilir kılmak ve performansı artırmak amacıyla başvurulan bir ölçüttür. Modelde karışıklık yaratan ve hedef değişkene etkisi olmayan değişkenler bu ölçütün değerini düşürmektedir. Böylece daha verimli bir uyum metriği bulunmuş olur. Yapılan hesap sonucu düzeltilmiş R karenin değeri 0.7 olarak bulunmuştur. Bu değer yeterli bir regresyon modeli için göstergedir.

**RMSE:**

SSE = sum((testing[,7] - prediction)^2)  
RMSE = sqrt(SSE/length(prediction))  
  
round(RMSE, 2)

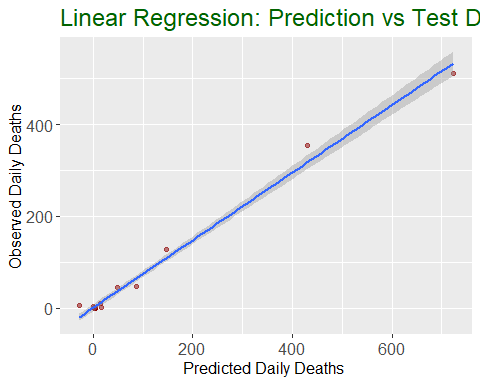
## [1] 208.78

Tahmin edilen değerler için oluşturulan RMSE değeri burada 208.78 olarak bulunmuştur.

**Regresyon modeli grafiği:**

my\_data = as.data.frame(cbind(predicted = prediction,  
 observed = testing$new\_deaths))  
  
ggplot(my\_data,aes(predicted, observed)) +  
 geom\_point(color = "darkred", alpha = 0.5) +   
 geom\_smooth(method=lm)+ ggtitle('Linear Regression ') +  
 ggtitle("Linear Regression: Prediction vs Test Data") +  
 xlab("Predicted Daily Deaths ") +  
 ylab("Observed Daily Deaths") +  
 theme(plot.title = element\_text(color="darkgreen",size=18),  
 axis.text.y = element\_text(size=12),  
   
 axis.text.x = element\_text(size=12,),  
 axis.title.x = element\_text(size=12),  
 axis.title.y = element\_text(size=12))

## `geom\_smooth()` using formula 'y ~ x'



Kurulan lineer regresyon modeli sonucu tahmin verilerinin test verileriyle doğrusal güçlü bir ilişkiye sahip olduğu görülmüştür. Böylece modelimiz başarılı olmuş ve günlük ölüm sayısını yüksek doğrulukta tahmin etmeye hazırdır.

**HATA MATRİSİ**

matrix<-table(factor(as.factor(nrow(as.data.frame(prediction))), levels=1:19),   
 factor(as.factor(nrow(testing)), levels=1:19))  
confusionMatrix(matrix)

##   
## Overall Statistics  
##   
## Accuracy : 1   
## 95% CI : (0.025, 1)  
## No Information Rate : 1   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : NaN   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6  
## Sensitivity NA NA NA NA NA NA  
## Specificity 1 1 1 1 1 1  
## Pos Pred Value NA NA NA NA NA NA  
## Neg Pred Value NA NA NA NA NA NA  
## Prevalence 0 0 0 0 0 0  
## Detection Rate 0 0 0 0 0 0  
## Detection Prevalence 0 0 0 0 0 0  
## Balanced Accuracy NA NA NA NA NA NA  
## Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12  
## Sensitivity NA NA NA NA NA NA  
## Specificity 1 1 1 1 1 1  
## Pos Pred Value NA NA NA NA NA NA  
## Neg Pred Value NA NA NA NA NA NA  
## Prevalence 0 0 0 0 0 0  
## Detection Rate 0 0 0 0 0 0  
## Detection Prevalence 0 0 0 0 0 0  
## Balanced Accuracy NA NA NA NA NA NA  
## Class: 13 Class: 14 Class: 15 Class: 16 Class: 17  
## Sensitivity NA NA NA NA NA  
## Specificity 1 1 1 1 1  
## Pos Pred Value NA NA NA NA NA  
## Neg Pred Value NA NA NA NA NA  
## Prevalence 0 0 0 0 0  
## Detection Rate 0 0 0 0 0  
## Detection Prevalence 0 0 0 0 0  
## Balanced Accuracy NA NA NA NA NA  
## Class: 18 Class: 19  
## Sensitivity NA 1  
## Specificity 1 NA  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0 1  
## Detection Rate 0 1  
## Detection Prevalence 0 1  
## Balanced Accuracy NA NA

**confusionMatrix() fonksiyonu:**

Bu fonksiyon, model oluşturulduktan sonra test ve tahmin verilerini matris halinde çapraz karşılaştırma yaparak model istatistiklerini çıkarır. Accuracy değeri modelin doğruluk oranını gösterir. Aynı zamanda %95 güven düzeyinde gücen aralıklarını verir ve p değerini hesaplar. McNemar Test istatistiği p-değerini hesaplayarak konum karşılaştırması yapar. Ayrıca verilerin sınıfsal hassasiyetlerini, özgüllüğünü ölçer.

## Veri Görselleştirme

library(esquisse)

## Warning: package 'esquisse' was built under R version 4.1.2

*Esquisse paketi, verilerimizi etkileşimli olarak keşfetmenize ve görselleştirmenize yardımcı olan bir pakettir. Değişkenlerimizi sürükle ve bırak ile etkileşimli olarak ggplot grafikleri oluşturmamızı sağlar. Görselimizi ‘PNG’ veya ’PowerPoint’e aktarabilir veya scriptimize eklemek için kodu içeriye aktarabiliriz.*

y\_corona$continent <- as.factor(y\_corona$continent)  
y\_corona$location <- as.factor(y\_corona$location)

*as.factor() kodunu kullanarak kıta ve ülke değişkenlerimizi faktör haline getirdik ki esquisse’de grafik çizdirirken kütüphanenin daha doğru tepki vermesini sağlayabilelim.*

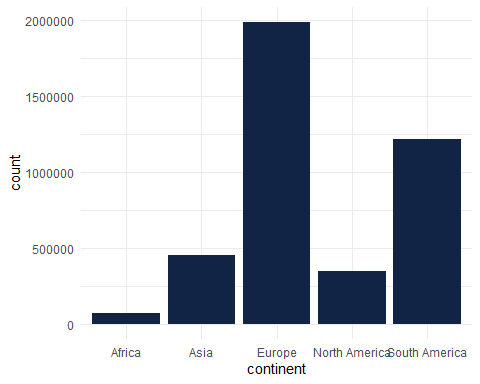
#esquisser(data=y\_corona)

*yorum satırına aldığımız esquisser() komutu, sürükle bırak şeklinde grafik çizdirebileceğimiz pencereryi açmamıza olanak sağlıyor.*

#### Bar Grafikleri

**TÜM KITALARA İLİŞKİN BAR GRAFİĞİ**

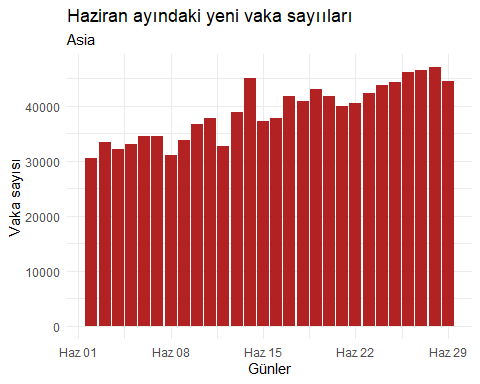
library(ggplot2)  
ggplot(y\_corona) +  
 aes(x = continent, weight = total\_cases) +  
 geom\_bar(fill = "#112446") +  
 theme\_minimal()



*Bu grafiğe bakarak, söylenebilir ki en fazla veri Avrupa’da gözlenmiş. Bunun yanında ise gözlenmiş en az değerler Afrika ya ait olmuş.*

**Asya’nın günlere göre yeni vaka sayısının bar grafiği:**

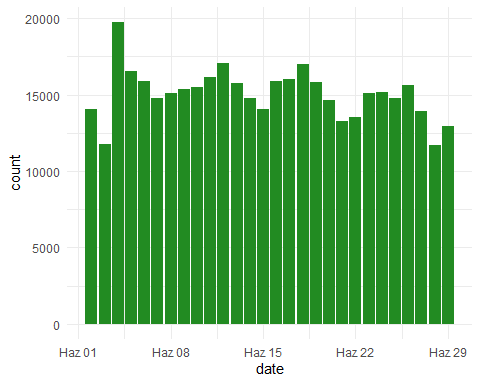
library(ggplot2)  
ggplot(Asia) +  
 aes(x = date, weight = new\_cases) +  
 geom\_bar(fill = "#B22222") +  
 labs(x = "Günler", y = "Vaka sayısı",   
 title = "Haziran ayındaki yeni vaka sayııları", subtitle = "Asia") +  
 theme\_minimal()



*Bu bar grafiğimizde ise gün geçtikçe yeni vaka sayısının arttığını da çok net bir şekilde söyleyebiliriz. Ayrıca Haziran ayına ait olan bu verilerde vaka sayısı günlük en az 30.000’i görmüştür.*

**Avrupa’nın günlere göre yeni vaka sayısının bar grafiği:**

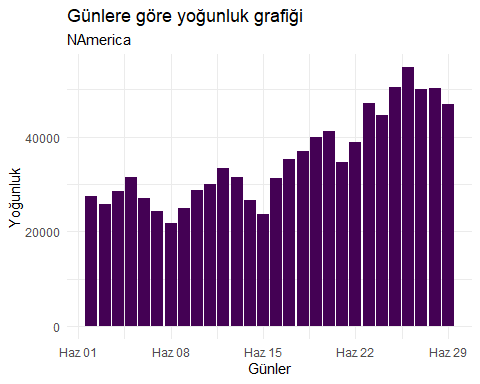
library(ggplot2)  
ggplot(Europe) +  
 aes(x = date, weight = new\_cases) +  
 geom\_bar(fill = "#228B22") +  
 theme\_minimal()



*Avrupaya ait olan grafikte Net bir şekilde yeni vaka sayıları gittikçe azalıyor diyemesek de söyleyebiliriz ki ayın başlarında görülen yüksek sayıdaki vaka sayıları haziran ayı sonlarına doğru biraz daha azalmıştır.*

**Kuzey Amerika’nın günlere göre yeni vaka sayısının bar grafiği:**

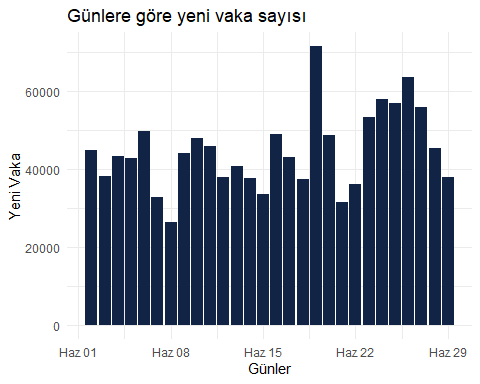
library(ggplot2)  
ggplot(North\_America) +  
 aes(x = date, weight = new\_cases) +  
 geom\_bar(fill = "#440154") +  
 labs(x = "Günler",   
 y = "Yoğunluk", title = "Günlere göre yoğunluk grafiği", subtitle = "NAmerica") +  
 theme\_minimal()



*Kuzey Amerika’ya ait bu grafikte çok net bir şekilde söyleyebiliriz ki gün geçtikçe vaka sayıları artmıştır. Demek ki rehavete kapılmamak gerekirmiş.*

**Güney Amerika’nın günlere göre yeni vaka sayısının bar grafiği:**

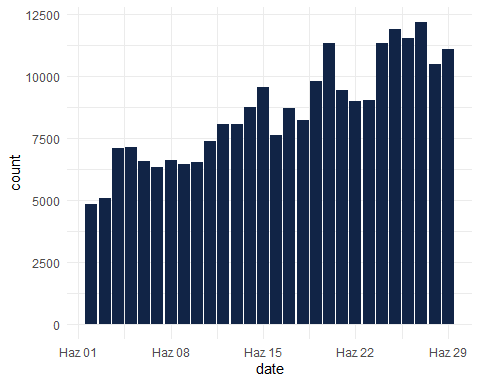
library(ggplot2)  
ggplot(South\_America) +  
 aes(x = date, weight = new\_cases) +  
 geom\_bar(position = "dodge", fill = "#112446") +  
 labs(x = "Günler", y = "Yeni Vaka", title = "Günlere göre yeni vaka sayısı") +  
 theme\_minimal()



*Güney Amerikaya ait veriler, haziran ayı başından ortalarına kadar dalgalanmalar olsa da yeni vaka sayısını max 50.000 civarında tutmayı başarabilmiş. haziranın 18 19 undan itibaren hızla yükselen vaka sayılarını yıl sonunda 40.000 civarına düşürmeyi başarabilmişler*

**Afrika’nın günlere göre yeni vaka sayısının bar grafiği:**

library(ggplot2)  
ggplot(africa) +  
 aes(x = date, weight = new\_cases) +  
 geom\_bar(fill = "#112446") +  
 theme\_minimal()

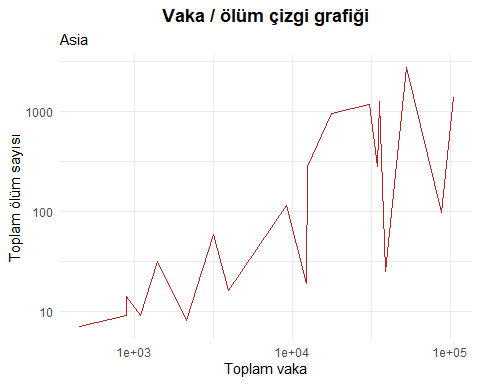


*Afrikaya ait grafiğe 1 belki 1.5 sn bakmamız yeterli olacaktır. Zira Afrika’da gün geçmemiş ki vaka sayısı artmasın.*

#### Çizgi Grafikleri

**Asya’nın toplam vaka / toplam ölüm çizgi grafiği:**

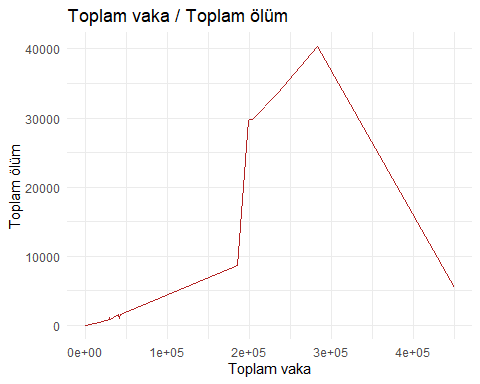
library(dplyr)  
library(ggplot2)  
y\_corona %>%  
 filter(continent %in% "Asia") %>%  
 ggplot() +  
 aes(x = total\_cases, y = total\_deaths) +  
 geom\_line(size = 0.5, colour = "#B22222") +  
 scale\_x\_continuous(trans = "log10") +  
 scale\_y\_continuous(trans = "log10") +  
 labs(x = "Toplam vaka",   
 y = "Toplam ölüm sayısı", title = "Vaka / ölüm çizgi grafiği", subtitle = "Asia") +  
 theme\_minimal() +  
 theme(plot.title = element\_text(face = "bold", hjust = 0.5))



*Eveet şimdi geldik Asyanın vaka sayısına göre olan ölüm sayısının çizgi grafiğine. Bu kısımda çeşitli iniş çıkışlar olsa da haticeye değil neticeye baktığımızda söyleyebiliriz ki toplam vaka sayısı arttıkça toplam ölüm sayısı da artmış, saygılar.*

**Avrupa’nın toplam vaka / toplam ölüm çizgi grafiği:**

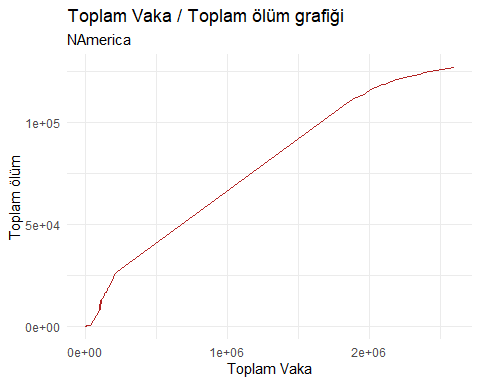
library(dplyr)  
library(ggplot2)  
y\_corona %>%  
 filter(continent %in% "Europe") %>%  
 ggplot() +  
 aes(x = total\_cases, y = total\_deaths) +  
 geom\_line(size = 0.5, colour = "#B22222") +  
 labs(x = "Toplam vaka", y = "Toplam ölüm", title = "Toplam vaka / Toplam ölüm ") +  
 theme\_minimal()



*Avrupa’nın grafiğine bakmak bizi bir hayli şaşırttı açıkcası. Pek çok konuda bizi şaşırtan Avrupa bu konuda da şaşırttı. Çizgi grafiğine yorum olarak söyleyebiliriz ki vaka sayısı 200000’e yaklaşırken belli bir ivmede seyreden ölüm sayısı 200000’ten 300000’e aşırı bir hızlanma ile vaka sayısı yaklaşık 280000 iken Avrupa haziran ayındaki en fazla ölüm sayısını görmüş. Bu saatten sonra dönüş yok derken Avrupalılar hızlıca toparlanmış dolayısıyla ölüm sayısı hızla düşmüş.*

**Kuzey Amerika’nın toplam vaka / toplam ölüm çizgi grafiği grafiği:**

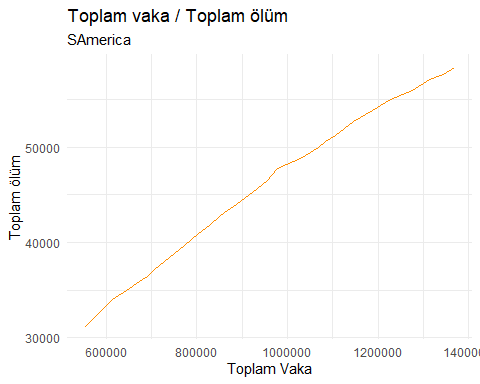
library(ggplot2)  
ggplot(North\_America) +  
 aes(x = total\_cases, y = total\_deaths) +  
 geom\_line(size = 0.5, colour = "#B22222") +  
 labs(x = "Toplam Vaka", y = "Toplam ölüm", title = "Toplam Vaka / Toplam ölüm grafiği", subtitle = "NAmerica") +  
 theme\_minimal()



*Kuzey Amerika’ya ait olan bu grafiğimizin yorumunda ise klasik olarak toplam vaka arttıkça toplam ölüm de vakaya bağlı olarak artmıştır.*

**Güney Amerika’nın toplam vaka / toplam ölüm çizgi grafiği grafiği:**

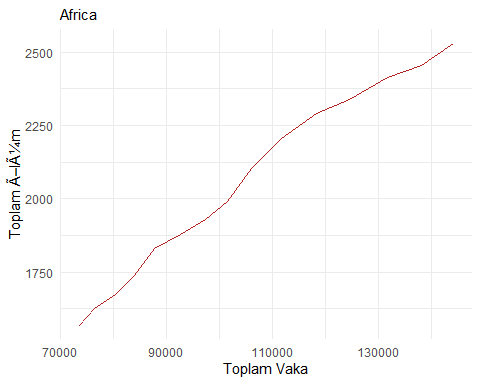
library(dplyr)  
library(ggplot2)  
South\_America %>%  
 filter(total\_cases >= 284703L & total\_cases <= 1368195L) %>%  
 ggplot() +  
 aes(x = total\_cases, y = total\_deaths) +  
 geom\_line(size = 0.5, colour = "#FF8C00") +  
 labs(x = "Toplam Vaka", y = "Toplam ölüm", title = "Toplam vaka / Toplam ölüm ", subtitle = "SAmerica") +  
 theme\_minimal()



*Güney Amerika’da Kuzey Amerika’dan çok farklı olmayarak vaka sayısının artışına bağlı olarak ölüm sayıları da artmış.*

**Afrika’nın toplam vaka / toplam ölüm çizgi grafiği grafiği:**

library(dplyr)  
library(ggplot2)  
africa %>%  
 filter(total\_cases >= 70217L & total\_cases <= 144264L) %>%  
 ggplot() +  
 aes(x = total\_cases, y = total\_deaths) +  
 geom\_line(size = 0.5, colour = "#B22222") +  
 labs(x = "Toplam Vaka", y = "Toplam Ã–lÃ¼m", subtitle = "Africa") +  
 theme\_minimal()



*Çoğu grafiğe yaptığımız yoruma Afrika’da dahil olarak sıraya kaynak yaptı. Afrika’nın da ölüm sayıları vaka artışına bağlı olarak artmıştır.*

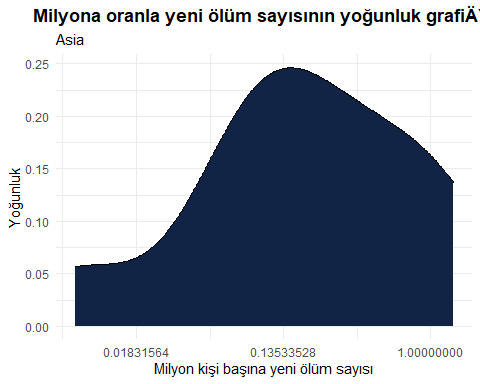
#### Yoğunluk Grafikleri:

**Asya’nın milyon kişi başına ölüm sayısına ait yoğunluk grafiği:**

library(dplyr)  
library(ggplot2)  
y\_corona %>%  
 filter(continent %in% "Asia") %>%  
 ggplot() +  
 aes(x = new\_deaths\_per\_million) +  
 geom\_density(adjust = 1L, fill = "#112446") +  
 scale\_x\_continuous(trans = "log") +  
 labs(x = "Milyon kişi başına yeni ölüm sayısı", y = "Yoğunluk", title = "Milyona oranla yeni ölüm sayısının yoğunluk grafiÄŸi",   
 subtitle = "Asia") +  
 theme\_minimal() +  
 theme(plot.title = element\_text(face = "bold", hjust = 0.5))

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 11 rows containing non-finite values (stat\_density).



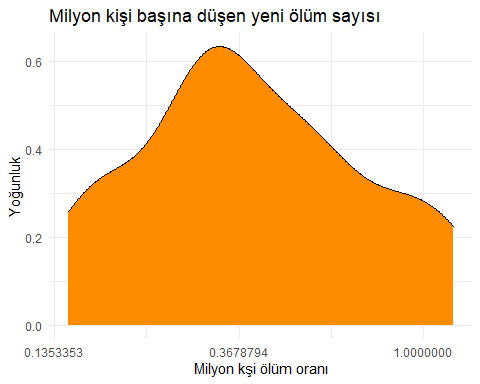
*Asya’nın yoğunluk grafiğine ilişkin yorumumuzda, en fazla gözlenen sıklığın 0.13533 lik bir ölümn/m.kişi oranıdır.*

**Avrupa’nın milyon kişi başına ölüm sayısına ait yoğunluk grafiği:**

library(dplyr)  
library(ggplot2)  
y\_corona %>%  
 filter(continent %in% "Europe") %>%  
 ggplot() +  
 aes(x = new\_deaths\_per\_million) +  
 geom\_density(adjust = 1L, fill = "#FF8C00") +  
 scale\_x\_continuous(trans = "log") +  
 labs(x = "Milyon kşi ölüm oranı", y = "Yoğunluk", title = "Milyon kişi başına düşen yeni ölüm sayısı") +  
 theme\_minimal()

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 5 rows containing non-finite values (stat\_density).



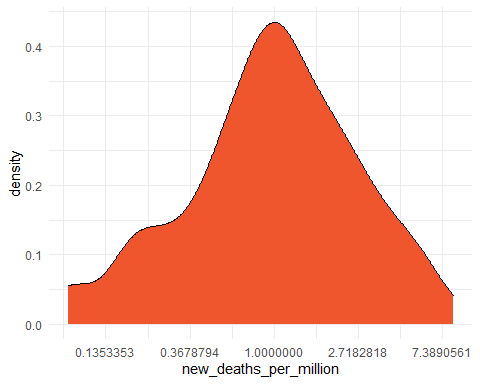
*Avrupa’ya ait yoğunluk grafiğinde aslında y tarafını yani yoğunluk değişkeninin tabanını 0.2 gibi bir değerle başlatıp x değişkenimizi de daha da yakınlaştırsaydık görebilirdik ki grağimiz görünüm olarak normal dağılıma yakınsıyor. Gerçi bu şekilde de fena bir grafik değil. Beklenildiği gibi ölüm oranı 1’e ve 0’a yakınsadıkça frekansı azalırken ortalama değer en fazla gözlenmiştir.*

**Kuzey Amerika’nın milyon kişi başına ölüm sayısına ait yoğunluk grafiği:**

library(ggplot2)  
ggplot(North\_America) +  
 aes(x = new\_deaths\_per\_million) +  
 geom\_density(adjust = 1L, fill = "#EF562D") +  
 scale\_x\_continuous(trans = "log") +  
 theme\_minimal()

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 448 rows containing non-finite values (stat\_density).



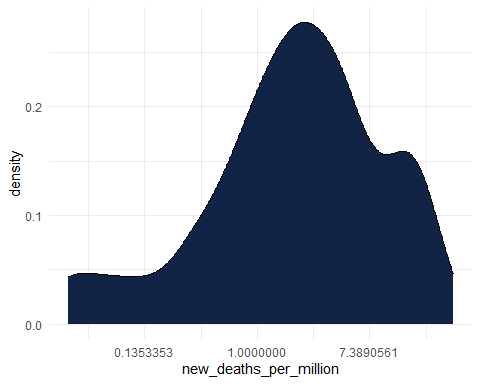
*Kuzey Amerika’nın yoğunluk grafiğine baktığımızda ise ece marka bayram çikolatalarını görmemek mümkün değil. Bu grafikte diğerlerinden farklı olarak, 7’ye ve 0 a yaklaşırken yoğunluğumuz düşüyor. Yoğunluğumuzun peak yaptığı oran konusunda ise 1 diyebiliriz. Oranımızın 7 ye kadar yükseldiğini görmekle de söyleyebiliriz ki milyon kişi başına düşen ölüm sayısı artıyor demek ki tedbirleri elden bırakıp maske ve mesafeye dikkat etmemişler ya da virüslü birisi sürekli insanlara tükürmüş ama bu sonuncu seçenek pek olası değil.*

**Güney Amerika’nın milyon kişi başına ölüm sayısına ait yoğunluk grafiği:**

library(ggplot2)  
ggplot(South\_America) +  
 aes(x = new\_deaths\_per\_million) +  
 geom\_density(adjust = 1L, fill = "#112446") +  
 scale\_x\_continuous(trans = "log") +  
 theme\_minimal()

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 109 rows containing non-finite values (stat\_density).



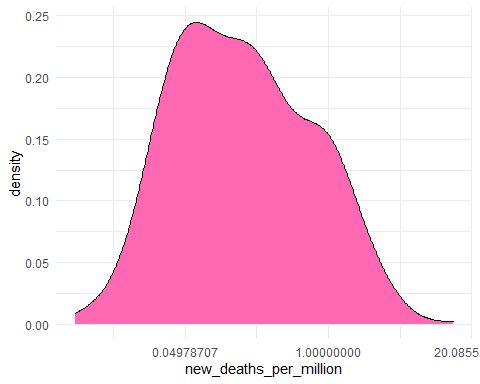
*Güney Amerika’da ise durumlar kuzeyden çok da farklı olmamakla birlikte medyana göre sağa çarpık olduğunu söyleyebiliriz. En fazla gözlenen sıklık ise tahmini olarak 3 gibi görünüyor.*

**Afrika’nın milyon kişi başına ölüm sayısına ait yoğunluk grafiği:**

library(ggplot2)  
ggplot(africa) +  
 aes(x = new\_deaths\_per\_million) +  
 geom\_density(adjust = 1.4, fill = "#FF69B4") +  
 scale\_x\_continuous(trans = "log") +  
 theme\_minimal()

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 872 rows containing non-finite values (stat\_density).



*Afrika’ya geldiğimizde ise durumlar Amerika kıtasına göre çok farklı. En fazla yoğunluk %10 civarlarında olurken yoğunluk %100’e ve %0’a yaklaştıkça azalmaktadır.*

*Genel manada kıtaların milyon kişi başına düşen ölüm miktarına ilişkin yoğunluk grafiğinde yoğunluğu en fazla hangi noktada toplandığı baz alınarak her kıta için, şu şekilde bir sıralama yapmak bizce doğru olur:*  *PAfrika < PAsya < PAvrupa < PK.Amerika < PG.Amerika*

# ShinyR Web Reporting

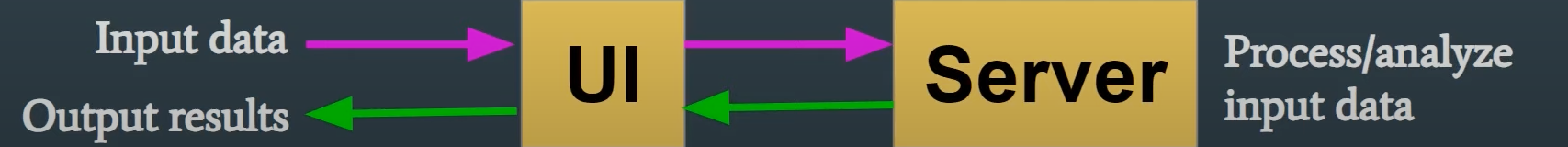
**Shiny**

Shiny, istatistiksel web aplikasyonlarını herhangi bir HTML, CSS, JavaScript bilgisi gerekmeden oluşturmaya yarayan bir kütüphanedir. Shiny, web elementlerini kullanmak için en popüler Frontend Frameworklerinden biri olan “Bootstrap” frameworkünü kullanır.

Shiny uygulamaları R-Studio ile localhost üzerinden çalıştırılabilir, linux işletim sistemine sahip bir sunucuya (digital ocean gibi) ShinyServer aracılığı ile kurulabilir, ya da bulut olarak shinyapps.io üzerinde çalışabilir.

**Çalışma Mantığı**

Kullanıcı arayüzü üzerinden alınan input değerleri servera gönderilir ve serverda çalışan R üzerinde işlemler yapılarak oluşan çıktı tekrar kullanıcı arayüzüne yansıtılır. Son kullanıcı serverda yapılan işlemleri görmez. Sadece kullanıcının istekleri doğrultusunda oluşan çıktıları görür.



**Genel Yapı**

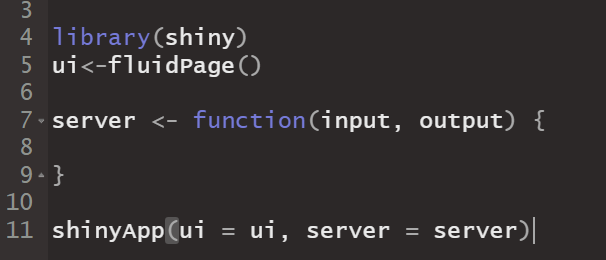
Shiny kod üzerinde 3 bileşenden oluşur.

1. UI (User Interface) - Kullanıcı arayüzü
2. Server
3. ShinyApp Fonksiyonu

**UI:** Frontend dediğimiz son kullanıcının etkileşim halinde olduğu bölüm.

**Server:** Backend dediğimiz, son kullanıcın göremediği R üzerinde işlemlerin yapıldığı bölüm.

İşlemler yapıldıktan sonra çıktılar UI üzerinde gözükür. Alttaki resimde Shiny'nin bu genel yapısını R Studio üzerinde görebilirsiniz.



Tek dosya olarak “app.R” olarak yada, iki farklı dosya olarak “ui.R”, “server.R” şeklinde kaydedilebilir. İki farklı dosya olduğundu shinyApp() fonksiyonunu tanımlamaya gerek yoktur.

**Fonksiyonlar:**

Fonksiyonlar genel olarak JavaScripte benzer şekilde, ilk kelime küçük, ikinci kelimenin ilk harfi büyük harfle “Camel Case” düzeninde yazılır.

Örneğin: tabPanel()

**UI’da Kullanılan Fonksiyonlar**

**navbarPage():** Navigation Bar oluşturmaya yarar.

**tabsetPanel():** Uygulamaların kullanıcı arayüzlerini ayırmak için kullanılır.

**tabPanel():** tabsetPanel() içindeki bölümleri oluşturmaya yarar.

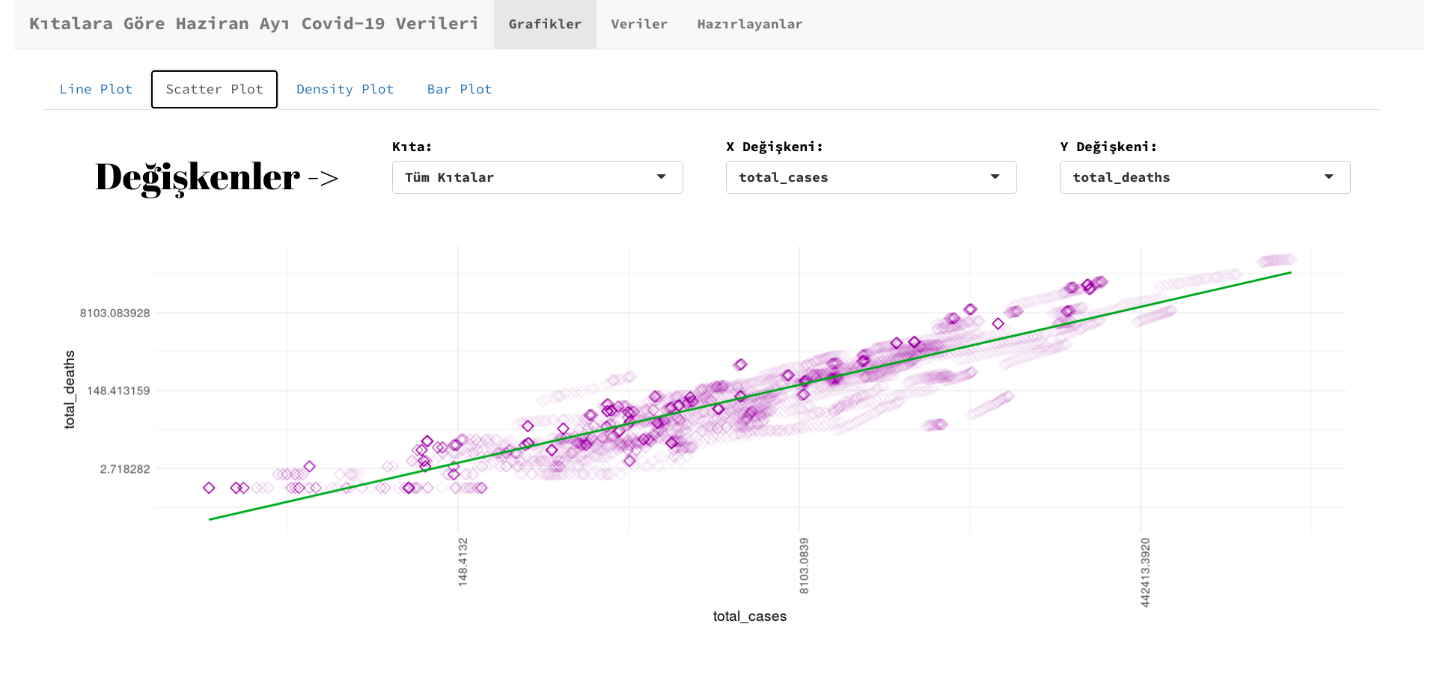
**mainPanel():** Arayüz üzerindeki ana kısım.

**fluidRow():** Grid düzeninde bir satır oluşturur.

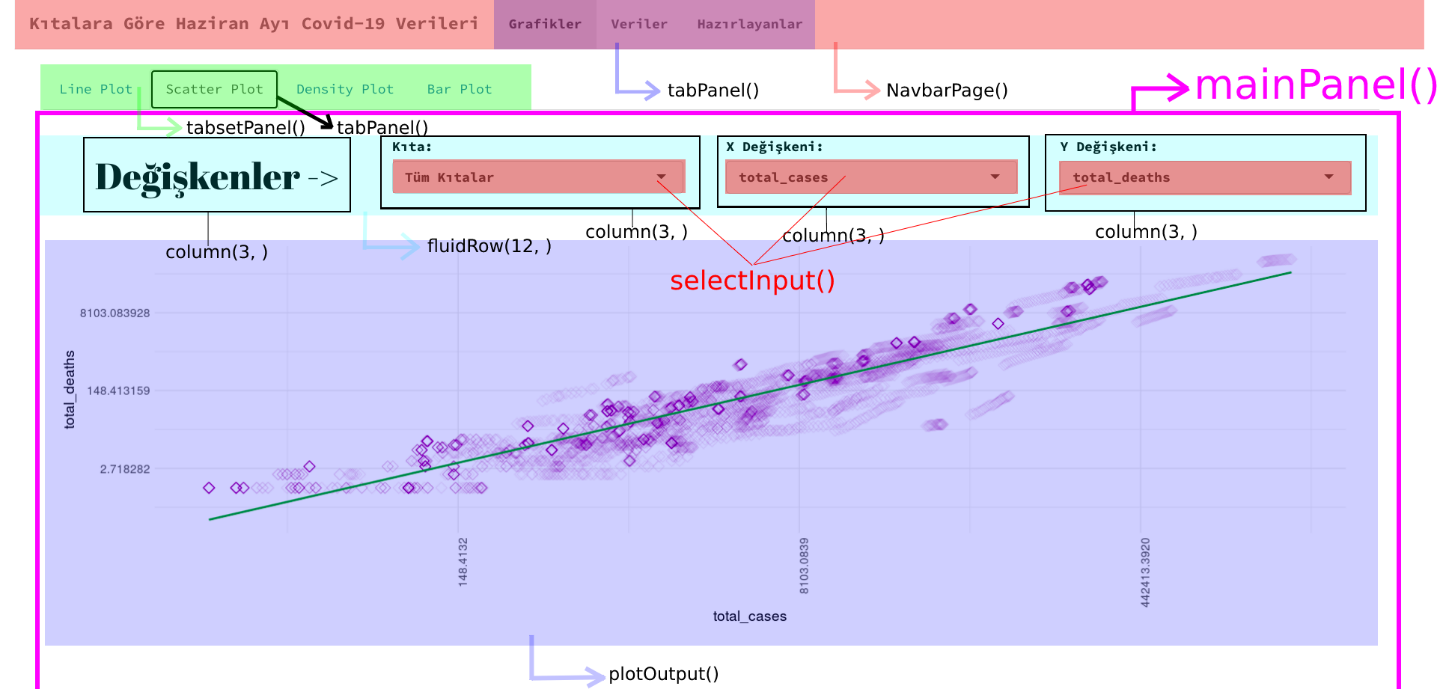
**column():** Grid düzeninde oluşturulan satıra içindeki diğer elementlerin nasıl bir düzende yerleşeceğini ayarlar.

**selectInput():** Bu fonksiyonda belirlenen input değeri, UI üzerinde belirlenen girdiyi servera aktarmak için kullanılır.

**plotOutput():** Serverda renderlanan plotun grafiğin çıktısını arayüz üzerinde göstermeye yarar.



Elimizde böyle bir arayüz var ise basitçe elementleri aşağıdaki resimdeki gibi olur.



**Server’da Kullanılan Fonksiyonlar**

**renderPlot():** UI daki selectInput() fonksiyonunun içindeki değerleri izler. Herhangi bir değişiklik olduğunda direk yeni bir plot renderlar.

**renderDataTable():** Reaktif veri tabloları yaratmaya yarar.

**reactive():** Reaktif fonkiyonlar yaratmaya yarar.

## Kaynakça:

[Stack Over Flow](https://stackoverflow.com/questions/60083062/tidyrspread-error-each-row-of-output-must-be-identified-by-a-unique-combina)

[RStudio](https://www.rstudio.com/blog/)

[Book Down](https://bookdown.org/content/2096/)

[dplyr.tidyverse.org](https://dplyr.tidyverse.org/reference/dplyr_tidy_select.html)

[Geeks for Geeks](https://www.geeksforgeeks.org)

[tidyverse.org](https://www.tidyverse.org/)

[Cran.Rproject](https://cran.r-project.org/web/packages/esquisse/vignettes/get-started.html)