1. plotly 시각화 만들기

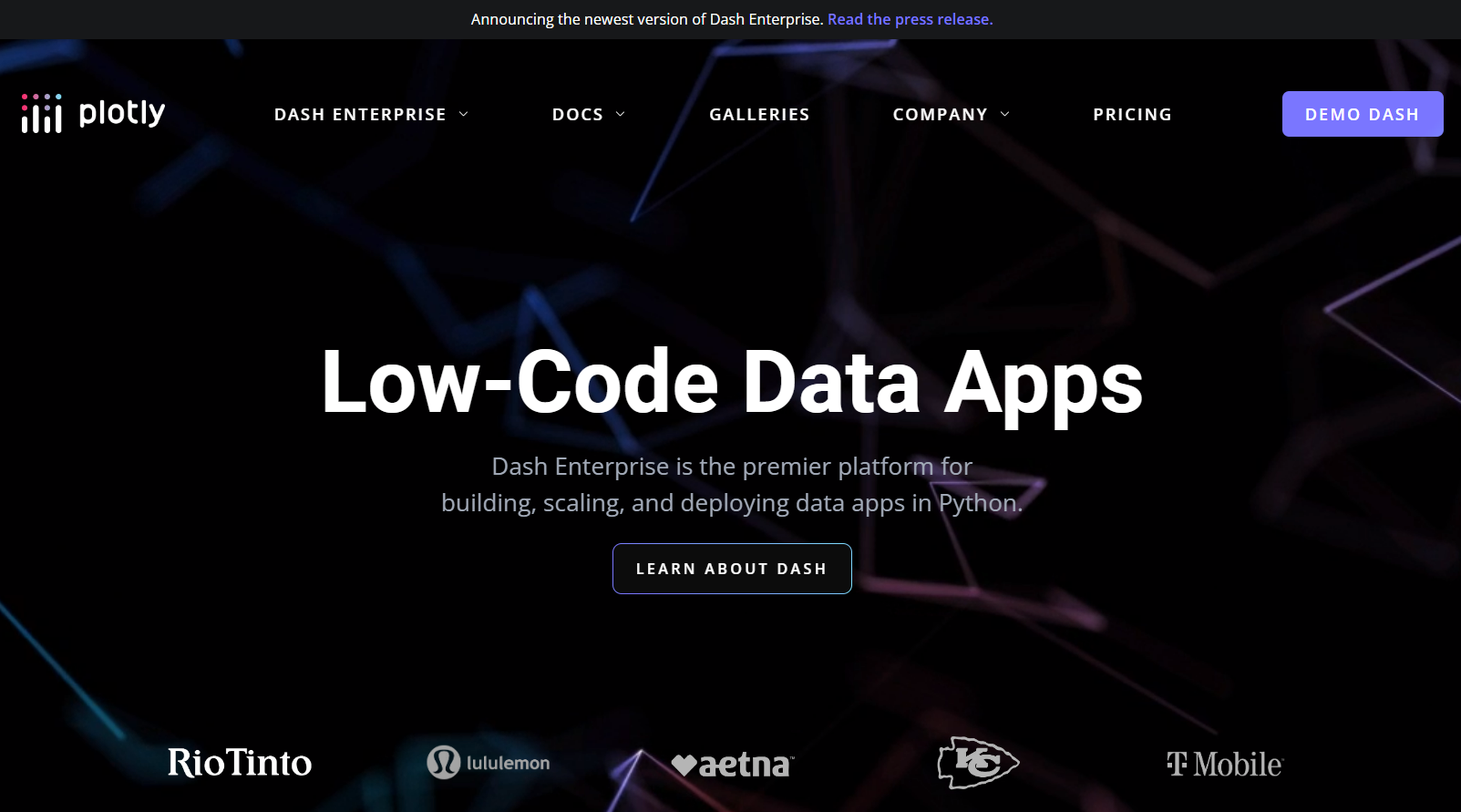
데이터 분석을 다루는 많은 교육코스나 서적에서 데이터의 시각화는 R의 경우 R base나 ggplot2 패키지를 사용하여는 방법, 파이썬의 경우 matplotlib이나 seaborn 패키지를 사용하는 방법을 위주로 설명한다. 이러한 방법들은 데이터 시각화 방법이 간단하고 그 품질이 좋은 편이기 때문에 많이 사용되고 있지만 정적(Static) 시각화라는 한계를 가진다. 정적 시각화는 대부분 이미지로 저장되며 인포그래픽(Infographic)이라고도 불리며, 일반적으로 문서나 인쇄물에 많이 사용되고 웹에 게시되는 이미지로도 사용된다. 그렇기 때문에 대부분 png, jpg, pdf 등의 벡터 혹은 픽셀 이미지 파일로 제공된다. 정적 데이터 시각화는 데이터 분석가의 의도에 맞춰 작성되기 때문에 데이터 분석가의 데이터 분석 관점에 의존적일 수 밖에 없으며 시각화를 사용하는 사용자의 의도에 따른 해석은 매우 제한될 수 밖에 없다.

이러한 제한점을 극복하기 위해 사용되는 데이터 시각화 방법이 동적(Dynamic) 시각화 혹은 인터랙티브(Interactive) 시각화라고 하는 방법이다. 이 동적 시각화는 시각화를 사용하는 사용자의 의도에 따라 데이터를 다각적 관점에서 살펴볼 수 있다는 점이 특징이다. 사용자의 의도에 따라 데이터가 동적으로 변동되어야 하기 때문에 데이터 시각화가 많이 사용되는 인쇄물 형태 매체에서 사용이 어렵고 웹을 통해 사용되어야 그 장점을 충실히 사용할 수 있다. 따라서 동적 시각화는 웹 사이트에서 제공하는 대시보드(DashBoard)의 형태로 제공되는 것이 일반적이기 때문에 동적 시각화를 위해서는 R이나 파이썬에서 주로 사용되는 시각화 패키지가 아닌 동적 시각화 전용 패키지를 사용해야 한다. R에서는 동적 시각화를 위해 rbokeh, highcharter 등이 사용되었고 파이썬에서는 bokeh, hvplot 등을 사용되었다. 하지만 R과 파이썬 모두에서 사용되는 plotly 패키지가 등장함으로써 R과 파이썬의 동적 시각화 패키지 시장에서 매우 빠르게 사용자층을 넓혀가고 있다.

하지만 정적 시각화와 동적 시각화의 어느것이 더 효용성이 있는지를 단언할 수 없다. 데이터 시각화가 사용되는 매체, 데이터 시각화를 보는 대상, 데이터 시각화에서 보여주고자 하는 스토리에 따라서 정적 시각화를 사용해야 할 때와 동적 시각화를 사용해야 할 때를 적절히 선택해야 한다.

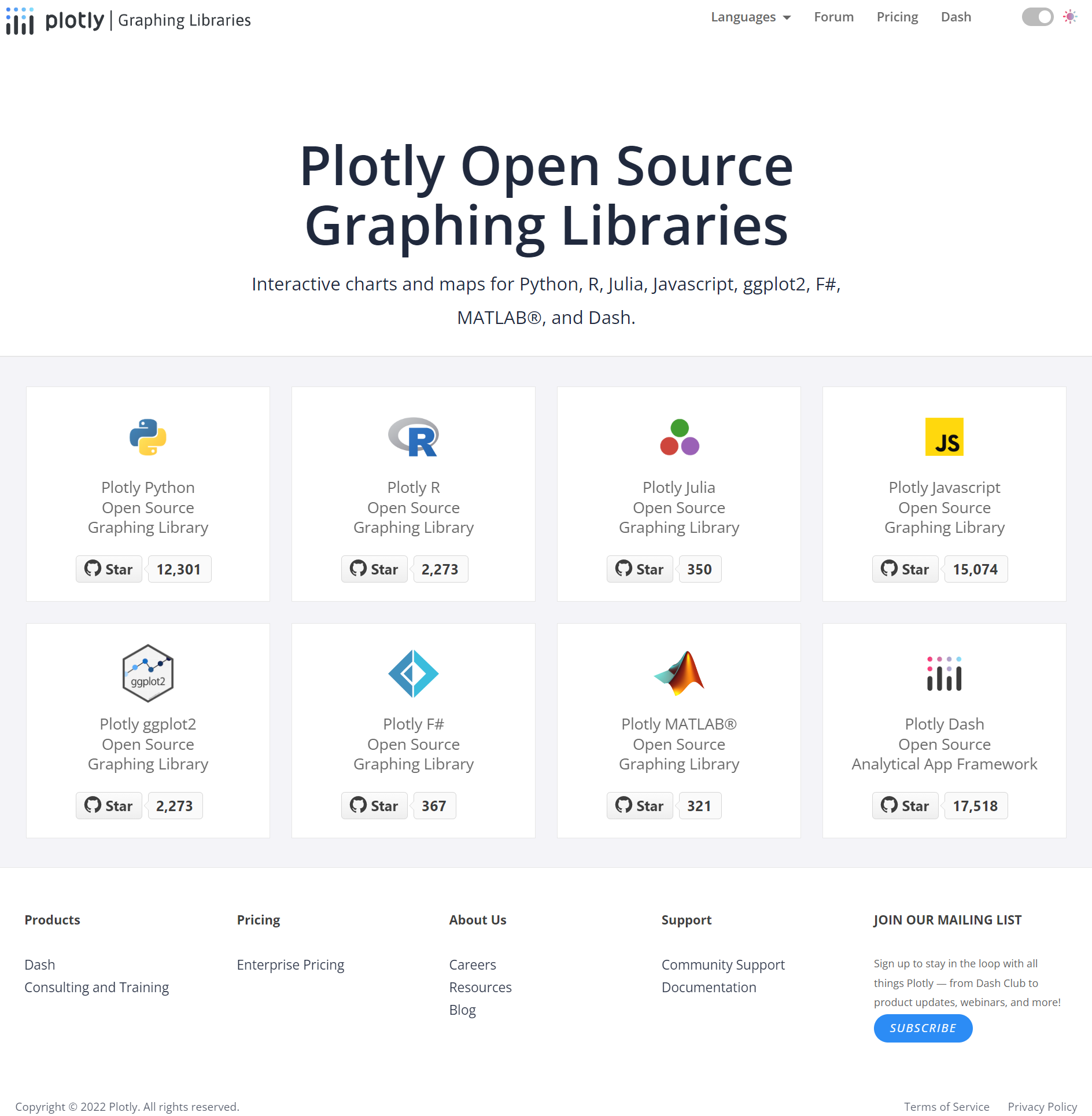
# plotly란?

‘plotly’는 캐나다 몬트리올에 본사를 두고 있는 데이터 시각화 전문 회사의 이름이다. 이 회사는 2012년 처음 설립되었는데 데이터 전문 분석 도구와 데이터 시각화 전문 도구를 개발하여 보급한다. ’plotly’는 이 책에서 설명할 R과 파이썬의 plotly 패키지 외에도 R, 파이썬, 줄리아 등의 개발도구에서 사용가능한 데이터 대시보드 플랫폼인 dash를 비롯해 dash 플랫폼으로 개발된 웹페이지를 배포하기 위한 ’Dash Enterprise’, plotly를 기반으로 온라인 동적 시각화를 만드는 ‘Chart Studio Cloud’ 등의 서비스를 제공하고 있다. 이 중 가장 유명한 제품이 회사의 이름에서도 나타나듯이 plotly 시각화 패키지이다.



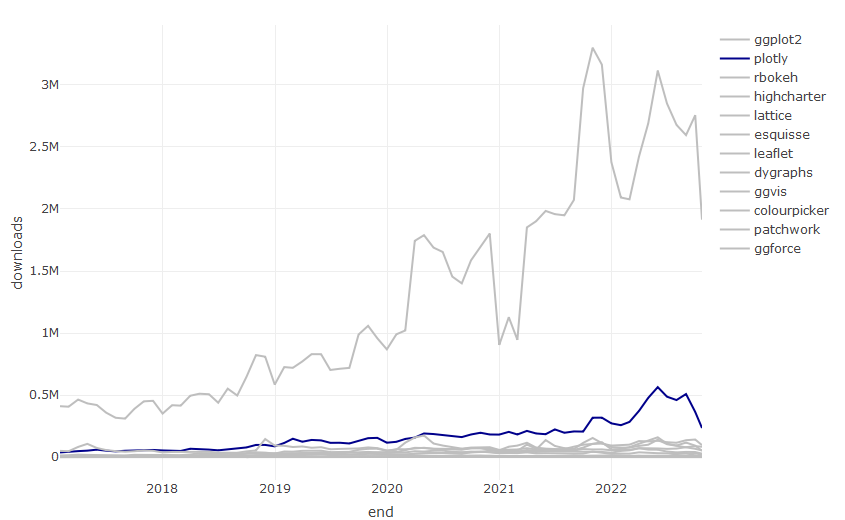
plotly 홈페이지 화면

plotly 패키지는 R, 파이썬, Julia, Java Script, F#, MATLAB 등의 다양한 언어에서 사용이 가능하도록 각각의 언어에 바인딩되는 패키지를 개발하여 제공하고 있다. plotly에서 제공하는 데이터 시각화는 산점도, 선 그래프와 같은 기본 차트(Basic Chart), 박스 플롯, 히스토그램과 같은 통계 차트(Statistical Chart), 히트맵, 삼각플롯(Ternary Plot)과 같은 과학 차트(Scienctific Chart), 시계열 차트, 캔들 차트와 같은 재정 차트(Finantial Chart) 등의 다양한 차트와 플롯을 제공한다.



plotly 패키지 홈페이지

다음의 R 그래픽 패키지 다운로드 현황에서 보듯이 여전히 R에서 가장 많이 사용되고 있는 그래픽 패키지는 ggplot2이지만 plotly는 2021년 하반기부터 다운로드가 늘고 있고 다른 그래픽 패키지에 비해서는 압도적인 다운로드 수를 보인다.



plotly의 다운로드 수 증가는 파이썬에서도 유사한 흐름을 보인다. 다음의 그림에서도 보이듯이 파이썬에서 많이 사용되는 시각화 패키지 중 가장 다운로드수가 많은 것은 역시나 matplotlib이다. 하지만 matplotlib을 제외하면 2022년 중순까지만 해도 seaborn 패키지의 다운로드 수가 가장 많았으나 이 후 plotly가 seaborn과 대등하거나 오히려 더 다운로드가 많은 날이 상당히 눈에 띈다. 하지만 파이썬에서 동적 시각화를 지원하는 bokeh나 hvplot보다는 월등히 많은 다운로드를 보인다. 결국 파이썬에서도 동적 시각화에서는 plotly가 가장 많이 사용되는 패키지인 것이다.

# 예제 데이터 Import와 전처리

먼저 plotly를 사용하여 시각화를 실습하는데 필요한 데이터 셋 두 가지를 전처리 하겠다.

## Covid19 데이터 셋

첫번째 데이터 셋은 2020년 1월부터 기록된 전세계 국가의 코로나19 발병 관련 데이터이다. 이 데이터는 Github에서 다양한 전세계 데이터를 배포하는 ’Open World in Data’에서 제공하는 ’COVID-19 Dataset by Our World in Data’를 사용한다.[[1]](#footnote-27) 이 데이터는 온라인으로 매일 업데이트되기 때문에 다운로드 시점에 따라 시각화 결과가 책과 다소 달라질 수 있다.[[2]](#footnote-29)

OWID에서 제공하는 데이터를 활용하여 4개의 데이터 셋을 만든다. 첫 번째 데이터 셋은 OWID에서 제공하는 원본 데이터를 가져와서 R에 로딩하는 원본 데이터 셋으로 ‘df\_covid19’ 데이터프레임에 저장한다. ‘df\_covid19’ 데이터 프레임은 2020년 1월 1일부터 기록되어 있기 때문에 데이터가 다소 많다. 따라서 이 데이터 중에 최근 100일간의 데이터와 한국과 각 대륙 데이터만을 필터링한 데이터 셋을 두 번째 데이터 셋인 ‘df\_covid19\_100’ 데이터프레임으로 저장한다. 세 번째 데이터 셋은 100일간의 데이터 셋을 넓은 형태의 데이터 셋으로 변환한 ‘df\_covid19\_100\_wide’로 저장한 데이터프레임이다. 네 번쨰는 2년 넘게 기록된 Covid19 데이터 셋의 각종 데이터를 국가별 요약 통계치를 산출하여 저장한 ’df\_covid19\_stat’ 데이터프레임이다.

## 데이터 전처리를 위한 패키지 설치 및 로딩  
if(!require(readr)) {  
 install.packages('readr')  
 library(readr)  
}  
  
if(!require(lubridate)) {  
 install.packages('lubridate')  
 library(lubridate)  
}  
  
if(!require(tidyverse)) {  
 install.packages('tidyverse')  
 library(tidyverse)  
}  
  
## 1. covid19 원본 데이터 셋 로딩  
## covid19 데이터 로딩(파일을 다운로드 받은 경우)  
# df\_covid19 <- read\_csv(file = "데이터저장경로/owid-covid-data.csv",  
# col\_types = cols(Date = col\_date(format = "%Y-%m-%d")  
# )  
# )  
## covid19 데이터 로딩(온라인에서 바로 로딩할 경우)  
df\_covid19 <- read\_csv(file = "https://covid.ourworldindata.org/data/owid-covid-data.csv",  
 col\_types = cols(Date = col\_date(format = "%Y-%m-%d")  
 )  
 )  
## 2. 전체 데이터셋 중 최근 100일간의 데이터를 필터링한 df\_covid19\_100 생성  
df\_covid19\_100 <- df\_covid19 |>   
 ## 한국 데이터와 각 대륙별 데이터만을 필터링  
 filter(iso\_code %in% c('KOR', 'OWID\_ASI', 'OWID\_EUR', 'OWID\_OCE', 'OWID\_NAM', 'OWID\_SAM', 'OWID\_AFR')) |>  
 ## 읽은 데이터의 마지막 데이터에서 100일전 데이터까지 필터링  
 filter(date >= max(date) - 100) |>  
 ## 국가명을 한글로 변환  
 mutate(location = case\_when(  
 location == 'South Korea' ~ '한국',   
 location == 'Asia' ~ '아시아',   
 location == 'Europe' ~ '유럽',   
 location == 'Oceania' ~ '오세아니아',   
 location == 'North America' ~ '북미',   
 location == 'South America' ~ '남미',   
 location == 'Africa' ~ '아프리카')) |>  
 ## 국가 이름의 순서를 설정   
 mutate(location = fct\_relevel(location, '한국', '아시아', '유럽', '북미', '남미', '아프리카', '오세아니아')) |>  
 ## 날짜로 정렬  
 arrange(date)  
  
  
## 3. df\_covid19\_100을 한국과 각 대륙별열로 배치한 넓은 형태의 데이터프레임으로 변환  
df\_covid19\_100\_wide <- df\_covid19\_100 |>  
 ## 날짜, 국가명, 확진자와, 백신접종완료자 데이터만 선택  
 select(date, location, new\_cases, people\_fully\_vaccinated\_per\_hundred) |>  
 ## 열 이름을 적절히 변경  
 rename('date' = 'date', '확진자' = 'new\_cases', '백신접종완료자' = 'people\_fully\_vaccinated\_per\_hundred') |>  
 ## 넓은 형태의 데이터로 변환  
 pivot\_wider(id\_cols = date, names\_from = location,   
 values\_from = c('확진자', '백신접종완료자')) |>  
 ## 날짜로 정렬  
 arrange(date)  
  
## 4. covid19 데이터를 국가별로 요약한 df\_covid19\_stat 생성  
df\_covid19\_stat <- df\_covid19 |>   
 group\_by(iso\_code, continent, location) |>  
 summarise(인구수 = max(population, na.rm = T),   
 인당GDP = max(gdp\_per\_capita, na.rm = T),  
 전체확진자수 = sum(new\_cases, na.rm = T),  
 전체사망자수 = sum(new\_deaths, na.rm = T),   
 십만명당중환자실 = last(icu\_patients\_per\_million),  
 재생산지수 = last(reproduction\_rate),  
 봉쇄지수 = max(stringency\_index),   
 전체검사자수 = max(total\_tests, na.rm = T),   
 신규검사자수 = sum(new\_tests, na.rm = T),  
 전체백신접종자수 = max(total\_vaccinations, na.rm = T),  
 백신접종자완료자수 = max(people\_fully\_vaccinated, na.rm = T),  
 부스터접종자수 = max(total\_boosters, na.rm = T),  
 인구백명당백신접종완료률 = max(people\_fully\_vaccinated\_per\_hundred, na.rm = T),  
 인구백명당부스터접종자수 = max(total\_boosters\_per\_hundred, na.rm = T)  
 ) |>   
 ungroup() |>  
 mutate(십만명당사망자수 = round(전체사망자수 / 인구수 \*100000, 5),  
 백신접종완료률 = 백신접종자완료자수 / 인구수)  
  
## 여백 설정을 위한 리스트 설정  
margins <- list(t = 50, b = 25, l = 25, r = 25)

파이썬 코드

import pandas as pd

1. <https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-data.csv> [↑](#footnote-ref-27)
2. 필자의 블로그에 업로드된 데이터를 활용하면 책과 동일한 결과를 얻을 수 있다. [↑](#footnote-ref-29)