Examine the COVID-19 pandemic with data

Using R and public data resources

Vincent J. Carey* Charles Morefield† John Mallery‡ Sean Davis§

2020-04-09

Contents

*Harard Medical School, Channing Laboratory, Brigham and Women's Hospital
†Arctan, Inc.
‡MIT Artificial Intelligence Laboratory
§National Cancer Institute, National Institutes of Health

Part I

Introduction

1 Motivation

The COVID-19 pandemic is ongoing. The situation on the ground is changing daily as captured by data reported around the world. The sars2pack package aims to:

- Provide timely, computable, easily accessible data for research, policy-making, and educational purposes.
- Promote easy computational experimentation with COVID-19 data
- Serve as a source of documentation and education for available COVID-19 analysis and visualization approaches.
- House recipes for regularly updated data products such as spreadsheets and maps for use by non-R-savvy data consumers.
- Collect interesting data stories along with code as data science training resources for the many biomedical researchers who cannot currently perform experiments

1.1 Origins of this package

John C. Mallery conveyed code of Charles Morefield to harvest COVID-19 time series data. Vince Carey then started an R package/github repo to manage the relevant code. The package was named `sars2pack` in hopes of avoiding name conflict with many other packages while remaining descriptive and focused.

2 Quick start

2.1 Installation

```
BiocManager::install('seandavi/sars2pack')
```

2.2 COVID-19 resources in this package

The COVID-19 data in this package are, right now, focused toward time-series descriptions of confirmed cases, deaths, testing, and recovered cases. **There is no requirement that this remain the case**. Contributions of additional data resources or simple accessor functions will only add to our abilities to use data science and modeling to understand COVID-19.

Request for help: We are looking for additional data resources. Consider a pull request (or an issue for non-programmer types) with suggestions.

3 TODO: give background on datasets

4 Epidemic time series data

Usage of each of the time series datasets follows a similar pattern.

1. Fetch a tidy `tbl_df` using a function such as `jhu_data()`
2. In the resulting `tbl_df`, the columns `date` (of type `date`) and `count` of type `numeric` columns are standard.
3. Additional columns describe locations, subsets of data (such as `confirmed`, `deaths`, `recovered`) and vary from dataset to dataset.

Regardless of the original format of the data, the `sars2pack` datasets are presented as tidy data to facilitate `dplyr`, `ggplot`, and other fluid analysis approaches to apply directly.

4.1 Access data

This section briefly introduces how to access the data resources in this package. Note that many of the functions below **require a network connection** to get updated data.

4.1.1 JHU Dataset

```
jhu = jhu_data()
class(jhu)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dim(jhu)
```

```
## [1] 60450     7
```

Column names include:

```
colnames(jhu)
```

```
## [1] "ProvinceState" "CountryRegion" "Lat"           "Long"
## [5] "date"          "count"         "subset"
```

And a very small subset of the data.

```
head(jhu,3)
```

```
## # A tibble: 3 x 7
##   ProvinceState CountryRegion   Lat  Long date       count subset
##   <chr>         <chr>         <dbl> <dbl> <date>     <dbl> <chr>
## 1 <NA>          Afghanistan      33    65 2020-01-22     0 confirmed
## 2 <NA>          Afghanistan      33    65 2020-01-23     0 confirmed
## 3 <NA>          Afghanistan      33    65 2020-01-24     0 confirmed
```

4.1.2 USAFacts Dataset

```
usa_facts = usa_facts_data()
class(usa_facts)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dim(usa_facts)
```

```
## [1] 434792      7
```

Column names include:

```
colnames(usa_facts)
```

```
## [1] "county_fips" "county"      "state"       "state_fips"  "subset"
## [6] "date"        "count"
```

And a very small subset of the data.

```
head(usa_facts,3)
```

```
## # A tibble: 3 x 7
##   county_fips county                state state_fips subset    date       count
##         <dbl> <chr>                 <chr>      <dbl> <chr>     <date>     <dbl>
## 1           0 Statewide Unallocated AL             1 confirmed 2020-01-22     0
## 2           0 Statewide Unallocated AL             1 confirmed 2020-01-23     0
## 3           0 Statewide Unallocated AL             1 confirmed 2020-01-24     0
```

4.1.3 NYTimes datasets

```
nytimes_state = nytimes_state_data()
class(nytimes_state)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dim(nytimes_state)
```

```
## [1] 3988    5
```

Column names include:

```
colnames(nytimes_state)
```

```
## [1] "date"   "state"  "fips"   "count"  "subset"
```

And a very small subset of the data.

```
head(nytimes_state,3)
```

```
## # A tibble: 3 x 5
##   date       state      fips  count subset
##   <date>     <chr>      <chr> <dbl> <chr>
## 1 2020-01-21 Washington 00053     1 confirmed
## 2 2020-01-22 Washington 00053     1 confirmed
## 3 2020-01-23 Washington 00053     1 confirmed
```

```
nytimes_county = nytimes_county_data()
class(nytimes_county)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dim(nytimes_county)
```

```
## [1] 81440     6
```

```
colnames(nytimes_county)
```

```
## [1] "date"   "county" "state"  "fips"   "count"  "subset"
```

5 Use cases

5.1 Basic data exploration

In this section, we will be using a combination of [dplyr] and [ggplot2] to explore the COVID-19 global data from JHU. For details on this dataset, see the help using `?jhu_data`.

The next line of code will do a (set of) network calls to fetch the most up-to-date dataset from the JHU github repository.

```
jhu = jhu_data()
head(jhu,3)
```

```
## # A tibble: 3 x 7
##   ProvinceState CountryRegion   Lat  Long date       count subset
##   <chr>         <chr>         <dbl> <dbl> <date>     <dbl> <chr>
## 1 <NA>          Afghanistan      33    65 2020-01-22     0 confirmed
## 2 <NA>          Afghanistan      33    65 2020-01-23     0 confirmed
## 3 <NA>          Afghanistan      33    65 2020-01-24     0 confirmed
```

We now want to ask a series of questions about the dataset.

- **How many records are in the dataset?**

```
nrow(jhu)
```

```
## [1] 60450
```

- **How many different countries are represented?**

```
length(unique(jhu$CountryRegion))
```

```
## [1] 184
```

Most records have no listing for `ProvinceState` column. Let's look at a few of those to get an idea of what is there when not empty:

- **What is in the `ProvinceState` column?**

To answer this question, we will be using `dplyr`, so some familiarity with that package will be helpful to follow this code.

```
jhu %>%
    dplyr::filter(!is.na(ProvinceState)) %>%
    dplyr::select(ProvinceState, CountryRegion) %>%
    unique() %>%
    head(10)
```

```
## # A tibble: 10 x 2
##    ProvinceState                CountryRegion
##    <chr>                        <chr>
##  1 Australian Capital Territory Australia
##  2 New South Wales              Australia
##  3 Northern Territory           Australia
```

```
##  4 Queensland                Australia
##  5 South Australia           Australia
##  6 Tasmania                  Australia
##  7 Victoria                  Australia
##  8 Western Australia         Australia
##  9 Alberta                   Canada
## 10 British Columbia          Canada
```

We still have not looked at the most valuable information, the `date` and `count` columns in any detail.

- **What is the current count of confirmed cases by country, ordered by highest count down?**

There is a lot to unpack in the next code block, but the results are quite useful. We will use the DT package to make the dataset searchable and sortable.

```
library(DT)
latest_jhu_data = jhu %>%
    dplyr::filter(subset=='confirmed' & is.na(ProvinceState)) %>%
    dplyr::group_by(CountryRegion) %>%
    dplyr::slice(which.max(date)) %>%
    dplyr::arrange(desc(count))
DT::datatable(latest_jhu_data, rownames=FALSE)
```

Show 10 entries Search:

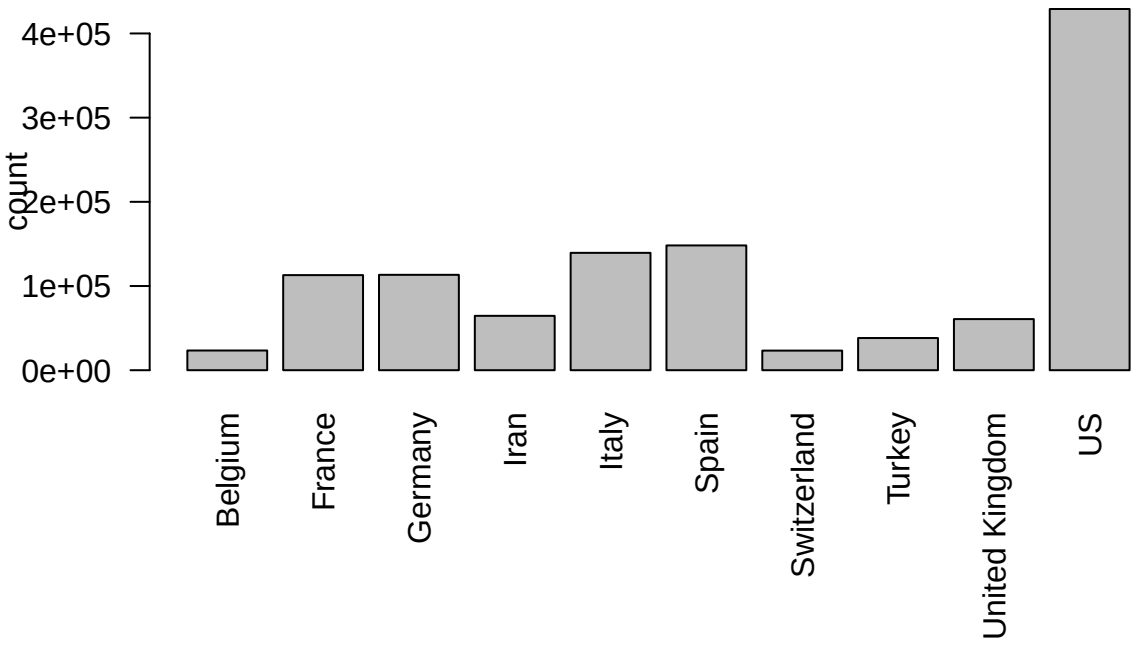
ProvinceState	CountryRegion	Lat	Long	date	count	subset
	US	37.0902	-95.7129	2020-04-08	429052	confirmed
	Spain	40	-4	2020-04-08	148220	confirmed
	Italy	43	12	2020-04-08	139422	confirmed
	Germany	51	9	2020-04-08	113296	confirmed
	France	46.2276	2.2137	2020-04-08	112950	confirmed
	Iran	32	53	2020-04-08	64586	confirmed
	United Kingdom	55.3781	-3.436	2020-04-08	60733	confirmed
	Turkey	38.9637	35.2433	2020-04-08	38226	confirmed
	Belgium	50.8333	4	2020-04-08	23403	confirmed
	Switzerland	46.8182	8.2275	2020-04-08	23280	confirmed

Showing 1 to 10 of 181 entries Previous 1 2 3 4 5 … 19 Next

Note: I included a little `is.na` in the filtering above to remove records where country data are split out over subparts. We revisit this below.

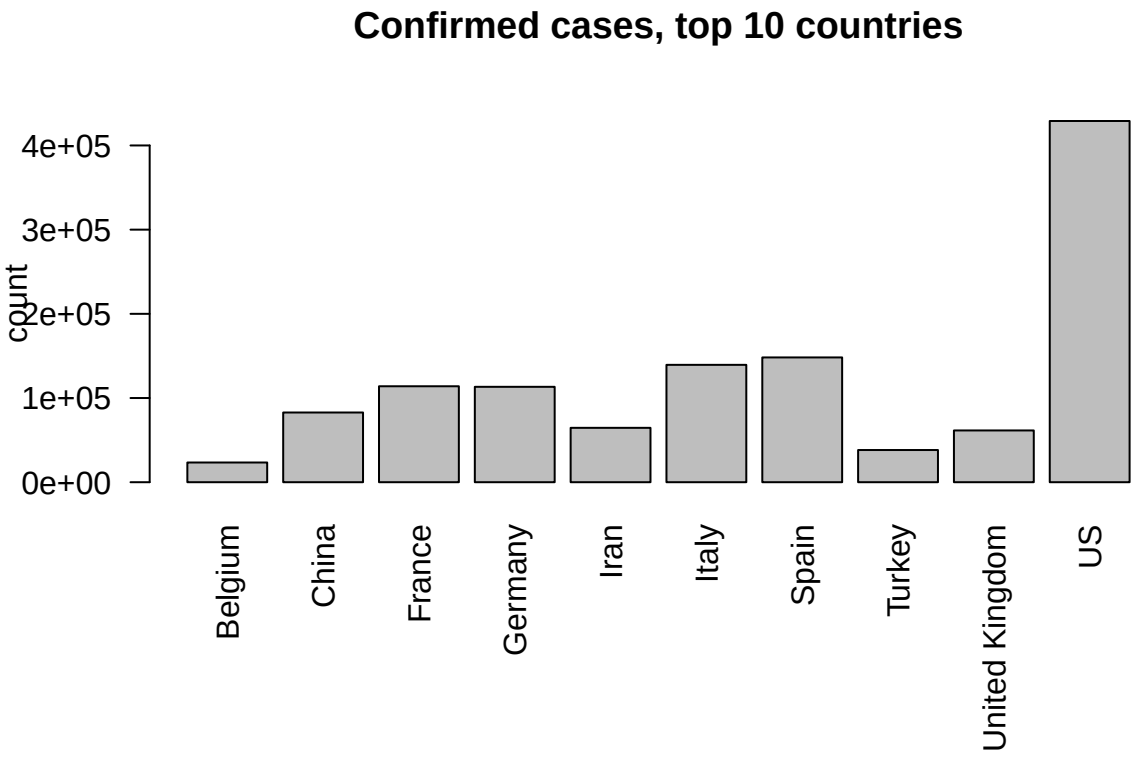
The data here could be usefully displayed as a graph as well.

```
par(las=2, mar=c(8,5,5,1))
barplot(count ~ CountryRegion, xlab = '',
        data=head(latest_jhu_data,10),
        main='Confirmed cases, top 10 countries')
```



We note here that China is not shown. That is because we limited the data to only rows that had empty ProvinceState records. To add those records back in, we sum all the China rows (and those of other countries like Australia, etc.) by country and then perform similar work to produce a final plot.

```
latest_jhu_data = jhu %>%
    dplyr::filter(subset=='confirmed') %>%
    dplyr::select(-c(ProvinceState,Lat,Long)) %>%
    dplyr::group_by(CountryRegion,date) %>%
    dplyr::summarize(count = sum(count)) %>%
    dplyr::slice(which.max(date)) %>%
    dplyr::arrange(desc(count))
par(las=2, mar=c(8,5,5,1))
barplot(count ~ CountryRegion, xlab = '',
        data=head(latest_jhu_data,10),
        main='Confirmed cases, top 10 countries')
```

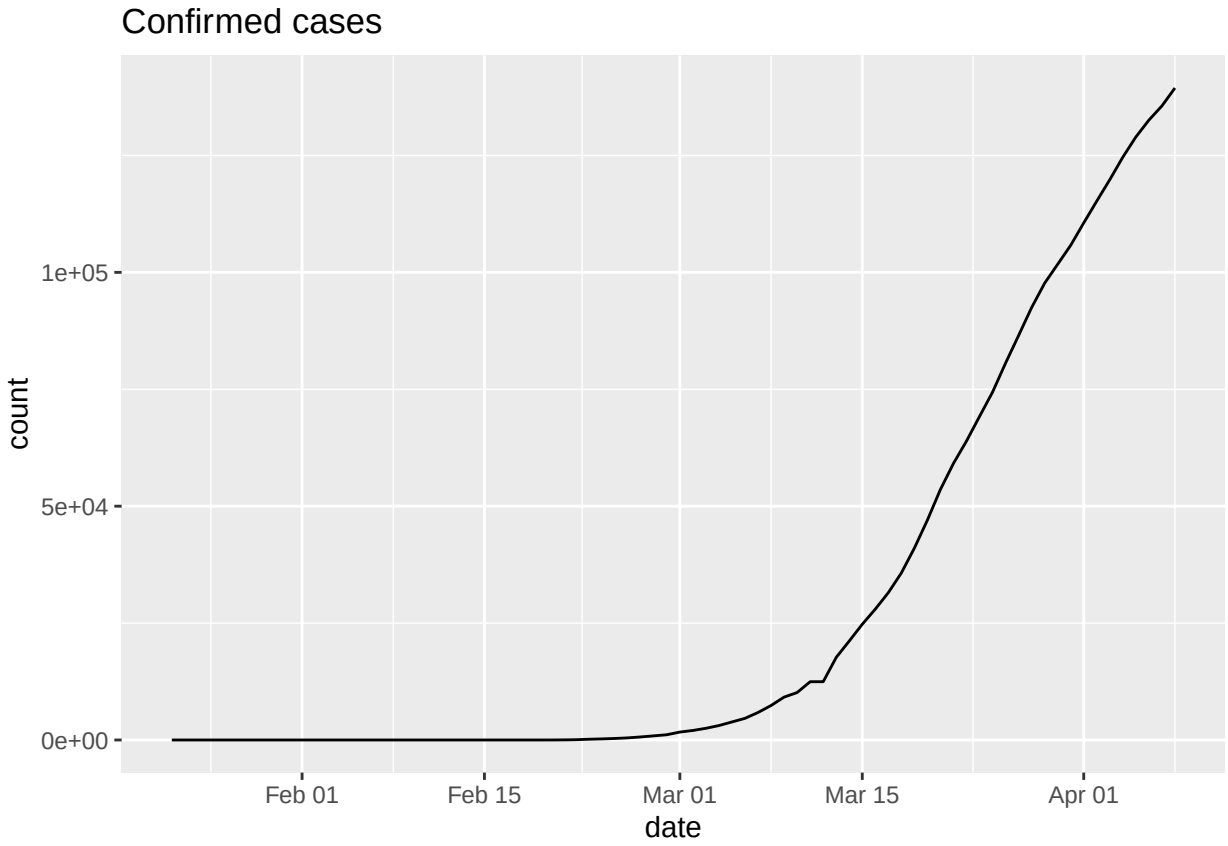


5.2 Visualize time series data

Up to now, we have ignored the time series aspects of the data and have sliced the dataset by country. In this section, we will be using dplyr and ggplot2 to visualize disease infection and deaths over time.

- **How have the cases in Italy changed over time?**

```
library(ggplot2)
italy_cc_ts = jhu %>%
    dplyr::filter(CountryRegion == 'Italy' & subset=='confirmed')
ggplot(italy_cc_ts,aes(x=date, y=count)) +
    geom_line() +
    ggtitle('Confirmed cases')
```



- **How do the confirmed cases in China, US, Italy, Spain, Germany, and Russia compare over time?**

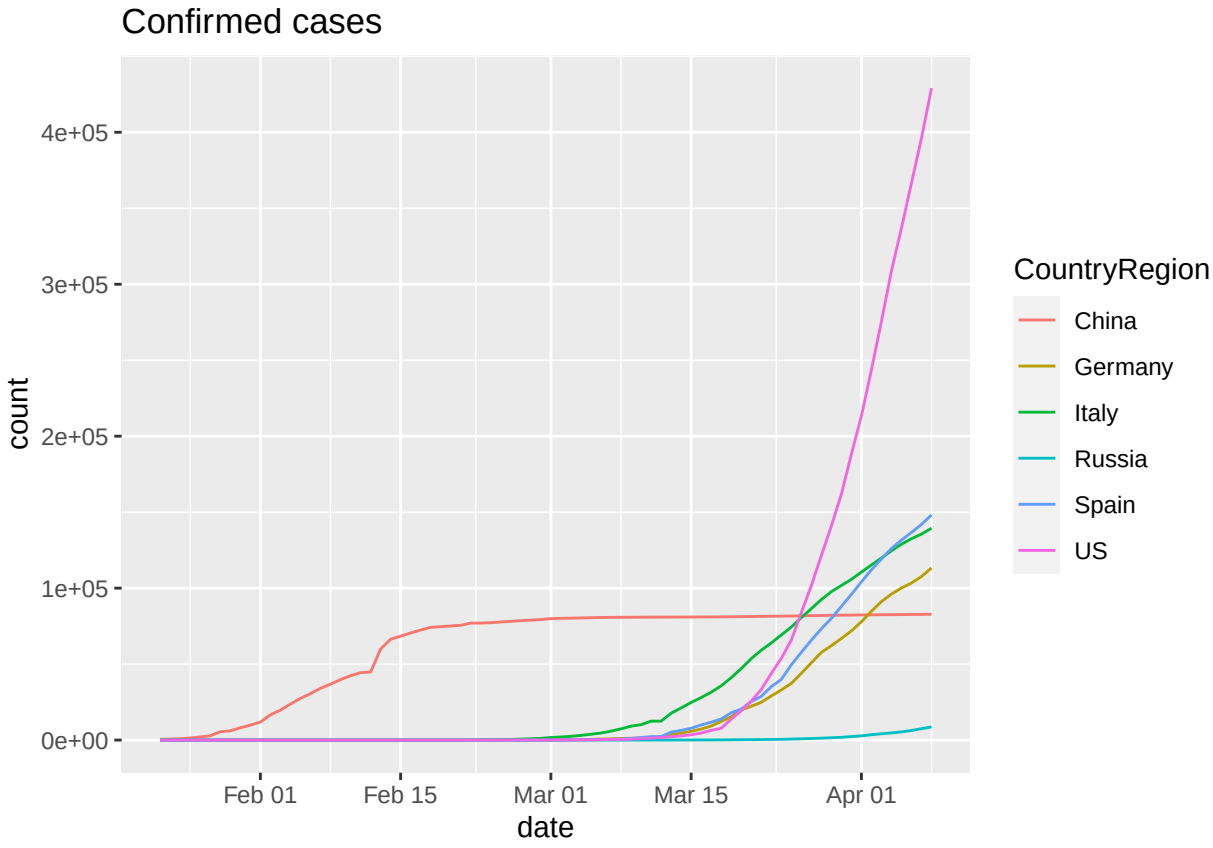
We have to play the same game of summing all values by country and date. Here, we filter the countries to be in a list of countries.

```
countries_of_interest = c('China','US','Italy','Spain','Germany','Russia')
library(ggplot2)
cc_ts = jhu %>%
    dplyr::group_by(CountryRegion,date) %>%
    dplyr::filter(CountryRegion %in% countries_of_interest & subset=='confirmed') %>%
    dplyr::summarize(count = sum(count))
head(cc_ts)
```

```
## # A tibble: 6 x 3
## # Groups:   CountryRegion [1]
##   CountryRegion date       count
##   <chr>         <date>     <dbl>
## 1 China         2020-01-22   548
## 2 China         2020-01-23   643
## 3 China         2020-01-24   920
## 4 China         2020-01-25  1406
## 5 China         2020-01-26  2075
## 6 China         2020-01-27  2877
```

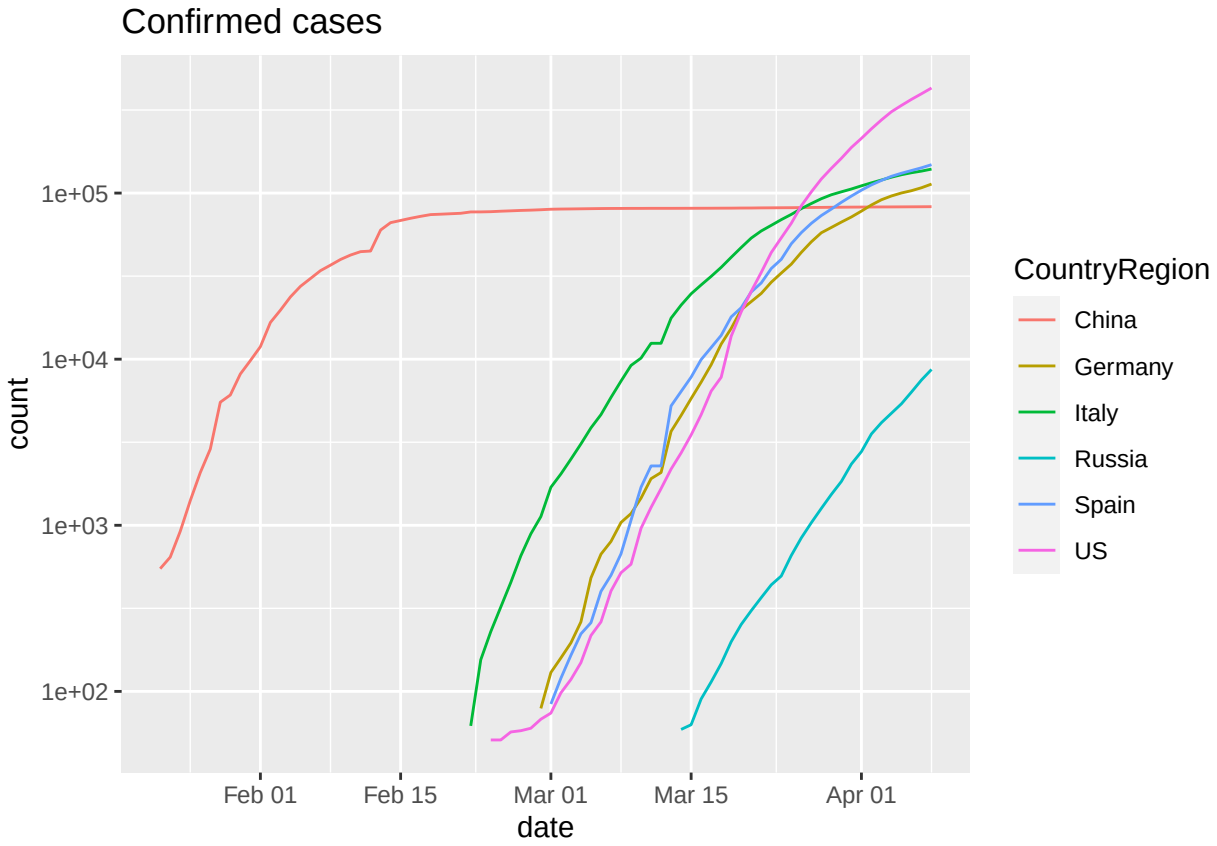
To make the plot, we use the ggplot2 grouping and coloring to provide curves for each country on the same axis.

```
ggplot(cc_ts,aes(x=date, y=count, group=CountryRegion)) +
    geom_line(aes(color=CountryRegion)) +
    ggtitle('Confirmed cases')
```



Changing to log scale can give a sense of the "exponentialness" of these data. Also, to remove zeros from the data (which cause problems when taking logs), we can filter data to include only values >=50. Note that ggplot2 will "do the right thing".

```
cc_ts %>%
    dplyr::filter(count>=50) %>%
    ggplot(aes(x=date, y=count, group=CountryRegion)) +
    geom_line(aes(color=CountryRegion)) +
    ggtitle('Confirmed cases') +
    scale_y_log10()
```



Consider the following questions based on the figure:

- What does the slope of the lines in this plot represent?
- What is the difference between China and other countries? What does this difference mean in terms of how the disease is spreading?
- What does each
- Pick an arbitrary level on the y-axis and look at the dates associated with each country's curve with respect to that level. What do differences along the x-axis tell us about where the countries are with respect to disease process?

6 The Replication rate, R_0

6.1 Learning goals and objectives

- Gain an intuitive understanding of R_0.
- Know that the value of R_0 determines how quickly a disease spreads or is eliminated.
- Name the three main drivers of R_0
- Learn to estimate R_0 from data on the number of infections over time occurring in a population.

6.2 Background

The replication rate, R_0 is a central value in understanding the rate at which a disease is spreading in a susceptible population.

6.2.1 What is R_0?

R_0 is pronounced "R naught." The R_0 value is an estimate of the average number of people who will be infected by one contagious person. It specifically applies to a population of people who are susceptible to the disease (have not been vaccinated and are not immune). If a disease has an R_0 of 18, for example, a contagious person will transmit it to an average of 18 other people, assuming that all people in the community are susceptible.

6.2.2 What do R_0 values mean?

The R_0 value of a disease is important to understanding the dynamics of disease spread. Depending on the R_0 value, a disease should follow one of three possible courses in the at-risk community.

- If R_0 is less than 1, each existing infection is spread on average to less than one additional person, leading to decline in the number of cases and eventual end to the spread.
- If R_0 equals 1, each existing infection causes one new infection, leading to stable infection numbers without increase or decrease with time, on average.
- If R_0 is more than 1, each existing infection leads to more than one infection, resulting in growth and potential for epidemic/pandemic conditions.

Importantly, the disease-specific R_0 value pplies when each member of the community is fully vulnerable to the disease with:

- no one vaccinated
- no one immune
- no way to control the spread of the disease

6.2.3 What variables contribute to R_0?

Three main factors impact the R_0 value of a disease:

- *Infectious period*: The time that an infected person can spread the disease varies from one disease to another. Additional factors such as age of the infected person may affect the period during which a person can infect others. A long period of infectiousness will contribute to a higher R_0 value.
- *Contact rate*: If a person who's infected with a contagious disease comes into contact with many people who aren't infected or vaccinated, the disease will spread more quickly. If that person remains at home, in a hospital, or otherwise quarantined while they're contagious, the disease will spread more slowly. A high contact rate will contribute to a higher R_0 value. The corollary, that lower contact rate, can reduce R_0 is the basis for flattening the curve through social distancing.
- *Mode of transmission*: Airborne illnesses tend to have a higher R_0 value than those spread through contact or through bodily fluids.

6.3 Simulated epidemic model

TODO: plot of incidence along with $R_0(t)$

Following code conveyed by John Mallery, we have the following approach for estimating R_0 using a single realization of an epidemic simulation.

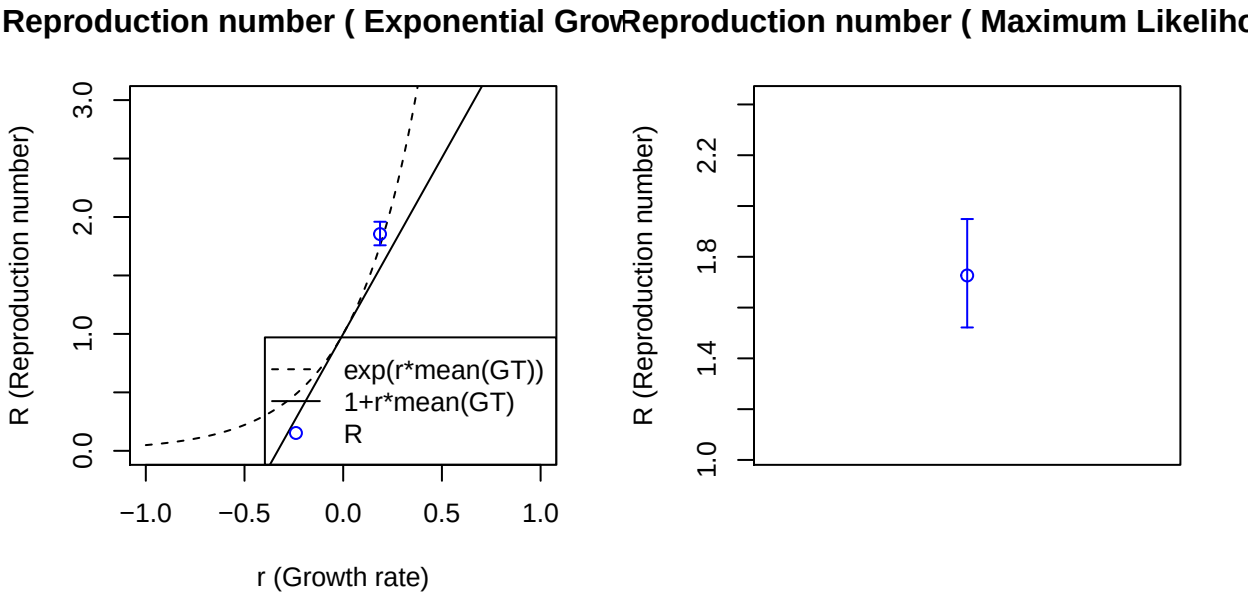
Note that there can be failures of `estimate.R` for certain inputs. We are working on that.

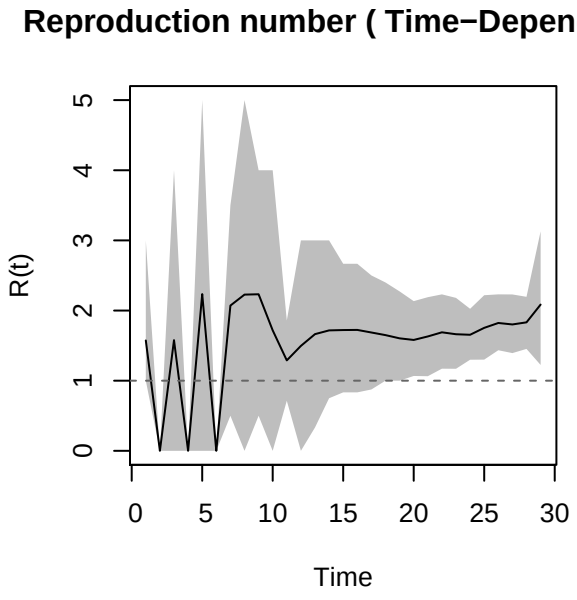
```
library(sars2pack)
library(R0)
library(lubridate)
```

```r
# Generating an epidemic with given parameters
mGT <- generation.time("gamma", c(3,1.5))
set.seed(5432)  # always initialize when simulating!
mEpid <- sim.epid(epid.nb=1, GT=mGT, epid.length=30,
     family="poisson", R0=1.67, peak.value=500)
mEpid <- mEpid[,1]
# Running estimations
est <- estimate.R(epid=mEpid, GT=mGT, methods=c("EG","ML","TD"), begin=1, end=30)
```

We modified the plotting function in *R0* which was calling `dev.new` too often. Use `plot2`.

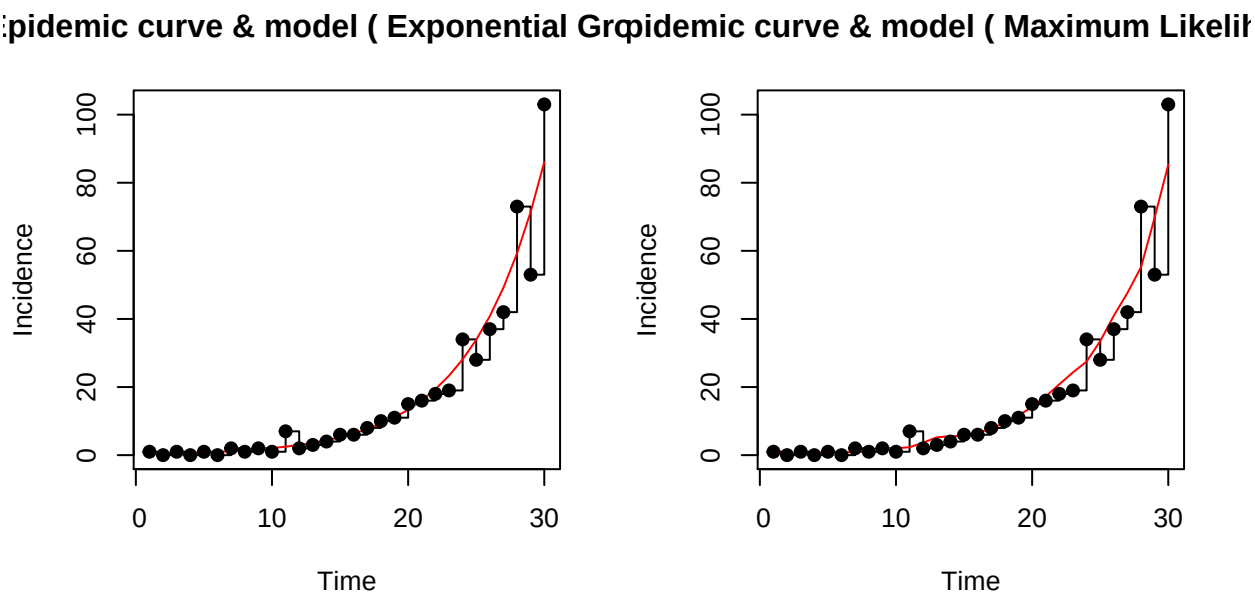
```r
par(mfrow=c(2,2))
sars2pack::plot2(est)
```

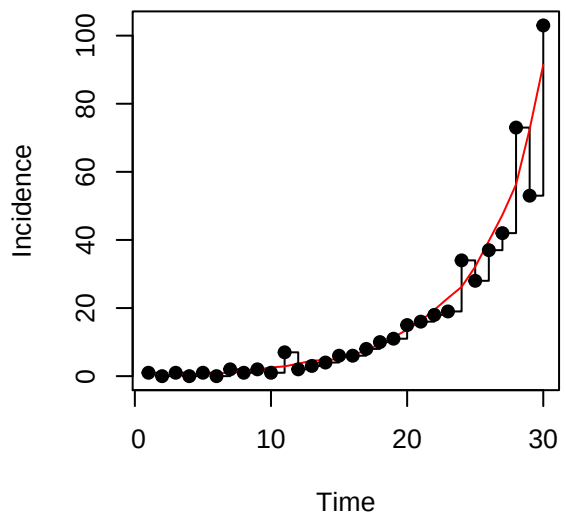




The `plotfit2` function is also useful. These fits look identical but they are not.

```
par(mfrow=c(2,2))
sars2pack::plotfit2(est)
```





6.4 Real data examples

```
library(dplyr)
library(magrittr)
```

Now we extract information from the time-series table and obtain estimates of R_0 under exponential growth.

6.4.1 Hubei Province

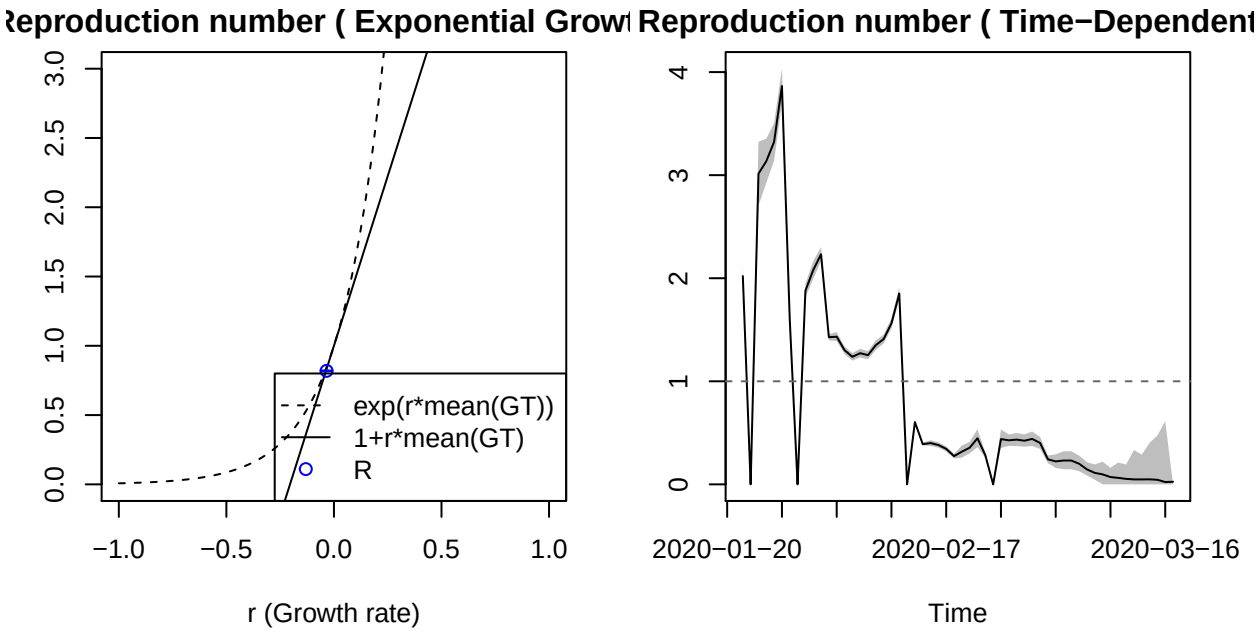
We are able to use exponential growth and time-dependent models with this data, using generation time model from a recent Annals of Internal Medicine paper.

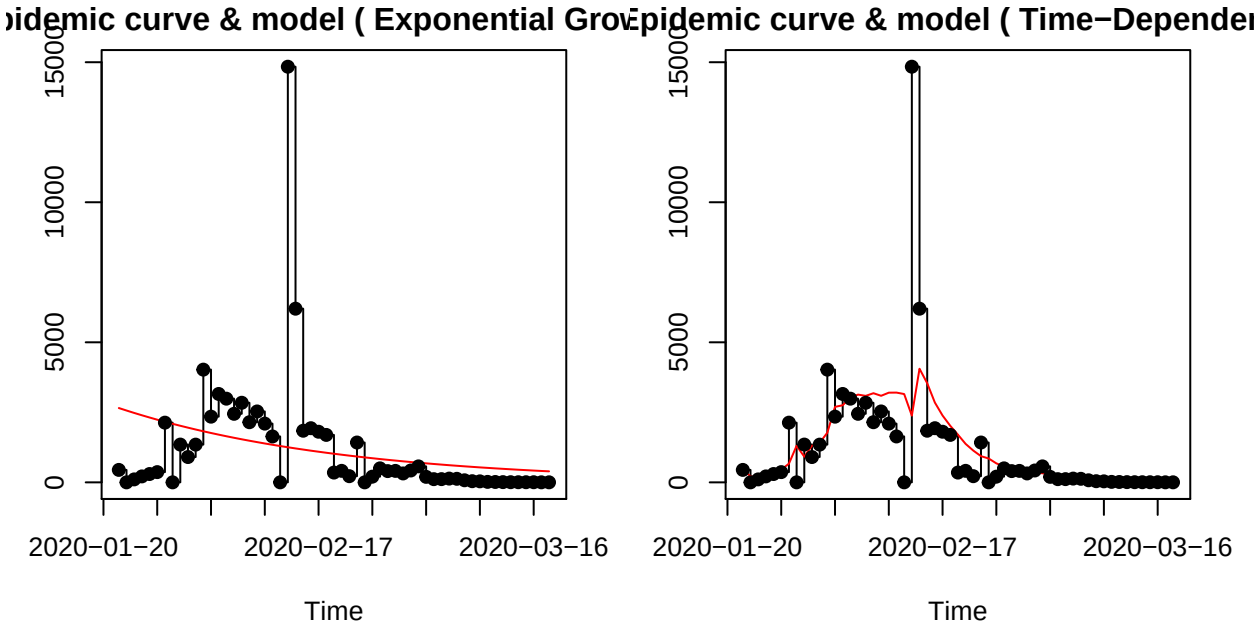
The incidence data probably need smoothing, and the time-dependent model has unreasonable fluctuations.

```
dates = lubridate::as_date(mdy(names(mar19df)[-c(1:4)]))
hubdat = as.numeric(get_series(province="Hubei", country="China",
    dataset=sars2pack::mar19df))
names(hubdat) = dates
mGT <- generation.time("gamma", c(5.8, 0.95)) # from DOI 10.7326/M20-0504
mGT <- generation.time("gamma", c(3.96, 4.75)) # from DOI 10.7326/M20-0504
hubdat.filt = trim_leading_values(c(hubdat[1], diff(hubdat)))
est.EG <- estimate.R(epid=hubdat.filt, GT=mGT,
    methods=c("EG", "TD"), begin=1L, end=as.integer(length(hubdat.filt)))
est.EG
```

```
## Reproduction number estimate using  Exponential Growth  method.
## R :  0.8190473[ 0.8164334 , 0.821658 ]
##
## Reproduction number estimate using  Time-Dependent  method.
## 2.020789 0 3.0142 3.134995 3.32356 3.865543 1.596743 0 1.878637 2.079345 ...
```

```
par(mfrow=c(2,2), mar=c(5,3,2,2))
plot2(est.EG)
plotfit2(est.EG)
```





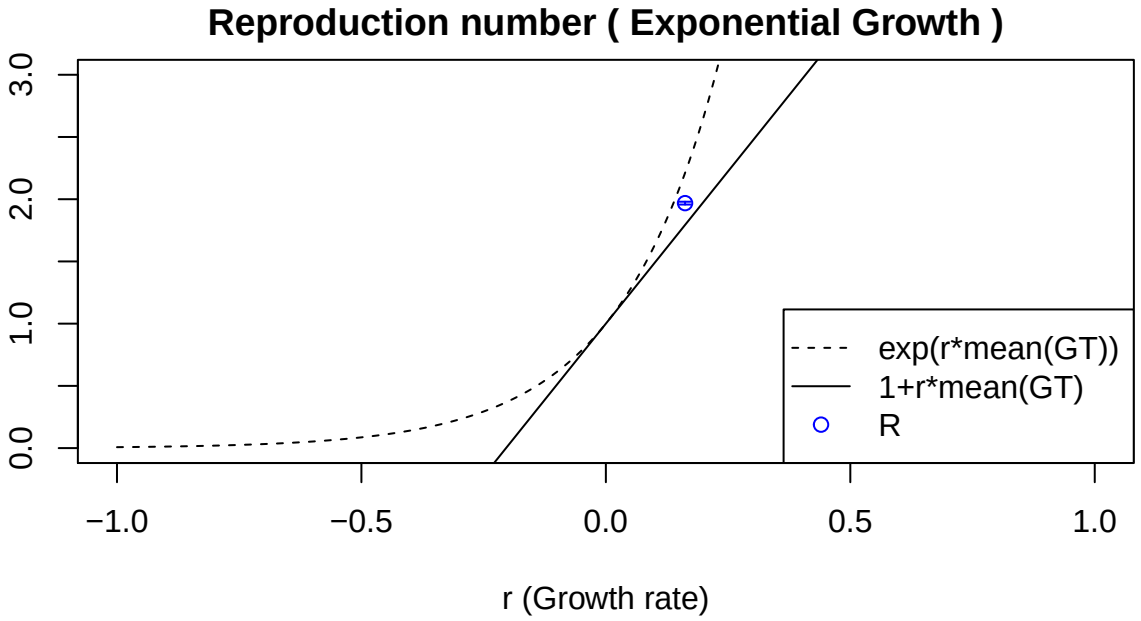
6.4.2 Italy

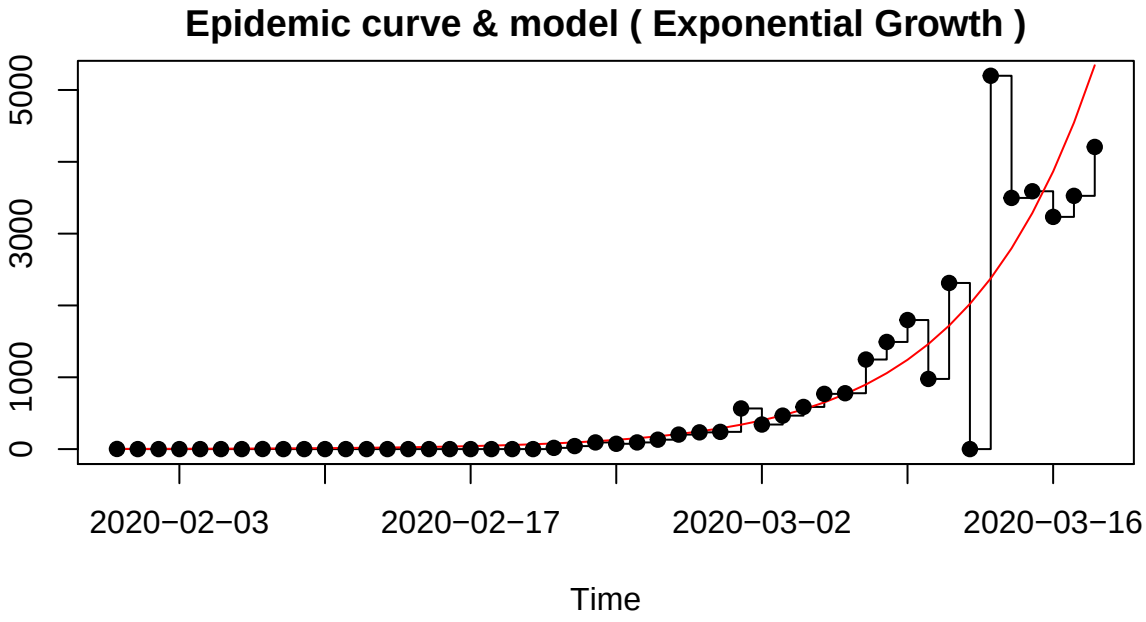
For Italy, only the EG model seems to work, with the Annals of Internal Medicine generation time model. It fits the data reasonably well, but the data seems to include a reporting gap.

```
itdat = as.numeric(get_series(province="", country="Italy",
    dataset=sars2pack::mar19df))
names(itdat) = dates
itdat.filt = trim_leading_values(c(itdat[1], diff(itdat)))
est.EG <- estimate.R(epid=itdat.filt, GT=mGT,
    methods=c("EG"), begin=1L, end=as.integer(length(itdat.filt)))
est.EG
```

```
## Reproduction number estimate using  Exponential Growth  method.
## R :  1.968466[ 1.957161 , 1.979874 ]
```

```
par(mfrow=c(2,1), mar=c(5,3,2,2))
plot2(est.EG, main="Italy")
plotfit2(est.EG, main="Italy")
```





6.4.3 New York City

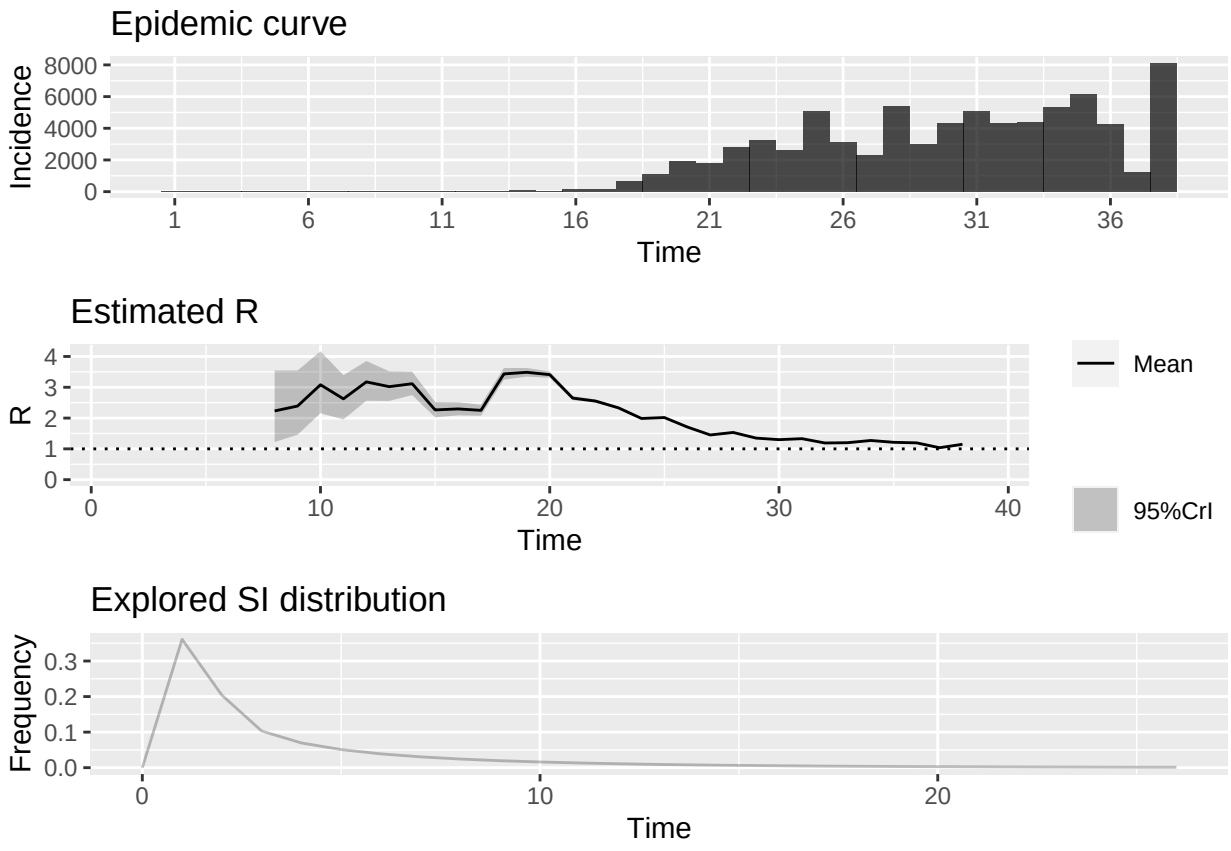
TODO: plot of incidence along with $R_0(t)$

```
nyt = nytimes_county_data() %>%
    dplyr::filter(county=='New York City' & subset=='confirmed') %>%
```

```
    dplyr::arrange(date)
nytdat = nyt$count
# do we need to chop zeros off? Seems like not.
nytdat.filt = c(nytdat[1], diff(nytdat))
est <- estimate.R(epid=nytdat.filt, GT=mGT,
                  methods=c("EG","TD","ML"), begin=1L, end=as.integer(length(nytdat.filt)))
```

We can also use the package *EpiEstim* to perform time-dependent R_0 calculations.

```
library(EpiEstim)
epiestim = EpiEstim::estimate_R(nytdat.filt, method = "parametric_si",
                                config = EpiEstim::make_config(list(
                                    mean_si = 3.96, std_si = 4.75)))
invisible(plot(epiestim))
```



7 European datasets

```
library(knitr)
knitr::opts_chunk$set(message=FALSE)
```

```
library(sars2pack)
library(eurostat)
library(dplyr)
library(tmap)
library(sf)
```

```
eucov = eu_data_cache_data()
```

```
geodata = get_eurostat_geospatial(output_class = "sf",
                                  resolution = "60",
                                  nuts_level = 2,
                                  year=2016)
## The names here need to be initialized to NULL
## for the join below to working
## See https://github.com/r-spatial/sf/issues/1177#issuecomment-541858742
head(names(geodata$geometry))
```

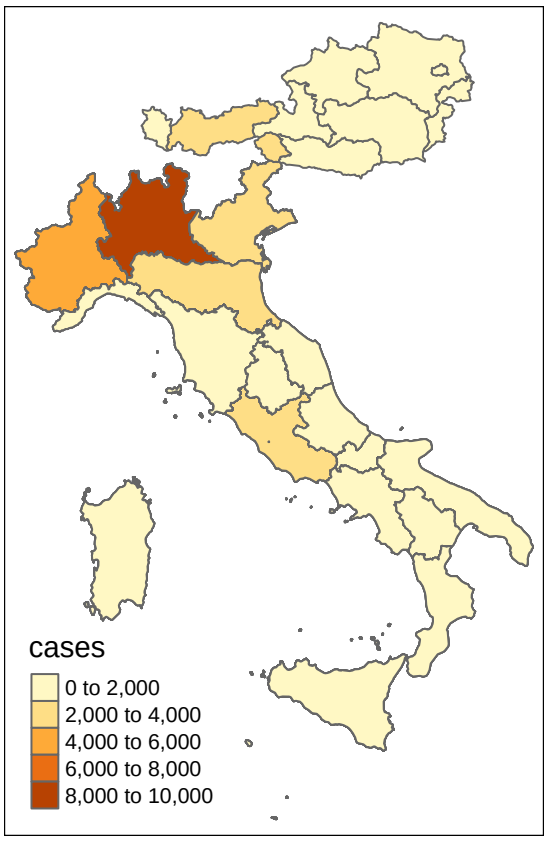
```
## [1] "0" "1" "2" "3" "4" "5"
```

```
names(geodata$geometry)=NULL
```

```
x = left_join(geodata, eucov, by=c('NUTS_NAME'='nuts_2'))
```

```
tmap::tm_shape(geodata) + tm_polygons()
```

```
x %>% filter(date=='2020-04-01') %>% tmap::tm_shape() + tm_polygons(col='cases')
```



8 Epicurve Datasets

```
library(sars2pack)
library(DT)
library(dplyr)
```

8.1 Worldwide datasets

```
jhu = jhu_data()
DT::datatable(jhu[sample(1:nrow(jhu),500),] %>%
              dplyr::arrange(CountryRegion, date),
              extensions = 'Scroller', options = list(scrollX=TRUE))
```

Show 10 entries Search:

	ProvinceState	CountryRegion	Lat	Long	date	count	subset
1		Afghanistan	33	65	2020-03-05	0	recovered
2		Albania	41.1533	20.1683	2020-01-26	0	recovered
3		Albania	41.1533	20.1683	2020-02-11	0	deaths
4		Albania	41.1533	20.1683	2020-02-20	0	recovered
5		Albania	41.1533	20.1683	2020-02-21	0	deaths
6		Albania	41.1533	20.1683	2020-02-28	0	recovered
7		Algeria	28.0339	1.6596	2020-02-02	0	recovered
8		Algeria	28.0339	1.6596	2020-03-09	20	confirmed
9		Algeria	28.0339	1.6596	2020-03-19	87	confirmed
10		Algeria	28.0339	1.6596	2020-04-03	62	recovered

Showing 1 to 10 of 500 entries Previous 1 2 3 4 5 … 50 Next

8.2 European datasets

```
eucov = eu_data_cache_data()
DT::datatable(eucov[sample(1:nrow(eucov),500),] %>%
                dplyr::arrange(country, date),
                extensions = 'Scroller', options = list(scrollX=TRUE))
```

Show 10 entries Search:

	country	nuts_2	cases	recovered	deaths	hospitalized	intensive_care	datetime	date
1	AT	Kärnten	1					2020-03-06T08:00:00Z	2020-03-06
2	AT	Burgenland	4					2020-03-07T08:00:00Z	2020-03-07
3	AT	Vorarlberg	1					2020-03-08T15:00:00Z	2020-03-08
4	AT	Kärnten	1					2020-03-09T15:00:00Z	2020-03-09
5	AT	Burgenland	4					2020-03-11T08:00:00Z	2020-03-11
6	AT	Wien	50					2020-03-11T08:00:00Z	2020-03-11
7	AT	Tirol	109					2020-03-12T15:00:00Z	2020-03-12
8	AT	Burgenland	6	0	0			2020-03-13T08:00:00Z	2020-03-13
9	AT	Salzburg	39	0	0			2020-03-15T15:00:00Z	2020-03-15
10	AT	Wien	180	5	0			2020-03-18T08:00:00Z	2020-03-18

Showing 1 to 10 of 500 entries Previous 1 2 3 4 5 … 50 Next

8.3 United States datasets

8.3.1 USAFacts

```
usa_facts = usa_facts_data()
DT::datatable(usa_facts[sample(1:nrow(usa_facts),500),] %>%
              dplyr::arrange(state, county, date, subset),
              extensions = 'Scroller', options = list(scrollX=TRUE))
```

Show 10 entries Search:

	county_fips	county	state	state_fips	subset	date	count
1	2050	Bethel Census Area	AK	2	deaths	2020-03-17	0
2	2195	Petersburg Census Area	AK	2	deaths	2020-02-13	0
3	0	Statewide Unallocated	AK	2	deaths	2020-03-06	0
4	1017	Chambers County	AL	1	deaths	2020-01-27	0
5	1035	Conecuh County	AL	1	deaths	2020-02-11	0
6	1057	Fayette County	AL	1	confirmed	2020-01-30	0
7	1087	Macon County	AL	1	deaths	2020-03-22	0
8	1095	Marshall County	AL	1	confirmed	2020-02-27	0
9	1097	Mobile County	AL	1	deaths	2020-02-03	0
10	1097	Mobile County	AL	1	deaths	2020-02-18	0

Showing 1 to 10 of 500 entries Previous 1 2 3 4 5 … 50 Next

8.3.2 New York Times County

```
nytimes = nytimes_county_data()
DT::datatable(nytimes[sample(1:nrow(nytimes),500),] %>%
              dplyr::arrange(state, county, date, subset),
              extensions = 'Scroller', options = list(scrollX=TRUE))
```

Show 10 entries Search:

	date	county	state	fips	count	subset
1	2020-03-23	Chambers	Alabama	01017	2	confirmed
2	2020-04-05	Clarke	Alabama	01025	0	deaths
3	2020-04-07	Crenshaw	Alabama	01041	0	deaths
4	2020-04-03	Dale	Alabama	01045	1	confirmed
5	2020-03-19	Elmore	Alabama	01051	5	confirmed
6	2020-03-31	Madison	Alabama	01089	100	confirmed
7	2020-03-28	Marshall	Alabama	01095	0	deaths
8	2020-04-06	Pike	Alabama	01109	14	confirmed
9	2020-03-27	Russell	Alabama	01113	1	confirmed
10	2020-04-07	Tallapoosa	Alabama	01123	33	confirmed

Showing 1 to 10 of 500 entries Previous 1 2 3 4 5 … 50 Next

8.3.3 New York Times State

```
nytimes = nytimes_state_data()
DT::datatable(nytimes[sample(1:nrow(nytimes),500),] %>%
              dplyr::arrange(state, date, subset),
              extensions = 'Scroller', options = list(scrollX=TRUE))
```

Show 10 entries Search:

	date	state	fips	count	subset
1	2020-03-19	Alabama	00001	0	deaths
2	2020-03-21	Alabama	00001	131	confirmed
3	2020-03-24	Alabama	00001	242	confirmed
4	2020-03-30	Alabama	00001	11	deaths
5	2020-04-01	Alabama	00001	28	deaths
6	2020-03-12	Alaska	00002	0	deaths
7	2020-03-14	Alaska	00002	0	deaths
8	2020-03-15	Alaska	00002	1	confirmed
9	2020-03-15	Alaska	00002	0	deaths
10	2020-03-17	Alaska	00002	6	confirmed

Showing 1 to 10 of 500 entries Previous 1 2 3 4 5 … 50 Next

9 Map visualizations

```
library(tmap)
library(dplyr)
library(sars2pack)
library(htmltools)
library(htmlwidgets)
```

```
ejhu = enriched_jhu_data()
```

```
glimpse(ejhu)
```

```
## Rows: 60,450
## Columns: 20
## $ name           <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghan...
## $ topLevelDomain <list> [".af", ".af", ".af", ".af", ".af", ".af", ".af", "...
## $ alpha2Code     <chr> "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF"...
## $ alpha3Code     <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AF...
## $ capital        <chr> "Kabul", "Kabul", "Kabul", "Kabul", "Kabul", "Kabul"...
## $ region         <chr> "Asia", "Asia", "Asia", "Asia", "Asia", "Asia", "Asi...
## $ subregion      <chr> "Southern Asia", "Southern Asia", "Southern Asia", "...
## $ population     <int> 27657145, 27657145, 27657145, 27657145, 27657145, 27...
## $ area           <dbl> 652230, 652230, 652230, 652230, 652230, 652230, 6522...
## $ gini           <dbl> 27.8, 27.8, 27.8, 27.8, 27.8, 27.8, 27.8, 27.8, 27.8...
## $ borders        <list> [<"IRN", "PAK", "TKM", "UZB", "TJK", "CHN">, <"IRN"...
## $ numericCode    <chr> "004", "004", "004", "004", "004", "004", "004", "00...
## $ cioc           <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AF...
```

```
## $ ProvinceState  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ CountryRegion  <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghan...
## $ Lat            <dbl> 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, ...
## $ Long           <dbl> 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, ...
## $ date           <date> 2020-01-22, 2020-01-23, 2020-01-24, 2020-01-25, 202...
## $ count          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ subset         <chr> "confirmed", "confirmed", "confirmed", "confirmed", ...
```

We need a description of the regions of the world.

```
data(World)
```

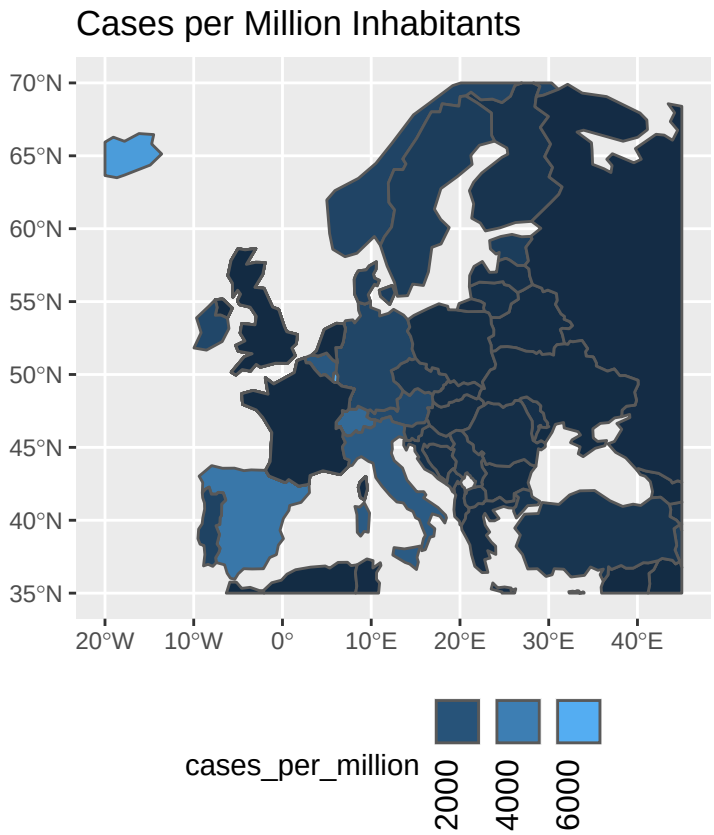
The `World` object has a column, `geometry`, that describes the shape of each country in the `World` dataset. Join the `ejhu` data.frame with the `World` data using `dplyr` join as normal.

```
geo_ejhu = World %>%
    dplyr::left_join(ejhu, by = c('iso_a3' = 'alpha3Code'))
```

```
w2 = geo_ejhu %>%
    filter(!is.na(date) & subset=='confirmed') %>%
    group_by(iso_a3) %>%
    filter(date==max(date)) %>%
    mutate(cases_per_million = 1000000*count/pop_est)
```

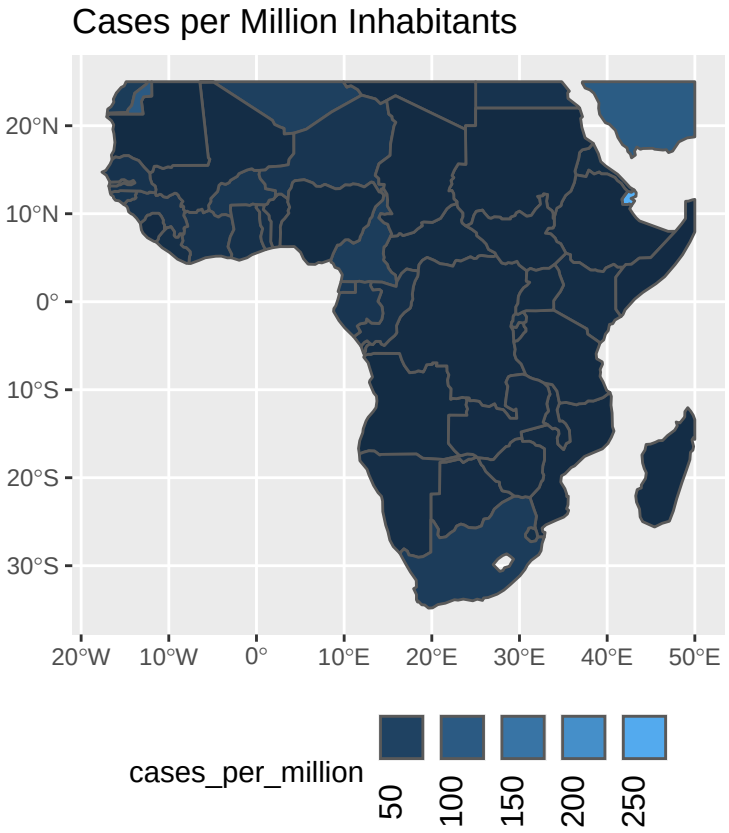
The R package `ggplot2` has geospatial plotting capabilities built in for geospatial `simple features (sf)` data types. In this first plot, we focus in on Europe.

```
library(ggplot2)
# transform to lat/long coordinates
st_transform(w2, crs=4326) %>%
# Crop to europe (rough, by hand)
    st_crop(xmin=-20,xmax=45,ymin=35,ymax=70) %>%
ggplot() +
    geom_sf(aes(fill=cases_per_million)) +
    scale_fill_continuous(
        guide=guide_legend(label.theme = element_text(angle = 90),
                           label.position='bottom')
        ) +
    labs(title='Cases per Million Inhabitants') +
    theme(legend.position='bottom')
```



Another plot, but now for Africa.

```
library(ggplot2)
# transform to lat/long coordinates
st_transform(w2, crs=4326) %>%
# Crop to europe (rough, by hand)
    st_crop(xmin=-20,xmax=50,ymin=-60,ymax=25) %>%
ggplot() +
    geom_sf(aes(fill=cases_per_million)) +
    scale_fill_continuous(
        guide=guide_legend(label.theme = element_text(angle = 90),
                           label.position='bottom')
        ) +
    labs(title='Cases per Million Inhabitants') +
    theme(legend.position='bottom')
```



9.1 Interactive maps

The following will not produce a plot when run non-interactively. However, pasting this into your R session will result in an interactive plot with multiple "layers" that you can choose to visualize different quantitative variables on the map. Zooming also works as expected.

```
tmap_mode('view')
## geo_ejhu %>%
##     filter(!is.na(date) & subset=='confirmed') %>%
##     group_by(iso_a3) %>%
##     filter(date==max(date)) %>%
##     tm_shape() +
##         tm_polygons(col='count')
w2 = geo_ejhu %>%
    filter(!is.na(date) & subset=='confirmed') %>%
    group_by(iso_a3) %>%
    filter(date==max(date)) %>%
    mutate(cases_per_million = 1000000*count/pop_est) %>%
    filter(region == 'Africa')
m = tm_shape(w2,id='name.x', name=c('cases_per_million'),popup=c('pop_est')) +
    tm_polygons(c('Cases Per Million' = 'cases_per_million','Cases' = 'count',"Well-being index"='well_
                selected='cases_per_million',
                border.alpha = 0.5,
                alpha=0.6,
                popup.vars=c('Cases Per Million'='cases_per_million',
                             'Confirmed Cases'  ='count',
                             'Population'       ='pop_est',
```

```
                            'gini'              ='gini',
                            'Life Expectancy'   ='life_exp')) +
    tm_facets(as.layers = TRUE)
tmap_save(m, filename='abc.html')
```

9.2 United States

```
library(ggplot2)
library(tigris)
library(tidycensus)
library(plotly)
library(sf)
```

```
county_geom = tidycensus::county_laea
nyt_counties = nytimes_county_data()
full_map = county_geom %>%
    left_join(
        nyt_counties %>%
        group_by(fips) %>%
        filter(date==max(date) & count>0 & subset=='confirmed'), by=c('GEOID'='fips')) %>%
    mutate(mid=sf::st_centroid(geometry))
z = ggplot(full_map, aes(label=county)) +
    geom_sf(aes(geometry=geometry),color='grey85') +
    geom_sf(aes(geometry=mid, size=count, color=count), alpha=0.5, show.legend = "point") +
    scale_color_gradient2(midpoint=5500, low="lightblue", mid="orange",high="red", space ="Lab" ) +
    scale_size(range=c(1,10))
ggplotly(z)
```

A static plot as a png:

```
z
```

Alternatively, produce a PDF of the same plot.

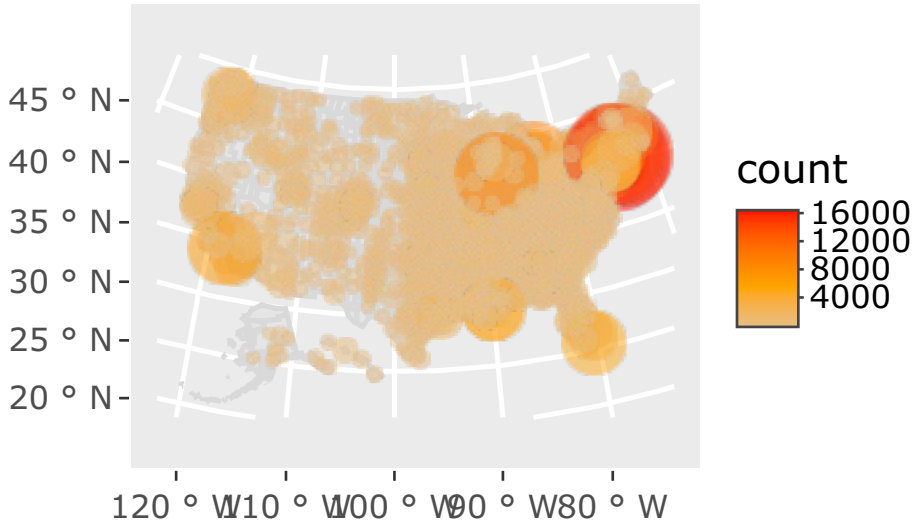


Figure 1: United States confirmed cases by County with interactive plotly library. Click and drag to zoom in to a region of interest.

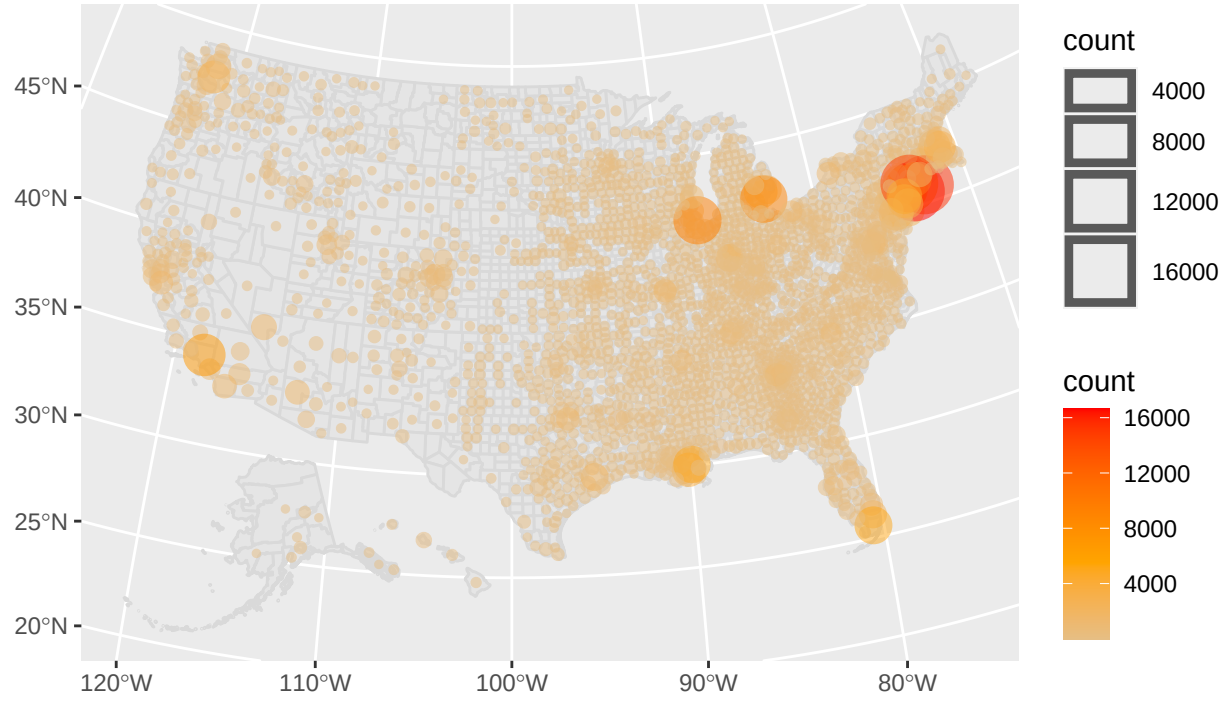


Figure 2: United States confirmed cases by County as a static graphic.

```r
pdf('us_county_numbers.pdf', width=11, height=8)
print(z)
dev.off()
```

```
## pdf
##   2
```