

Understanding the COVID-19 pandemic

A data-driven approach using the sars2pack R package

Vincent Carey Charles Morefield John Mallory Sean Davis

2020-04-02

Contents

I	Introduction	1
1	Motivation	2
2	Quick start	2
2.1	Installation	2
2.2	COVID-19 resources in this package	2
2.3	Additional resources described in this vignette	2
3	Epidemic time series data	3
3.1	Access data	3
4	Use cases	5
4.1	Basic data exploration	5
4.2	Visualize time series data	8
5	Modeling Replication rate	11
5.1	Background	11
5.2	Simulated epidemic model	12
5.3	Real data examples	15
6	European datasets	19
7	Epicurves	21
7.1	Worldwide datasets	22
7.2	European datasets	22
7.3	United States datasets	23
8	Map visualizations	24
8.1	Interactive maps	27
	Origins of this package	28

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

```
## Warning: package 'BiocStyle' was built under R version 3.6.2
```

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: I would be more than happy to accept help with defining new data resources. Consider a pull request (or an issue for non-programmer types).

2.2.1 Epidemic time-series data

- JHU : global deaths, confirmed cases, and recovered time series data; *does not include fine-level United States data.* See `jhu_data()`.
- New York Times : United states state and county level deaths, confirmed cases time series. See `nytimes_county_data()` and `nytimes_state_data()`.
- USAFacts : Alternative United states state and county level deaths and confirmed cases time series

2.3 Additional resources described in this vignette

-

3 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.

3.1 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.

3.1.1 JHU Dataset

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

## [1] "tbl_df"     "tbl"        "data.frame"
dim(jhu)

## [1] 37152      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 Afghanistan    Afghanistan    33    65 2020-01-22     0 confirmed
## 2 Afghanistan    Afghanistan    33    65 2020-01-23     0 confirmed
## 3 Afghanistan    Afghanistan    33    65 2020-01-24     0 confirmed
```

3.1.2 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

```

3.1.3 NYTimes datasets

```

nytimes_state = nytimes_state_data()
class(nytimes_state)

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

dim(nytimes_state)

## [1] 3328      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] 52398      6

```

```

colnames(nytimes_county)

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

```

4 Use cases

4.1 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] 37152
```

- How many different countries are represented?

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

```
## [1] 181
```

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: <input type="text"/>					
ProvinceState	CountryRegion	Lat	Long	date	count	subset	
	US	37.0902	-95.7129	2020-04-02	243453	confirmed	
	Italy	43	12	2020-04-02	115242	confirmed	
	Spain	40	-4	2020-04-02	112065	confirmed	
	Germany	51	9	2020-04-02	84794	confirmed	
	France	46.2276	2.2137	2020-04-02	59105	confirmed	
	Iran	32	53	2020-04-02	50468	confirmed	
	United Kingdom	55.3781	-3.436	2020-04-02	33718	confirmed	
	Switzerland	46.8182	8.2275	2020-04-02	18827	confirmed	
	Turkey	38.9637	35.2433	2020-04-02	18135	confirmed	
	Belgium	50.8333	4	2020-04-02	15348	confirmed	

Showing 1 to 10 of 178 entries

Previous 1 2 3 4 5 ... 18 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.

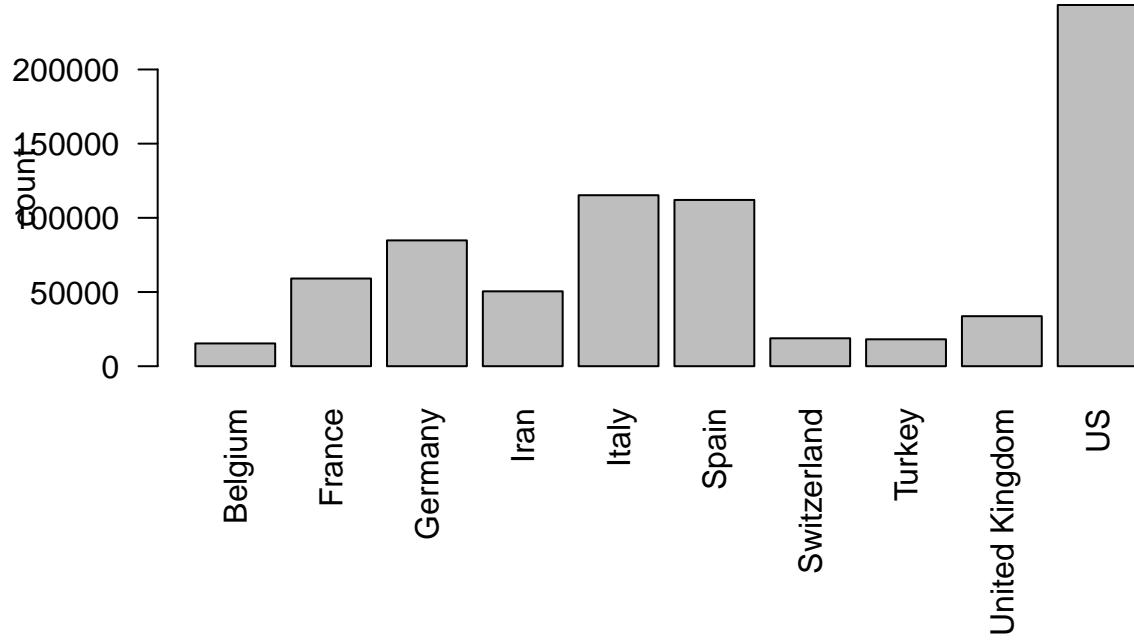
The data here could be usefully displayed as a graph as well.

```

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')

```

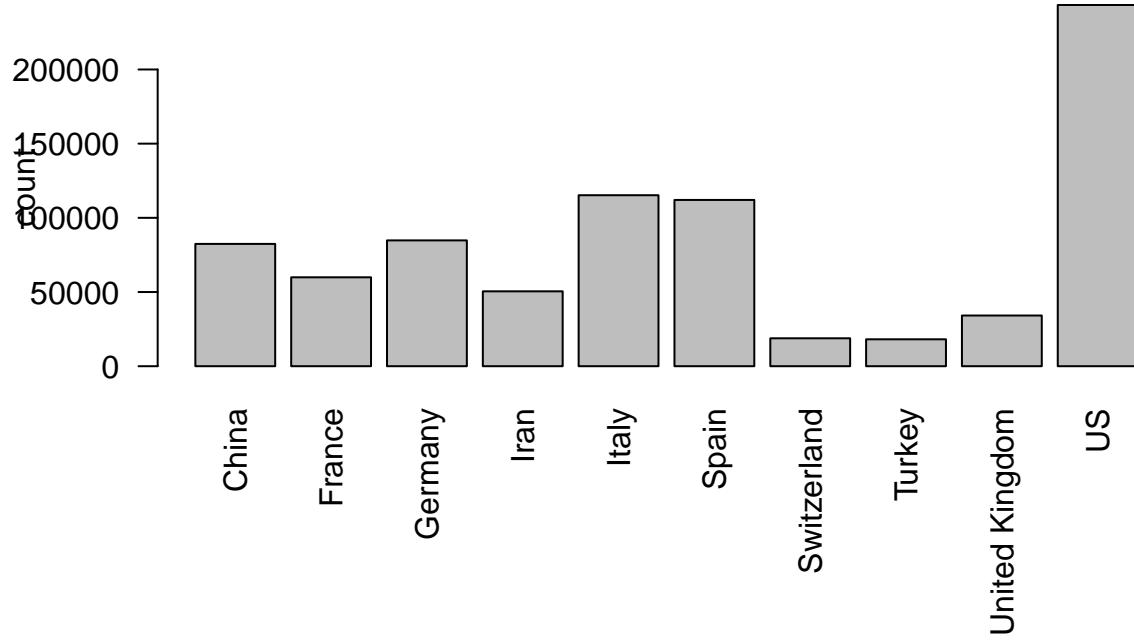
Confirmed cases, top 10 countries



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')
```

Confirmed cases, top 10 countries

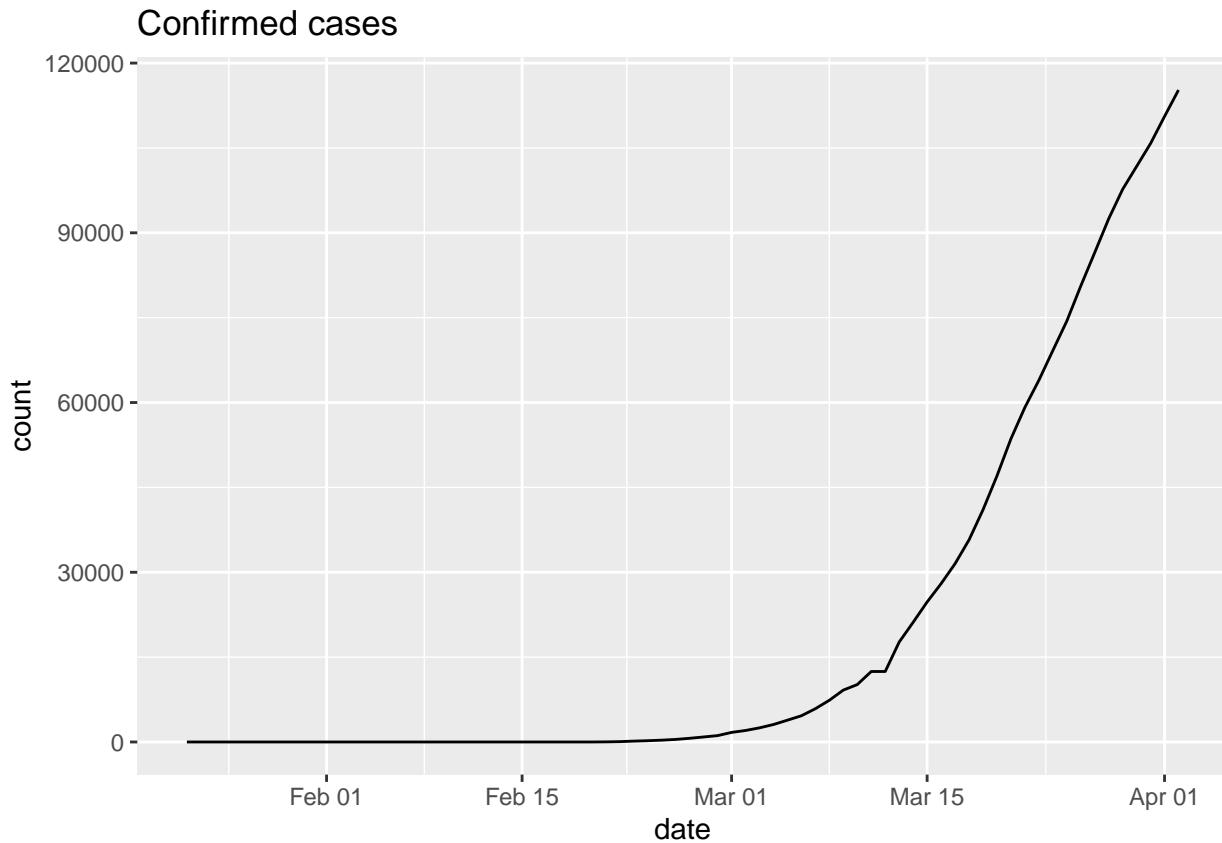


4.2 Visualize time series data

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?

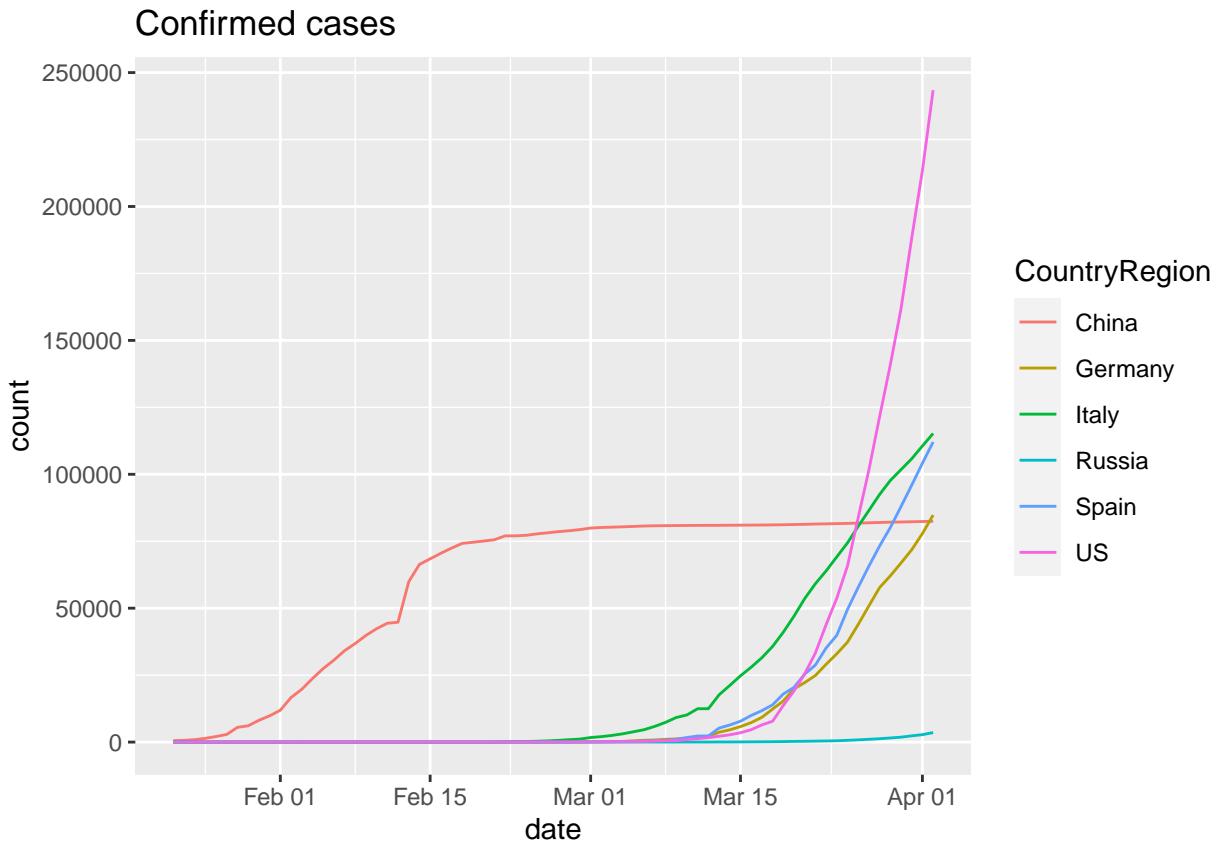
We have to play the same game of summing all values by country and date. Here, we filter the countries to be in a list of countries.

```
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
```

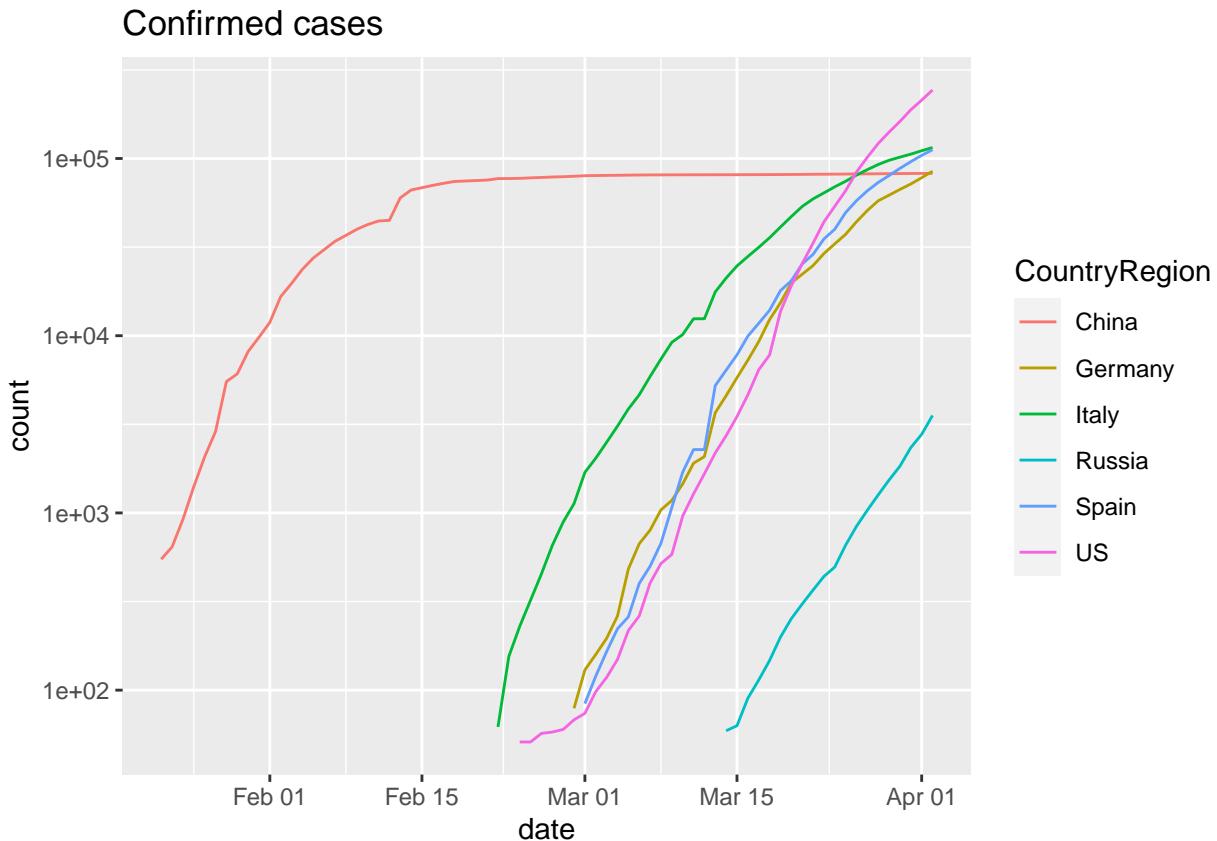
To make the plot, we use the ggplot2 grouping and coloring to provide curves for each country on the same axis.

```
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?

5 Modeling Replication rate

5.1 Background

TODO: PARAPHRASE!!!!

5.1.1 What is R₀?

R_0 is pronounced “R naught.” It’s a mathematical term that indicates how contagious an infectious disease is. It’s also referred to as the reproduction number. As an infection spreads to new people, it reproduces itself.

R_0 tells you the average number of people who will catch a disease from one contagious person. It specifically applies to a population of people who were previously free of infection and haven’t been vaccinated. If a disease has an R_0 of 18, a person who has the disease will transmit it to an average of 18 other people, as long as no one has been vaccinated against it or is already immune to it in their community.

5.1.2 What do R0 values mean?

Three possibilities exist for the potential spread or decline of a disease, depending on its R0 value:

- If R0 is less than 1, each existing infection causes less than one new infection. In this case, the disease will decline and eventually die out.
- If R0 equals 1, each existing infection causes one new infection. The disease will stay alive and stable, but there won't be an outbreak or an epidemic.
- If R0 is more than 1, each existing infection causes more than one new infection. The disease will spread between people, and there may be an outbreak or epidemic.

Importantly, a disease's R0 value only applies when everyone in a population is completely vulnerable to the disease. This means:

- no one has been vaccinated
- no one has had the disease before
- there's no way to control the spread of the disease

This combination of conditions is rare nowadays thanks to advances in medicine. Many diseases that were deadly in the past can now be contained and sometimes cured. For example, in 1918 there was a worldwide outbreak of the swine flu that killed 50 million people. According to a review article published in BMC Medicine, the R0 value of the 1918 pandemic was estimated to be between 1.4 and 2.8. But when the swine flu, or H1N1 virus, came back in 2009, its R0 value was between 1.4 and 1.6, report researchers in the journal Science. The existence of vaccines and antiviral drugs made the 2009 outbreak much less deadly.

5.1.3 How is the R0 of a disease calculated?

The following factors are taken into account to calculate the R0 of a disease:

- *Infectious period:* Some diseases are contagious for longer periods than others. For example, according to the Centers for Disease Control and Prevention, adults with the flu are typically contagious for up to eight days, while children can be contagious for up to two weeks. The longer the infectious period of a disease, the more likely an infected person is to spread the disease to other people. A long period of infectiousness will contribute to a higher R0 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 R0 value.
- *Mode of transmission:* The diseases that spread most quickly and easily are the ones that can travel through the air, such as the flu or measles. Physical contact with an infected person isn't necessary for the transmission of such conditions. You can catch the flu from breathing near someone who has the flu, even if you never touch them.

In contrast, diseases that are transmitted through bodily fluids, such as Ebola or HIV, aren't as easy to catch or spread. This is because you need to come into contact with infected blood, saliva, or other bodily fluids to contract them. Airborne illnesses tend to have a higher R0 value than those spread through contact.

5.2 Simulated epidemic model

Following code conveyed by John Mallory, we have the following approach for estimating R0 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(R0)
library(lubridate)

## 
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
## 
##     date

# 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)

## Waiting for profiling to be done...
## Warning in est.R0.TD(epid = c(1, 0, 1, 0, 1, 0, 2, 1, 2, 1, 7, 2, 3, 4, :
##   Simulations may take several minutes.

## Warning in est.R0.TD(epid = c(1, 0, 1, 0, 1, 0, 2, 1, 2, 1, 7, 2, 3, 4, :
##   Using initial incidence as initial number of cases.

```

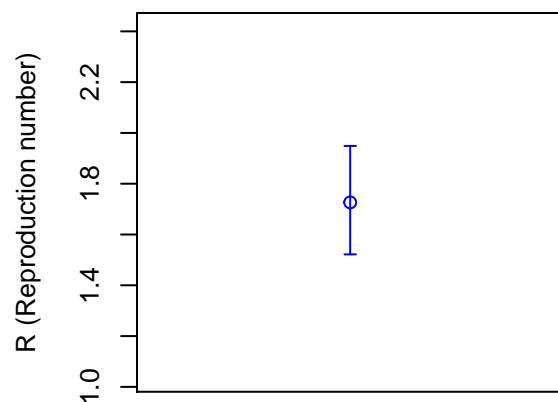
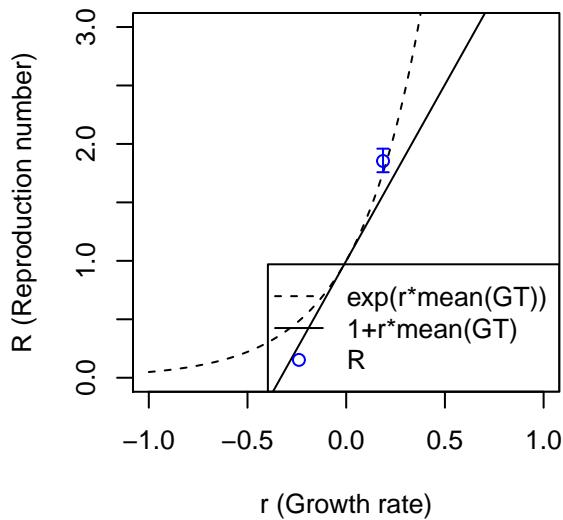
We modified the plotting function in *R0* which was calling `dev.new` too often. Use `plot2`.

```

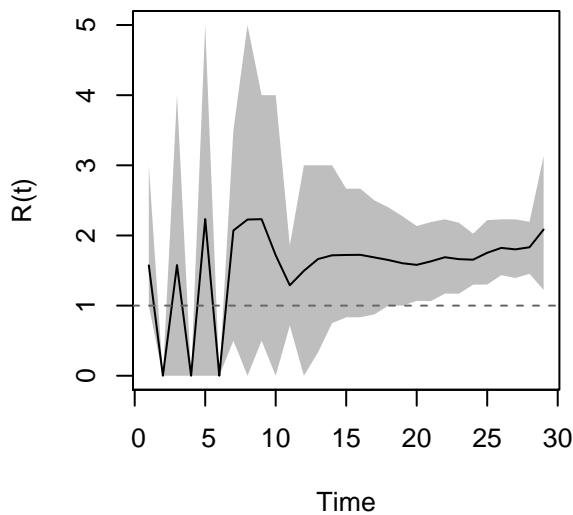
par(mfrow=c(2,2))
plot2(est)

```

Reproduction number (Exponential Growth) vs Reproduction number (Maximum Likelihood)



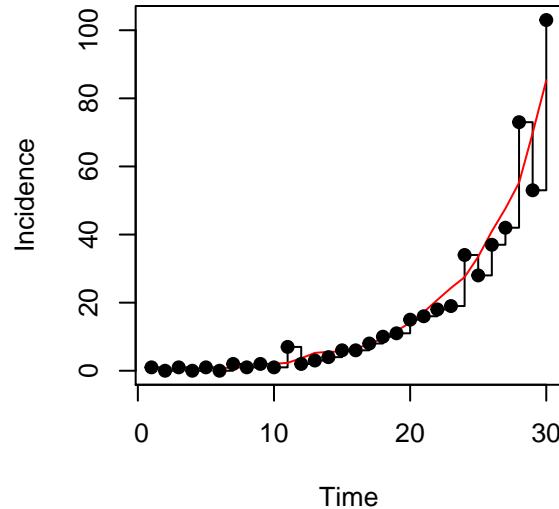
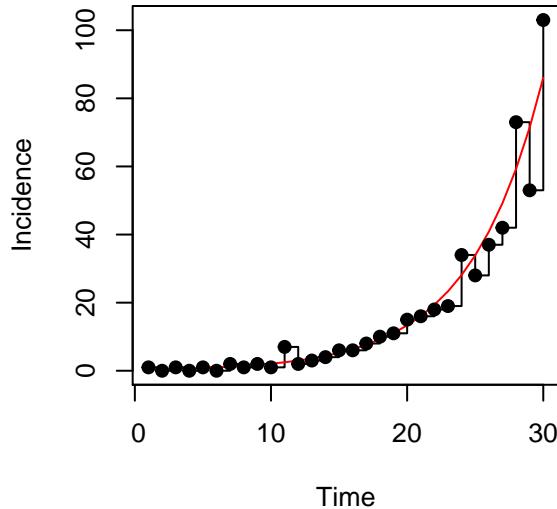
Reproduction number (Time-Depender)



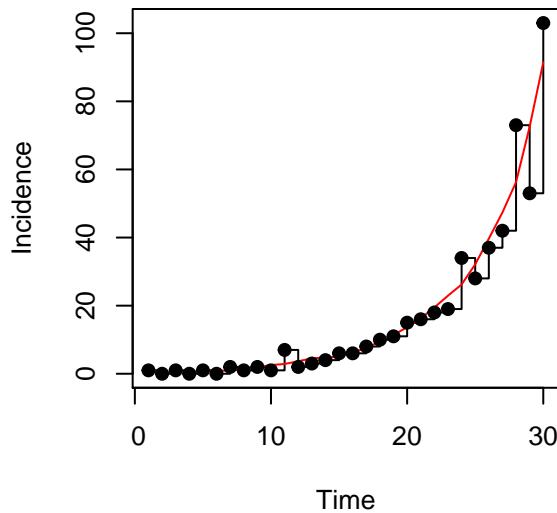
The `plotfit2` function is also useful. These fits look identical but they are not.

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

Epidemic curve & model (Exponential Growth)



Epidemic curve & model (Time-Dependent)



5.3 Real data examples

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

5.3.1 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)))

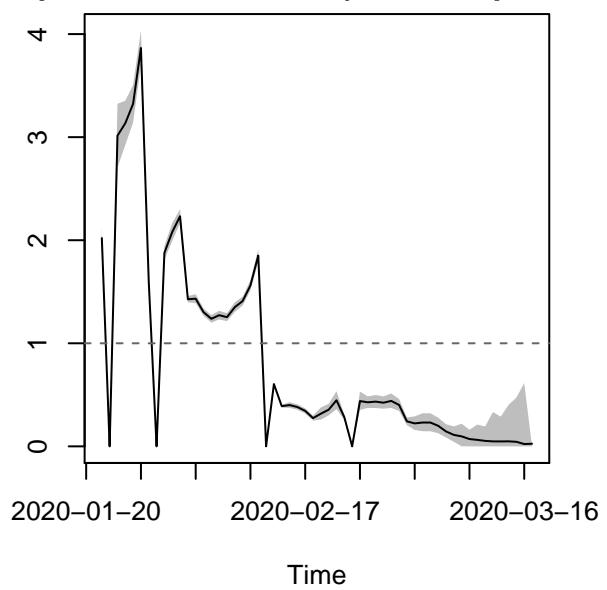
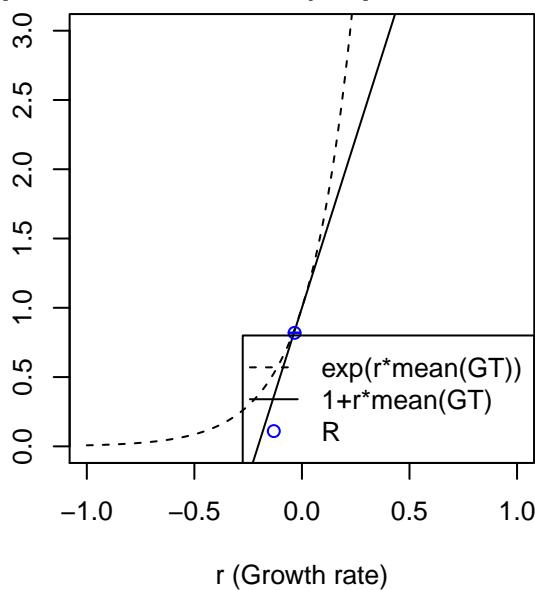
## Waiting for profiling to be done...
## Warning in est.R0.TD(epid = c(`2020-01-22` = 444, `2020-01-23` = 0, `2020-01-24` =
## = 105, : Simulations may take several minutes.
## Warning in est.R0.TD(epid = c(`2020-01-22` = 444, `2020-01-23` = 0, `2020-01-24` =
## = 105, : Using initial incidence as initial number of cases.
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 ...

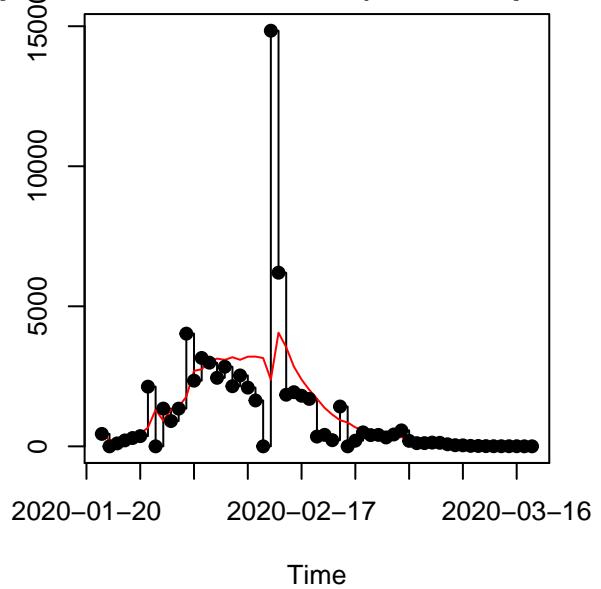
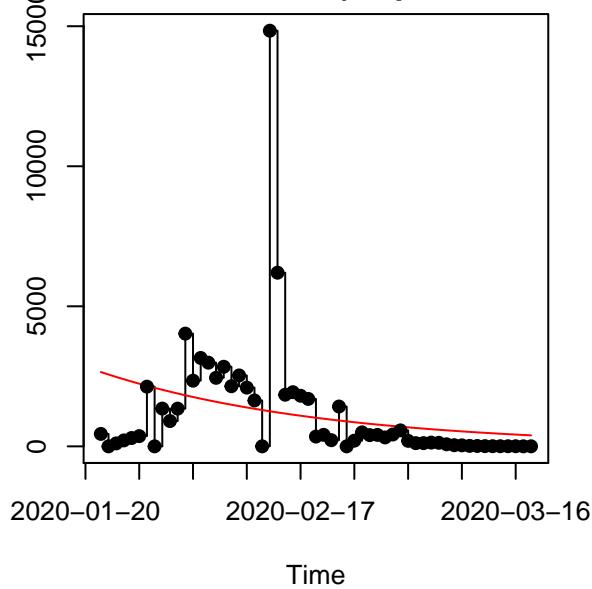
par(mfrow=c(2,2), mar=c(5,3,2,2))
plot2(est.EG)
plotfit2(est.EG)

```

Reproduction number (Exponential Growth)



Epidemic curve & model (Exponential Growth)



5.3.2 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="",
  dataset=sars2pack::mar19df))
names(itdat) = dates
itdatfilt = trim_leading_values(c(itdat[1], diff(itdat)))
est.EG <- estimate.R(epid=itdatfilt, GT=mGT,
  methods=c("EG"), begin=1L, end=as.integer(length(itdatfilt)))
```

```

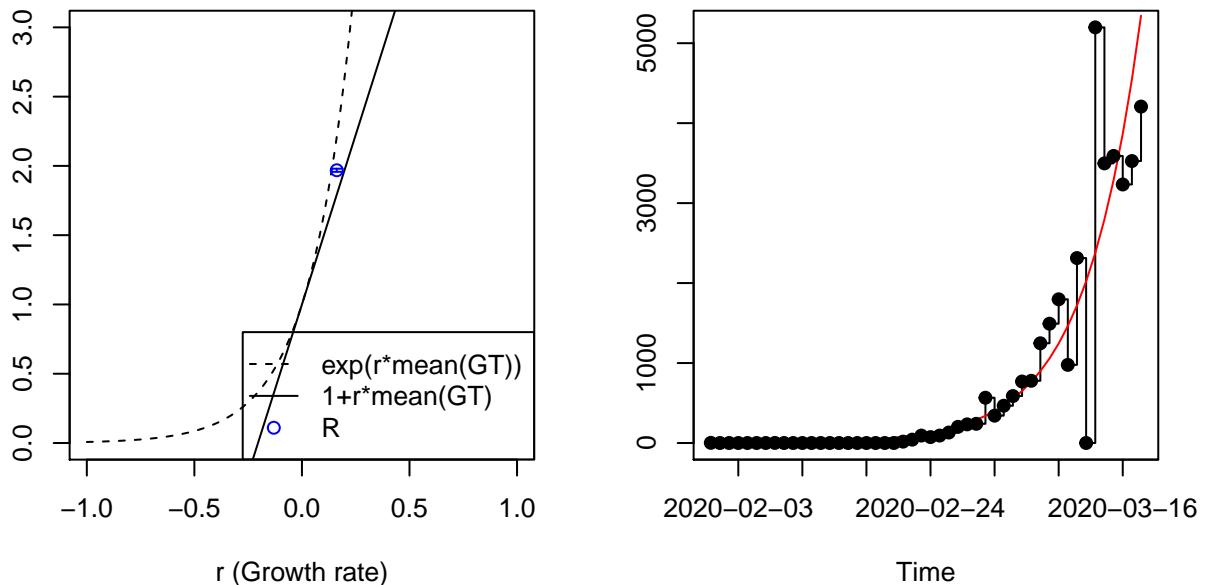
## Waiting for profiling to be done...
est.EG

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

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

```

Reproduction number (Exponential Growth) & epidemic curve & model (Exponential Growth)



5.3.3 NYC

```

nyt = nytimes_county_data() %>%
  dplyr::filter(county=='New York City' & subset=='confirmed') %>%
  dplyr::arrange(date)
nyt$count
# do we need to chop zeros off? Seems like not.
nyt.filt = c(nyt$date[1], diff(nyt$date))
est <- estimate.R(epid=nyt.filt, GT=mGT,
  methods=c("EG", "TD", "ML"), begin=1L, end=as.integer(length(nyt.filt)))

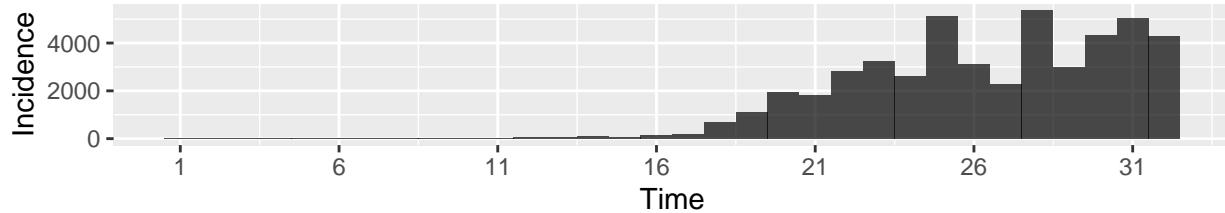
epiestim = EpiEstim::estimate_R(nyt.filt, method = "parametric_si",
  config = make_config(list(
    mean_si = 3.96, std_si = 4.75)))

## Default config will estimate R on weekly sliding windows.
## To change this change the t_start and t_end arguments.

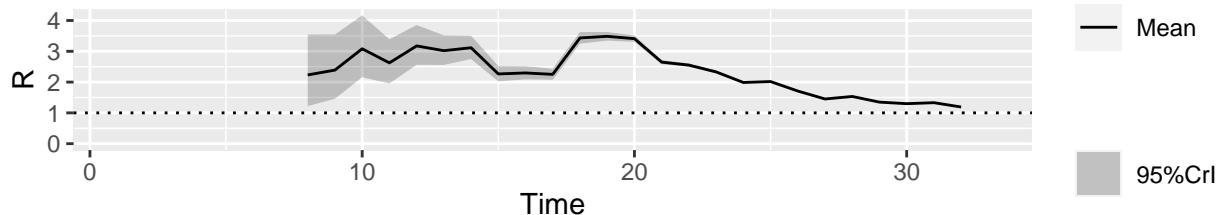
```

```
plot(epiestim)
```

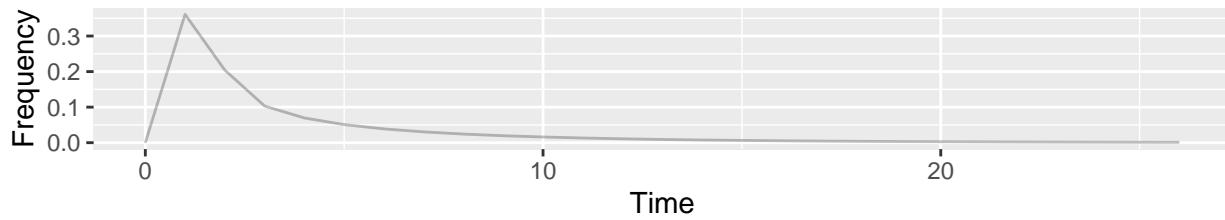
Epidemic curve



Estimated R



Explored SI distribution



```
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells    name      grob
## 1 incid 1 (1-1,1-1) arrange gtable[layout]
## 2 R     2 (2-2,1-1) arrange gtable[layout]
## 3 SI    3 (3-3,1-1) arrange gtable[layout]
```

6 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 work
## See https://github.com/r-spatial/sf/issues/1177#issuecomment-541858742
head(names(geodata$geometry))

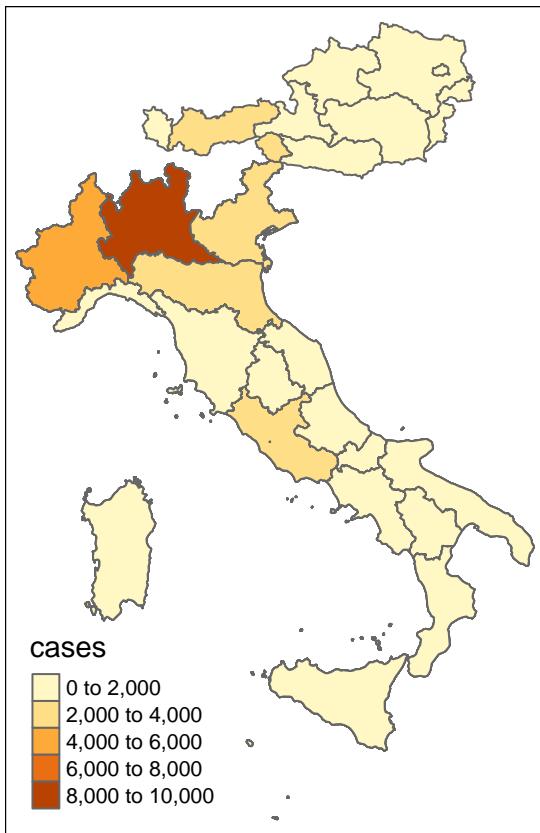
## [1] "0" "1" "2" "3" "4" "5"
names(geodata$geometry)=NULL

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')
```



7 Epicurves

```

library(sars2pack)

## Loading required package: R0
## Loading required package: MASS
## Loading required package: sf
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
library(DT)
library(dplyr)

##
## Attaching package: 'dplyr'
## The following object is masked from 'package:MASS':
##   select
## The following objects are masked from 'package:stats':
##   filter, lag
## The following objects are masked from 'package:base':
##   intersect, setdiff, setequal, union

```

7.1 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: <input type="text"/>					
	ProvinceState	CountryRegion	Lat	Long	date	count	subset
1		Albania	41.1533	20.1683	2020-02-24	0	deaths
2		Algeria	28.0339	1.6596	2020-03-06	0	deaths
3		Angola	-11.2027	17.8739	2020-02-03	0	deaths
4		Antigua and Barbuda	17.0608	-61.7964	2020-02-19	0	deaths
5		Antigua and Barbuda	17.0608	-61.7964	2020-02-22	0	deaths
6		Antigua and Barbuda	17.0608	-61.7964	2020-03-06	0	confirmed
7		Antigua and Barbuda	17.0608	-61.7964	2020-03-29	0	deaths
8		Argentina	-38.4161	-63.6167	2020-03-14	34	confirmed
9		Argentina	-38.4161	-63.6167	2020-03-21	4	deaths
10		Armenia	40.0691	45.0382	2020-02-08	0	deaths

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

7.2 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	Oberösterreich	7					2020-03-08T08:00:00Z	2020-03-08
2	AT	Kärnten	1					2020-03-08T11:45:00Z	2020-03-08
3	AT	Kärnten	5	0	0			2020-03-14T15:00:00Z	2020-03-14
4	AT	Steiermark	233	0	0			2020-03-19T08:00:00Z	2020-03-19
5	AT	Oberösterreich	421	0	0			2020-03-20T15:00:00Z	2020-03-20
6	AT	Kärnten	74	0	0			2020-03-21T08:00:00Z	2020-03-21
7	AT	Steiermark	423	0	0			2020-03-22T15:00:00Z	2020-03-22
8	AT	Vorarlberg	379	0	0			2020-03-24T15:00:00Z	2020-03-24
9	AT	Burgenland	115	0	0			2020-03-26T15:00:00Z	2020-03-26
10	AT	Burgenland	140			11	4	2020-03-27T09:30:00Z	2020-03-27

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

7.3 United States datasets

7.3.1 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	2070	Dillingham Census Area	AK	2	confirmed	2020-03-23	0
2	2090	Fairbanks North Star Borough	AK	2	deaths	2020-02-14	0
3	2158	Kusilvak Census Area	AK	2	confirmed	2020-02-09	0
4	2158	Kusilvak Census Area	AK	2	confirmed	2020-02-11	0
5	2261	Valdez-Cordova Census Area	AK	2	confirmed	2020-02-07	0
6	1007	Bibb County	AL	1	deaths	2020-02-12	0
7	1041	Crenshaw County	AL	1	deaths	2020-01-31	0
8	1047	Dallas County	AL	1	confirmed	2020-03-12	0
9	1053	Escambia County	AL	1	deaths	2020-02-17	0
10	1053	Escambia County	AL	1	confirmed	2020-02-29	0

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

7.3.2 New York Times

```
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	2020-03-14	Baldwin	Alabama	01003	0	deaths
2	2020-03-26	Bullock	Alabama	01011	2	confirmed
3	2020-03-28	Butler	Alabama	01013	1	confirmed
4	2020-03-22	Lamar	Alabama	01075	0	deaths
5	2020-03-25	Lee	Alabama	01081	0	deaths
6	2020-04-01	Mobile	Alabama	01097	57	confirmed
7	2020-03-16	Montgomery	Alabama	01101	0	deaths
8	2020-03-23	Washington	Alabama	01129	1	confirmed
9	2020-03-21	Kenai Peninsula Borough	Alaska	02122	0	deaths
10	2020-03-25	Apache	Arizona	04001	0	deaths

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

8 Map visualizations

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

ejhu = enriched_jhu_data()

glimpse(ejhu)

## Observations: 37,152
## Variables: 20
## $ name      <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghan...
## $ topLevelDomain <list> [".af", ".af", ".af", ".af", ".af", ".af", ...
## $ alpha2Code   <chr> "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...
## $ 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, ...
## $ CountryRegion <chr> "Afghanistan", "Afghanistan", "Afghanistan", ...
## $ Lat <dbl> 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, ...
## $ Long <dbl> 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, 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)

```

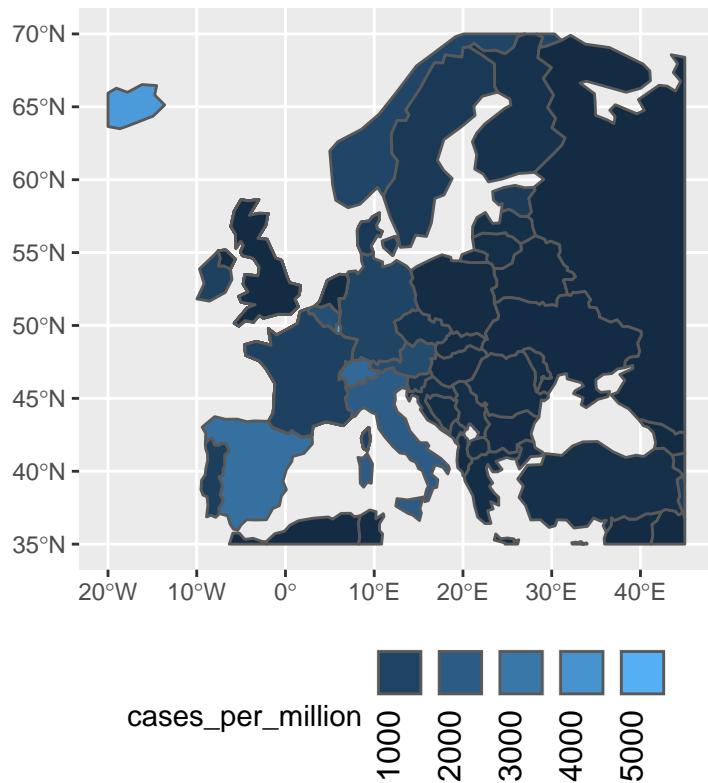
The R package `ggplot2` has geospatial plotting capabilities built in for geospatial `simple features` (`sf`) data types. In this first plot, we focus in on Europe.

```

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')

```

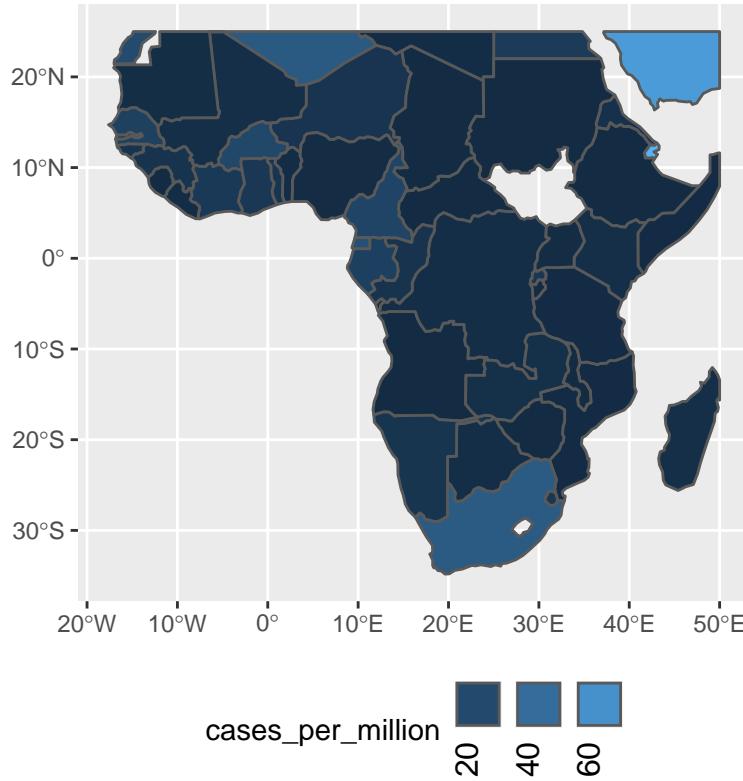
Cases per Million Inhabitants



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')
```

Cases per Million Inhabitants



8.1 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)
tm_shape(w2,id='name.x', name=c('cases_per_million'),popup=c('pop_est')) +
  tm_polygons(c('cases_per_million','count','well_being'),
  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)
```



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.