**Introduction**

As I write this on 4th March, 2020, the world is on the cusp of a global COVID-19 pandemic caused by the SARS-Cov2 virus. Every news report is dominated by alarming, and ever-growing cumulative counts of global cases and deaths due to COVID-19. [Dashboards of global spread](https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html) are beginning to light up like Christmas trees.

For R users, an obvious question is: “Does R have anything to offer in helping to understand the situation?”.

The answer is: “Yes, a lot!”.

In fact, R is one of the tools of choice for outbreak epidemiologists, and a quick search will yield many R libraries on CRAN and elsewhere devoted to outbreak management and analysis. This post doesn’t seek to provide a review of the available packages – rather it illustrates the utility of a few of the excellent packages available in the [**R** **E**pidemics **Con**sortium (**RECON**) suite](https://www.repidemicsconsortium.org/), as well as the use of base R and [tidyverse](https://www.tidyverse.org/) packages for data acquisition, wrangling and visualization.

**Data acquisition**

Obtaining detailed, accurate and current data for the COVID-19 epidemic is not as straightforward as it might seem. Various national and provincial/governmental web sites in affected countries provide detailed summary data on incident cases, recovered cases and deaths due to the virus, but these data tend to be in the form of counts embedded in (usually non-English) text.

There are several potential sources of data which have been abstracted and collated from such governmental sites. A widely-used source is a [dataset](https://github.com/CSSEGISandData/COVID-19) which is being collated by Johns Hopkins University Center for Systems Science and Engineering (JHU CCSE) and which is used as the source for [the dashboard](https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6) mentioned above. It is very easy to use – just read CSV files from the appropriate GitHub URL. However, it lacks detail (that wasn’t its intended purpose) and contains quite a few missing or anomalous data points when examined as a differenced daily time-series of incident cases.

Another set of convenient sources are relevant wikipedia pages, such as [this one for China](https://en.wikipedia.org/wiki/Timeline_of_the_2019%E2%80%9320_Wuhan_coronavirus_outbreak). There are equivalent pages for [Japan](https://en.wikipedia.org/wiki/Template:2019%E2%80%9320_coronavirus_outbreak_data/Japan_medical_cases), [South Korea](https://en.wikipedia.org/wiki/Template:2019%E2%80%9320_coronavirus_outbreak_data/South_Korea_medical_cases), [Iran](https://en.wikipedia.org/wiki/Template:2019%E2%80%9320_coronavirus_outbreak_data/Iran_medical_cases), [Italy](https://en.wikipedia.org/wiki/Template:2019%E2%80%9320_coronavirus_outbreak_data/Italy_medical_cases) and many other countries. These wikipedia pages tend to be much more detailed and are well-referenced back to the original source web pages, but they are quite challenging to web-scrape because the format of the tables in which the data appears changes quite often, as various wikipedia contributors adjust their appearance. Nonetheless, we’ll scrape some detailed data about the initial set of identified COVID-19 cases in the United States (as at 4th March) from a suitable wikipedia page. The saving grace is that wikipedia pages are versioned, and thus it is possible to scrape data from a specific version of a table. But if you want daily updates to your data, using wikipedia pages as a source will involve daily maintenance of your web scraping code.

**Incidence data collated by John Hopkins University**

Acquiring these data is easy. The time-series format they provide is the most convenient for our purposes. We’ll also remove columns of US cases associated with the *Diamond Princess* cruise ship because we can assume that those cases were (home) quarantined on repatriation and were unlikely, or at least a lot less likely, to give rise to further cases. We also shift the dates in the JHU data back one day reflect US time zones, somewhat approximately, because the original dates are with respect to midnight UTC (Greenwich time). That is necessary because we will be combining the JHU data with wikipedia-sourced US data, which is tabulated by dates referencing local US time zones.

We also need to difference the JHU data, which is provided as cumulative counts, to get daily incident counts. Incident counts of cases are a lot more epidemiologically useful than cumulative counts. dplyr makes short work of all that.

jhu\_url <- paste("<https://raw.githubusercontent.com/CSSEGISandData/>",

"COVID-19/master/csse\_covid\_19\_data/", "csse\_covid\_19\_time\_series/",

"time\_series\_19-covid-Confirmed.csv", sep = "")

us\_confirmed\_long\_jhu <- read\_csv(jhu\_url) %>% rename(province = "Province/State",

country\_region = "Country/Region") %>% pivot\_longer(-c(province,

country\_region, Lat, Long), names\_to = "Date", values\_to = "cumulative\_cases") %>%

# adjust JHU dates back one day to reflect US time, more or

# less

mutate(Date = mdy(Date) - days(1)) %>% filter(country\_region ==

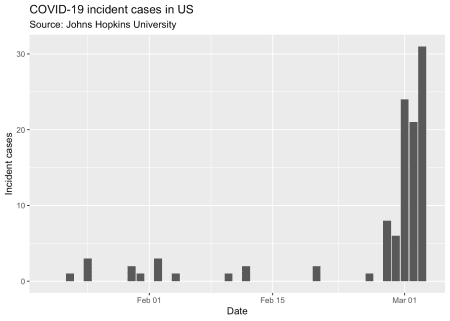
"US") %>% arrange(province, Date) %>% group\_by(province) %>%

mutate(incident\_cases = c(0, diff(cumulative\_cases))) %>%

ungroup() %>% select(-c(country\_region, Lat, Long, cumulative\_cases)) %>%

filter(str\_detect(province, "Diamond Princess", negate = TRUE))

We can now visualize the JHU data using ggplot2, summarized for the whole of the US:



So, not a lot of data to work with as yet. One thing that is missing is information on whether those cases were *imported*, that is, the infection was most likely acquired outside the US, or whether they were *local*, as a result of local transmission (possibly from imported cases).

**Scraping wikipedia**

The wikipedia page on COVID-19 in the US contains several tables, one of which contains a line-listing of all the initial US cases, which at the time of writing appeared to be fairly complete up to 2nd March. We’ll scrape that line-listing table from [this version of the wikipedia page](https://en.wikipedia.org/w/index.php?title=2020_coronavirus_outbreak_in_the_United_States&oldid=944107102), leveraging the excellent rvest package, part of the tidyverse. Once again we need to do a bit of data wrangling to get the parsed table into a usable form.

# the URL of the wikipedia page to use is in wp\_page\_url

wp\_page\_url

## [1] "<https://en.wikipedia.org/w/index.php?title=2020_coronavirus_outbreak_in_the_United_States&oldid=944107102>"

# read the page using the rvest package.

outbreak\_webpage <- read\_html(wp\_page\_url)

# parse the web page and extract the data from the eighth

# table

us\_cases <- outbreak\_webpage %>% html\_nodes("table") %>% .[[8]] %>%

html\_table(fill = TRUE)

# The automatically assigned column names are OK except that

# instead of County/city and State columns we have two

# columns called Location, due to the unfortunate use of

# colspans in the header row. The tidyverse abhors

# duplicated column names, so we have to fix those, and make

# some of the other colnames a bit more tidyverse-friendly.

us\_cases\_colnames <- colnames(us\_cases)

us\_cases\_colnames[min(which(us\_cases\_colnames == "Location"))] <- "CityCounty"

us\_cases\_colnames[min(which(us\_cases\_colnames == "Location"))] <- "State"

us\_cases\_colnames <- us\_cases\_colnames %>% str\_replace("Location",

"CityCounty") %>% str\_replace("Location", "State") %>% str\_replace("Case no.",

"CaseNo") %>% str\_replace("Date announced", "Date") %>% str\_replace("CDC origin type",

"OriginTypeCDC") %>% str\_replace("Treatment facility", "TreatmentFacility")

colnames(us\_cases) <- us\_cases\_colnames

# utility function to remove wikipedia references in square

# brackets

rm\_refs <- function(x) stringr::str\_split(x, "\\[", simplify = TRUE)[,

1]

# now remove references from CaseNo column, convert it to

# integer, convert the date column to date type and then lose

# all rows which then have NA in CaseNo or NA in the date

# column

us\_cases <- us\_cases %>% mutate(CaseNo = rm\_refs(CaseNo)) %>%

mutate(CaseNo = as.integer(CaseNo), Date = as.Date(parse\_date\_time(Date,

c("%B %d, %Y", "%d %B, %Y")))) %>% filter(![is.na](http://is.na)(CaseNo),

![is.na](http://is.na)(Date)) %>% # convert the various versions of unknown into NA in the

# OriginTypeCDC column

mutate(OriginTypeCDC = if\_else(OriginTypeCDC %in% c("Unknown",

"Undisclosed"), NA\_character\_, OriginTypeCDC))

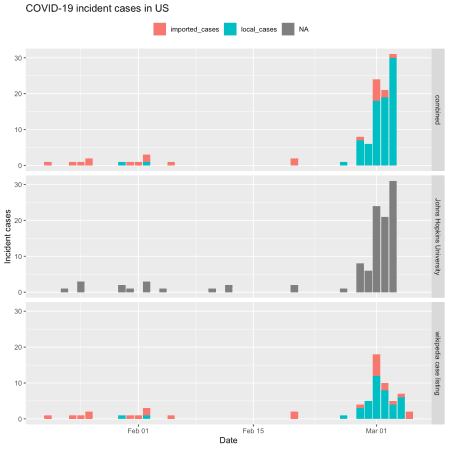
At this point we have an acceptably clean table, that looks like this:

| **US cases of COVID-19 as at 3rd March 2020** | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | |
| **CaseNo** | **Date** | **Status** | **OriginTypeCDC** | **Origin** | **CityCounty** | **State** | **TreatmentFacility** | **Sex** | **Age** |
| Last 6 rows | | | | | | | | | |
| 60 | 2020-03-04 | Deceased | NA | Unknown | Placer County | California | Hospitalized | Male | Elderly adult |
| 61 | 2020-03-04 | Confirmed | NA | Unknown | Santa Clara County | California | Hospitalized | Male | Undisclosed |
| 62 | 2020-03-04 | Confirmed | Person-to-person spread | Undisclosed | Santa Clara County | California | In-home isolation | Male | Undisclosed |
| 63 | 2020-03-04 | Confirmed | Person-to-person spread | Undisclosed | Santa Clara County | California | In-home isolation | Male | Undisclosed |
| 64 | 2020-03-04 | Confirmed | NA | Unknown | Williamson County | Tennessee | In-home isolation | Male | 44 |
| 65 | 2020-03-05 | Confirmed | Travel-related | Unknown | Clark County | Nevada | VA Southern Nevada Healthcare System | Male | 50’s |
| 66 | 2020-03-05 | Confirmed | Travel-related | Unknown | Chicago | Illinois | Rush University Medical Center | Male | 20’s |
| First 6 rows | | | | | | | | | |
| 1 | 2020-01-21 | Recovered | Travel-related | Wuhan, China | Snohomish County | Washington | Providence Regional Medical Center Everett | Male | 35 |
| 2 | 2020-01-24 | Recovered | Travel-related | Wuhan, China | Chicago | Illinois | St. Alexius Medical Center | Female | 60s |
| 3 | 2020-01-25 | Recovered | Travel-related | Wuhan, China | Orange County | California | In-home isolation | Male | 50s |
| 4 | 2020-01-26 | Confirmed | Travel-related | Wuhan, China | Los Angeles County | California | Undisclosed | Undisclosed | Undisclosed |
| 5 | 2020-01-26 | Recovered | Travel-related | Wuhan, China | Tempe | Arizona | In-home isolation | Male | Under 60 |
| 6 | 2020-01-30 | Recovered | Person-to-person spread | Spouse | Chicago | Illinois | St. Alexius Medical Center | Male | 60s |
| Source: [wikipedia: 2020 coronavirus outbreak in the United States](https://en.wikipedia.org/w/index.php?title=2020_coronavirus_outbreak_in_the_United_States&oldid=944107102) | | | | | | | | | |

We’ll won’t bother cleaning all the columns, because we’ll only be using a few of them here.

OK, so we’ll use the wikipedia data prior to 20th February, but the JHU counts after that, but we’ll use the wikipedia data after 20th February to divide the JHU counts into travel-related or not, more or less. That will give us a data set which distinguishes local from imported cases, at least to the extent of completeness of our data sources. Outbreak epidemiology involves practicing the art of *good enough for now*.

Let’s visualize each of our two sources, and the combined data set.



**Analysis with the earlyR and EpiEstim packages**

The [earlyR](https://www.repidemicsconsortium.org/earlyR/) package, as its name suggests, is intended for use early in an outbreak to calculate several key statistics. In particular the get\_R() function in earlyR calculates a maximum-likelihood estimate for the reproduction number, which is the mean number of new cases each infected person give rise to. The overall\_infectivity() function in the EpiEstim package calculates \(\lambda\) (lambda), which is a relative measure of the current “force of infection” or infectivity of an outbreak:

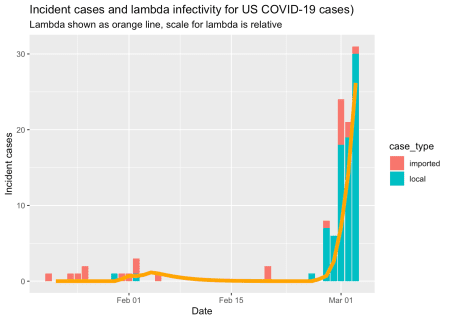
\[ \lambda = \sum\_{s=1}^{t-1} {y\_{s} w (t – s)} \]

where \(w()\) is the probability mass function (PMF) of the serial interval, and \(y\_s\) is the incidence at time \(s\). If \(\lambda\) is falling, then that’s good: if not, bad.

The critical parameter for these calculations is the distribution of *serial intervals* (SI), which is the time between the date of onset of symptoms for a case and the dates of onsets for any secondary cases that case gives rise to. Typically a discrete \(\gamma\) distribution for these *serial intervals* is assumed, parameterised by a mean and standard deviation, although more complex distributions are probably more realistic.

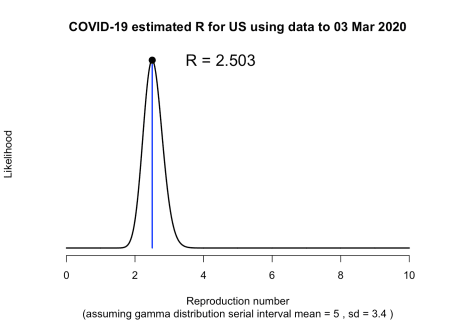
For now, we’ll just use a discrete \(\gamma\) distribution with a mean of 5.0 days and a standard deviation of 3.4 for the *serial interval* distribution. That mean is less than estimates published earlier in the outbreak in China, but appears to be closer to the behavior of the COVID-19 virus (based on a personal communication from an informed source who is party to WHO conference calls on COVID-19). Obviously a sensitivity analysis, using different but plausible *serial interval* distributions, should be undertaken, but we’ll omit that here for the sake of brevity.

Note that only local transmission is used to calculate \(\lambda\). If we just use the JHU data, which includes both local and imported cases, our estimates of \(\lambda\) would be biased, upwards.



The US is not winning the war against COVID-19, but it is early days yet.

We can also estimate the *reproduction number*.



That estimate of \(R\_{0}\) is consistent with those reported recently by WHO, although higher than some initial estimates from Wuhan. The key thing is that it is well above 1.0, meaning that the outbreak is growing, rapidly.

**Fitting a log-linear model to the *epidemic curve***

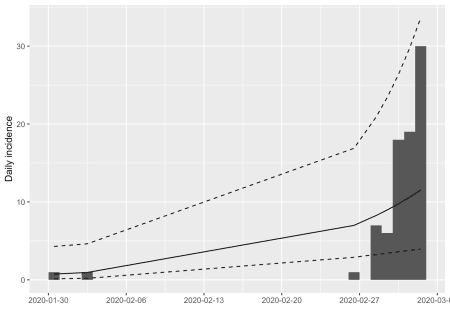
We can also use functions in the **RECON** incidence package to fit a log-linear model to our epidemic curve. Typically, two models are fitted, one for the growth-phase and one for the decay phase. Functions are provided in the package for finding the peak of the epidemic curve using naïve and optimizing methods.

But for now, the US outbreak is still in growth phase, so we only fit one curve.

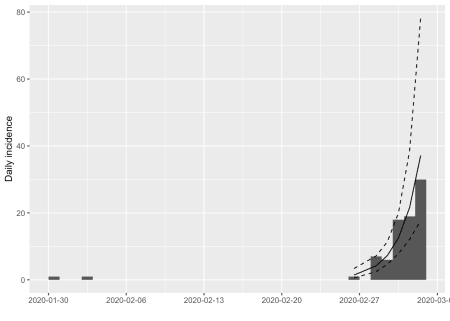
us\_incidence\_fit <- incidence::fit(local\_cases\_obj, split = NULL)

# plot the incidence data and the model fit

plot(local\_cases\_obj) %>% add\_incidence\_fit(us\_incidence\_fit)



That’s clearly not a good fit, because we are including the handful of very early cases that did not appear to establish sustained chains of local transmission. Let’s exclude them.

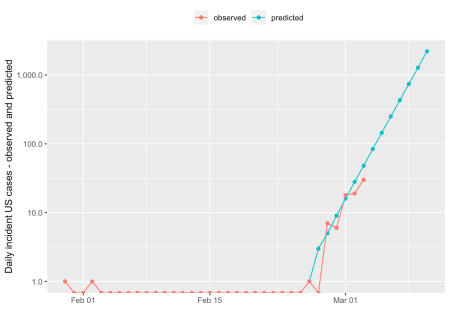


That’s a much better fit!

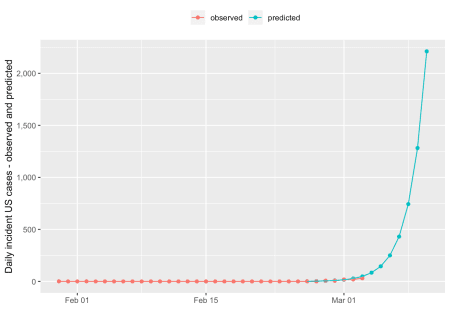
From the that model, we can extract various (very preliminary at this early stage) parameters of interest: the **growth rate is 0.54** (95% CI 0.32 – 0.77), which is equivalent to a **doubling time of 1.3 days** (95% CI 0.9 – 2.2 days).

That’s all a bit alarming, but these estimates are almost certainly biased because cases are being tabulated by their **date of reporting**, and not by their **date of symptom onset**. Nonetheless, as the the epidemic in the US spreads, the bias due to use of date of reporting should diminish, provided that testing and reporting of cases occurs swiftly and consistently. The ability to test and report cases promptly is a key indicator of the quality of public health intervention capability.

We can also project how many cases might be expected in the next week, assuming that public health controls don’t start to have an effect, and subject to the estimation biases discussed above, and bearing it mind our model is fitted to just a few days of data, so far. We’ll plot on a log scale so the observed cases so far aren’t obscured by the predicted values.



On a linear scale, that looks like this:

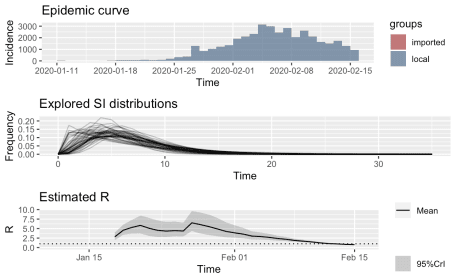


So, we predict, on very sketchy preliminary data, over 2211 new cases per day by 10 March. That’s probably an overestimate, due to potential reporting-date-not-onset-date bias already discussed, but it nevertheless illustrates the exponential nature of typical epidemic behavior.

Humans tend to use linear heuristics when contemplating trends, and thus tend to be surprised by such exponential behavior, and fail to plan for it accordingly.

**Estimating the instantaneous *effective reproduction ratio***

One other statistic which the EpiEstim package estimates is the instantaneous effective reproduction number, based on an adjustable sliding window. This is very useful for assessing how well public health interventions are working. There isn’t enough US data available, yet, to estimate this, but here is an example of a plot of the instantaneous \(R\_{e}\) for Hubei province in China.



You can clearly see the effect of the lock-down implemented in Hubei province and Wuhan city on or around 24th January, and the fact that the instantaneous \(R\_{e}\) started to fall a long time before the daily incidence of new cases reached its peak. Availability of such information helps governmental authorities to keep their nerve and to persist with unpopular public health measures, even in the face of rising incidence.

**Conclusion**

In this post we have seen how base R, the tidyverse packages, and libraries provided by [**R** **E**pidemics **Con**sortium (**RECON**)](https://www.repidemicsconsortium.org/) can be used to assemble COVID-19 outbreak data, visualize it, and estimate some key statistics from it which are vital for assessing and planning the public health response to this disease. There are several other libraries for R than can also be used for such purposes. It should only take a small team of data scientists a few days, using these and related tools, to construct ongoing reports or decision support tools, able to be updated continuously, or at least daily, to help support public health authorities in their (literally) life-and-death fight against COVID-19.