EpiEstim: Practical

Rob Challen

2024-03-12

We are going to use some UK COVID-19 data to estimate the reproduction number using infectivity profiles from the literature. We'll use a couple of data sets and compare.

Grab some data

First of all lets get some retrospective COVID data from the UK coronavirus site. The best place to go is to this URL: https://coronavirus.data.gov.uk/details/download, and select `AreaType = Nation`; `AreaName = England`; `Metrics = "newCasesBySpecimenDate"`. We are going to focus only on the first wave up to 1st Sept 2020:

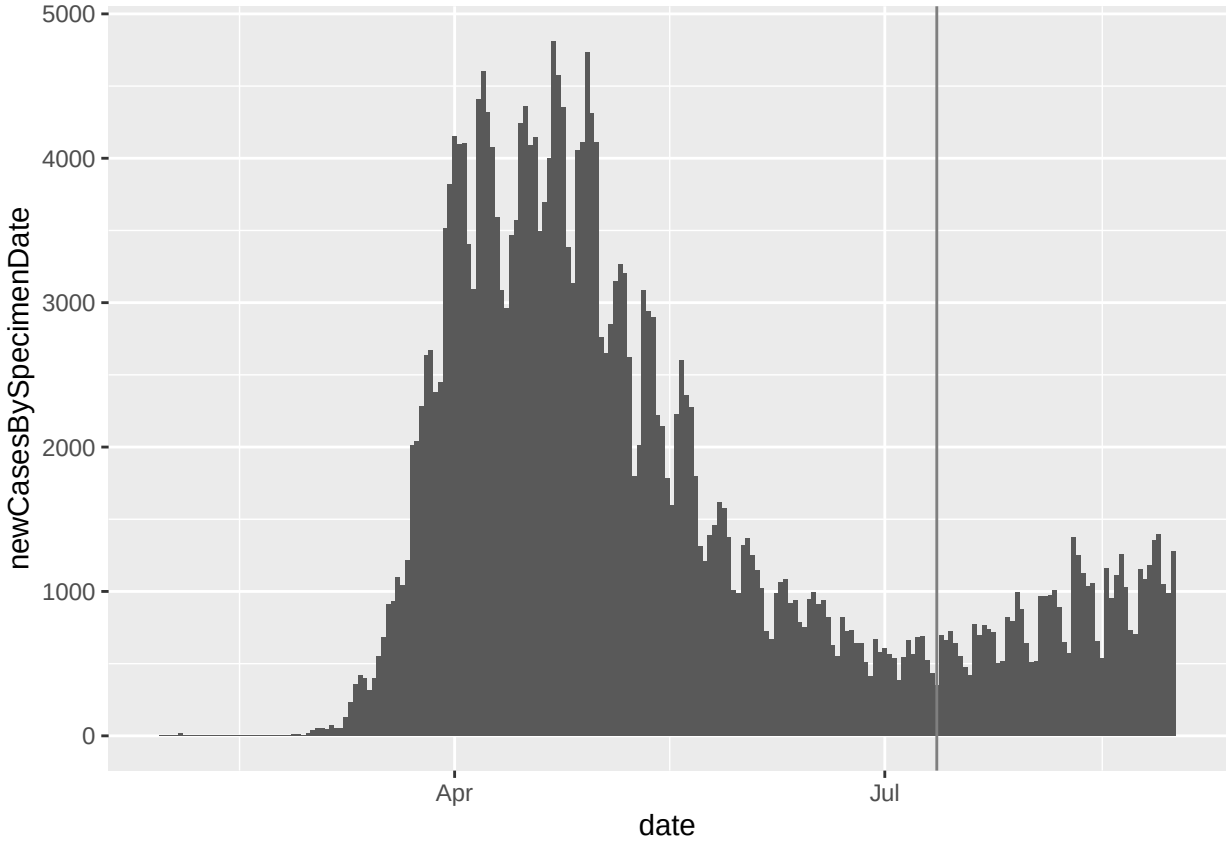
- Read the link to the CSV file in using `readr::read_csv`.
- Filter the data set to < 2020-09-01.
- Quickly explore the structure using `glimpse`.

```
england_cases = readr::read_csv("https://api.coronavirus.data.gov.uk/v2/data?areaType=nation&areaCode=E
  filter(date < "2020-09-01")

england_cases %>% glimpse()
```

```
## Rows: 215
## Columns: 5
## $ areaCode               <chr> "E92000001", "E92000001", "E92000001", "E920000~
## $ areaName               <chr> "England", "England", "England", "England", "En~
## $ areaType               <chr> "nation", "nation", "nation", "nation", "nation~
## $ date                   <date> 2020-08-31, 2020-08-30, 2020-08-29, 2020-08-28~
## $ newCasesBySpecimenDate <dbl> 1279, 987, 1051, 1395, 1356, 1181, 1085, 1156, ~
```

We will use cases as a proxy for infections. We can quickly plot a timeline with `ggplot`

```
ggplot(england_cases,aes(x=date,y=newCasesBySpecimenDate))+
  geom_bar(stat="identity")+
  geom_vline(xintercept = as.Date("2020-07-12"), colour="grey50")
```



The epidemic clearly reached a low in late June, early July. Lets see if we can get EpiEstim to give us a time.

First we need:

- ✓ Incident cases
- ✗ Prior for R
- ✗ Window
- ✗ Infectivity profile (e.g. Generation time distribution)

Infectivity profile

We are going to use an estimate from the literature. We will use Ganyani 2020 estimate of the generation time from Singapore:

```
tmp = tempfile()
download.file("https://github.com/bristol-vaccine-centre/edam-epiestim/raw/main/input/serial-interval-li
si_lit = readxl::read_excel(tmp)
si_ganyani = si_lit %>% filter(estimate_type == "generation interval" & population == "Singapore")

si_ganyani %>% glimpse()
```

```
## Rows: 1
## Columns: 13
## $ mean_si_estimate         <dbl> 5.2
## $ mean_si_estimate_low_ci  <dbl> 3.78
## $ mean_si_estimate_high_ci <dbl> 6.78
## $ std_si_estimate          <dbl> 1.72
## $ std_si_estimate_low_ci   <dbl> 0.91
```

```
## $ std_si_estimate_high_ci  <dbl> 3.93
## $ sample_size              <dbl> 54
## $ population               <chr> "Singapore"
## $ assumed_distribution     <chr> "gamma"
## $ estimate_type            <chr> "generation interval"
## $ label                    <chr> "Ganyani et al. 2020 (singapore)"
## $ source                   <chr> "Ganyani, T. et al. Estimating the generation~
## $ note                     <chr> NA
```

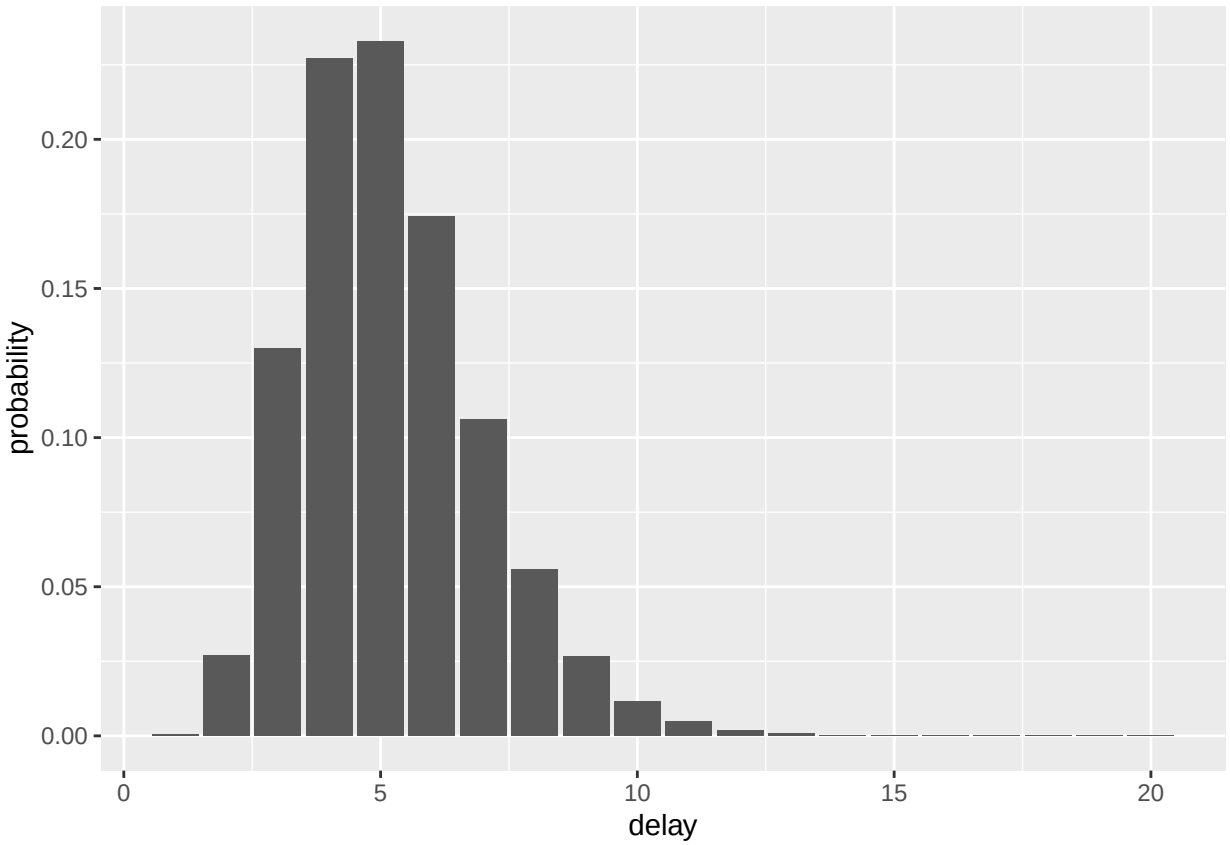
We can get EpiEstim to calculate the discrete infectivity profile distribution using the `EpiEstim::discr_si` function.

- Get a discrete SI distribution.
- Plot it.
- Does the mean of the SI distribution equal that of the original ganyani estimate?

$$mean = \sum x \times P(x)$$

$$std = \sqrt{\sum (x - mean)^2 \times P(x)}$$

```
si_probability = EpiEstim::discr_si(1:20, si_ganyani$mean_si_estimate, si_ganyani$std_si_estimate)

mean_si = sum(1:20*si_probability)
sd_si = sqrt(sum((1:20-mean_si)^2*si_probability))

print(mean_si)
print(sd_si)

ggplot(tibble(
  delay = 1:20,
  probability = si_probability
), aes(x=delay, y=probability))+geom_bar(stat="identity")
```




```
## [1] 5.199997
## [1] 1.767775
```

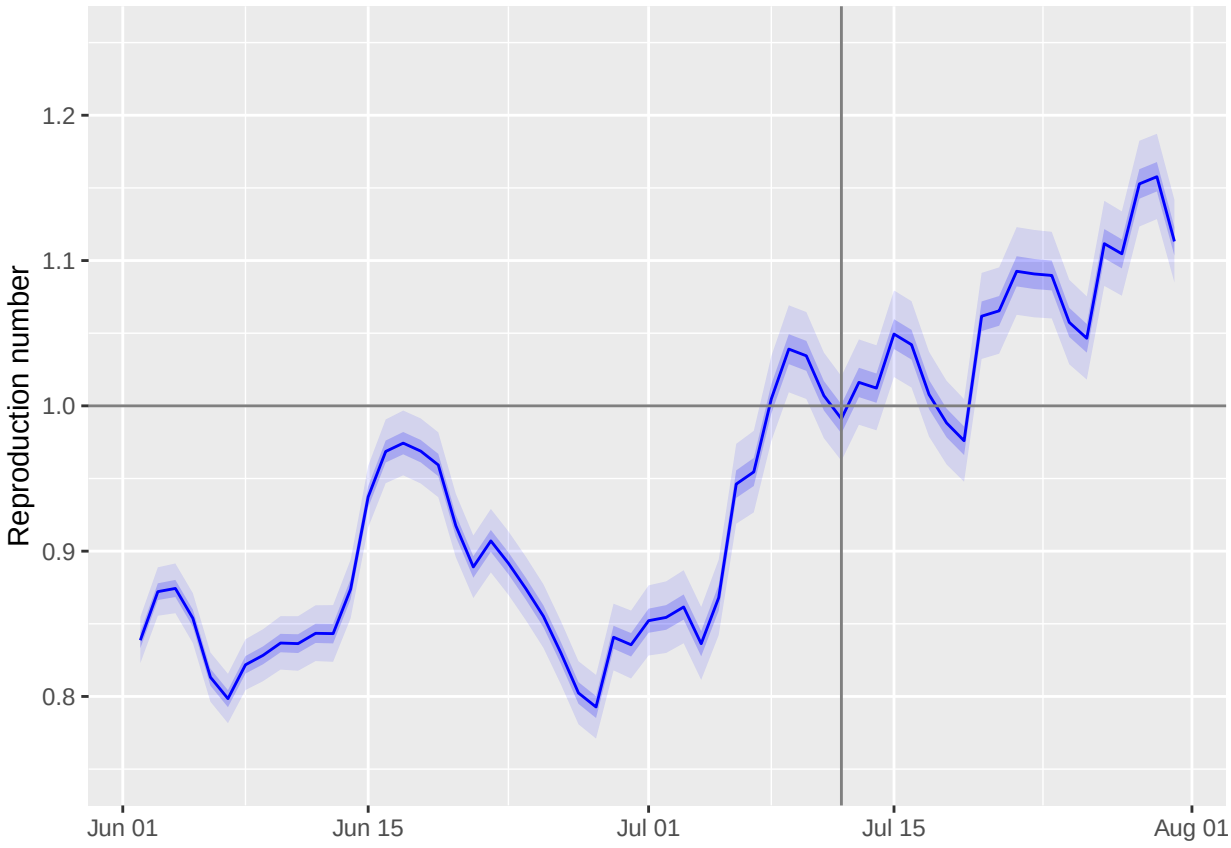
We don't actually need the SI distribution as EpiEstim does it automatically. We select a minimally informative prior for R and select a window of duration 7 days. `EpiEstim::make_config` helps you set up a configuration for EpiEstim, and then you can use `EpiEstim::estimate_R`:

```
window = 7
t_start = 2:(nrow(england_cases)-window)
t_end = t_start+window

simple_config = EpiEstim::make_config(
  method = "parametric_si",
  mean_si = si_ganyani$mean_si_estimate,
  std_si = si_ganyani$std_si_estimate,
  mean_prior = 1,
  std_prior = 1,
  t_start = t_start,
  t_end = t_end
)

simple_ts = england_cases %>%
  select(dates = date, I = newCasesBySpecimenDate) %>%
  arrange(dates)

epiestim_out = estimate_R(simple_ts,method = "parametric_si", config=simple_config)

epiestim_out$R %>% glimpse()
```

```
## Rows: 207
## Columns: 11
## $ t_start             <dbl> 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
## $ t_end               <dbl> 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,~
## $ `Mean(R)`           <dbl> 2.2048534, 1.7084275, 1.4283350, 1.2717674, 0.5223~
## $ `Std(R)`            <dbl> 0.4811385, 0.3487313, 0.2856670, 0.2543535, 0.1575~
## $ `Quantile.0.025(R)` <dbl> 1.36483902, 1.09462174, 0.92434312, 0.82302083, 0.~
## $ `Quantile.0.05(R)`  <dbl> 1.4774644, 1.1780347, 0.9931000, 0.8842409, 0.2929~
## $ `Quantile.0.25(R)`  <dbl> 1.8641464, 1.4621088, 1.2267137, 1.0922469, 0.4093~
## $ `Median(R)`         <dbl> 2.1699562, 1.6847588, 1.4093364, 1.2548513, 0.5066~
## $ `Quantile.0.75(R)`  <dbl> 2.5075615, 1.9289700, 1.6092652, 1.4328649, 0.6182~
## $ `Quantile.0.95(R)`  <dbl> 3.0513091, 2.3195736, 1.9283896, 1.7170083, 0.8055~
## $ `Quantile.0.975(R)` <dbl> 3.2430641, 2.4566684, 2.0402393, 1.8165976, 0.8733~
```

The output is a list with `R` and `dates` items. You almost always want to combine the two. Plotting the timeseries:

```
plot_data = epiestim_out$R %>%
  mutate(date = epiestim_out$dates[t_end]) %>%
  filter(date > "2020-06-01" & date < "2020-08-01")

type = "blue"

ggplot(plot_data)+
    geom_line(mapping=aes(x=date, y=`Mean(R)`, colour=type))+
    geom_ribbon(mapping=aes(x=date, ymin=`Quantile.0.25(R)`,ymax=`Quantile.0.75(R)`, fill=type), colour=
    geom_ribbon(mapping=aes(x=date, ymin=`Quantile.0.025(R)`,ymax=`Quantile.0.975(R)`, fill=type), colou
    geom_hline(yintercept = 1, colour="grey50")+
    geom_vline(xintercept = as.Date("2020-07-12"), colour="grey50")+
    scale_color_identity(aesthetics=c("fill","colour"))+
    xlab(NULL)+
    ylab("Reproduction number")+
    coord_cartesian(ylim=c(0.75,1.25))
```



- On the 12th July, if they had asked, what would you advise a senior government official who is thinking about stimulating the economy with a scheme that involves increasing contacts?
- What happens if you change the window?
- When would you have been confident that the epidemic was growing?

"Real time" cases

During the early pandemic there was no government dashboard. Public data was extremely limited, and published haphazardly. A community effort to collate the data emerged and you can still see the effort made (and frustrations encountered) scraping the data today: https://github.com/tomwhite/covid-19-uk-data/tree/master

The last update to this data was Aug 2020. The data for England is in the `data` subdirectory. We can explore what this data looked like at any point in time by looking at the commit history:

https://github.com/tomwhite/covid-19-uk-data/commits/master/data/covid-19-totals-england.csv

Lets look at early July. I'll show you how to get there but the commit for the 12th July (a Sunday) can be found here:

https://github.com/tomwhite/covid-19-uk-data/tree/b5e3384a4a6cb6bbf3aa292395ccd90a36771af1

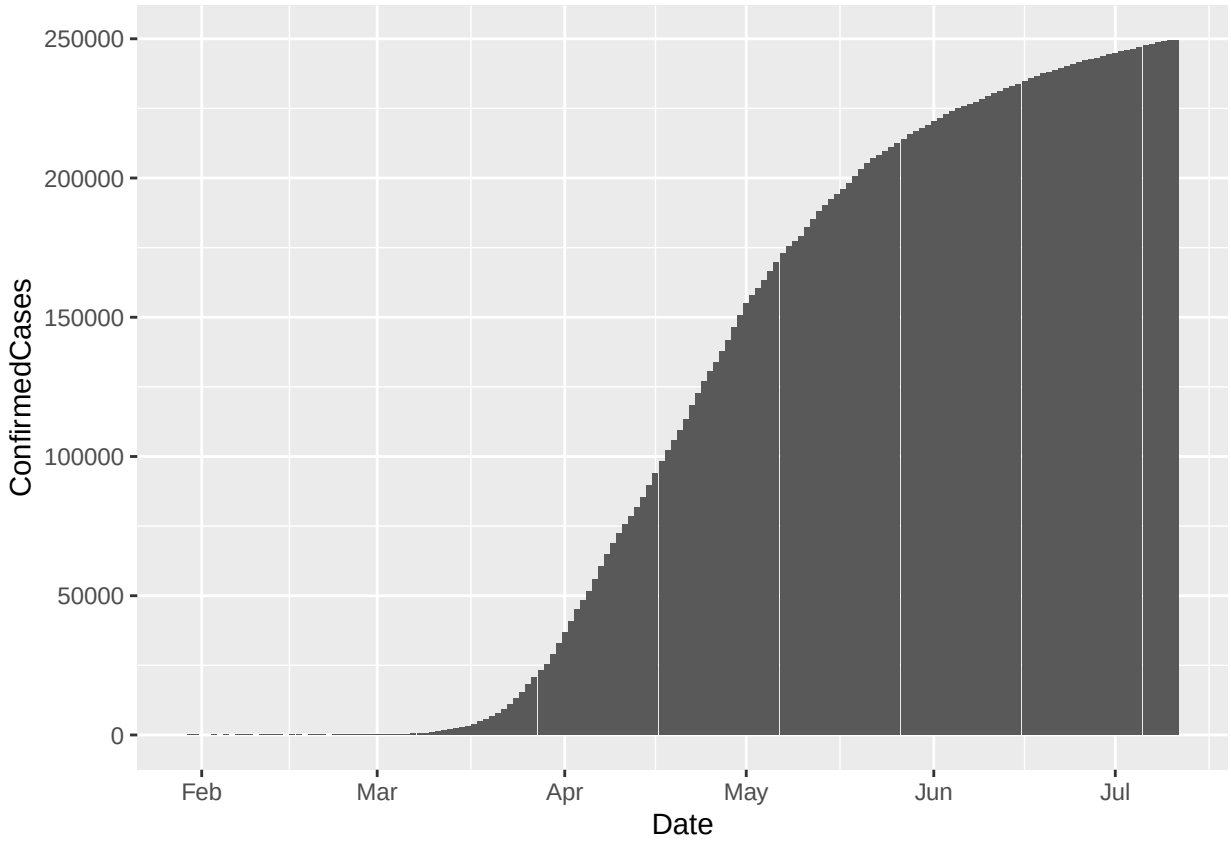
```
tom_white_cases = readr::read_csv("https://github.com/tomwhite/covid-19-uk-data/raw/b5e3384a4a6cb6bbf3a

tom_white_cases %>% glimpse()
```

```
## Rows: 157
## Columns: 4
## $ Date           <date> 2020-01-30, 2020-01-31, 2020-02-03, 2020-02-05, 2020-0~
## $ Tests          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

```
## $ ConfirmedCases <dbl> 2, 3, 8, 3, 6, 7, 8, 9, 18, 19, 20, 18, 21, 22, 11, 12,~
## $ Deaths         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

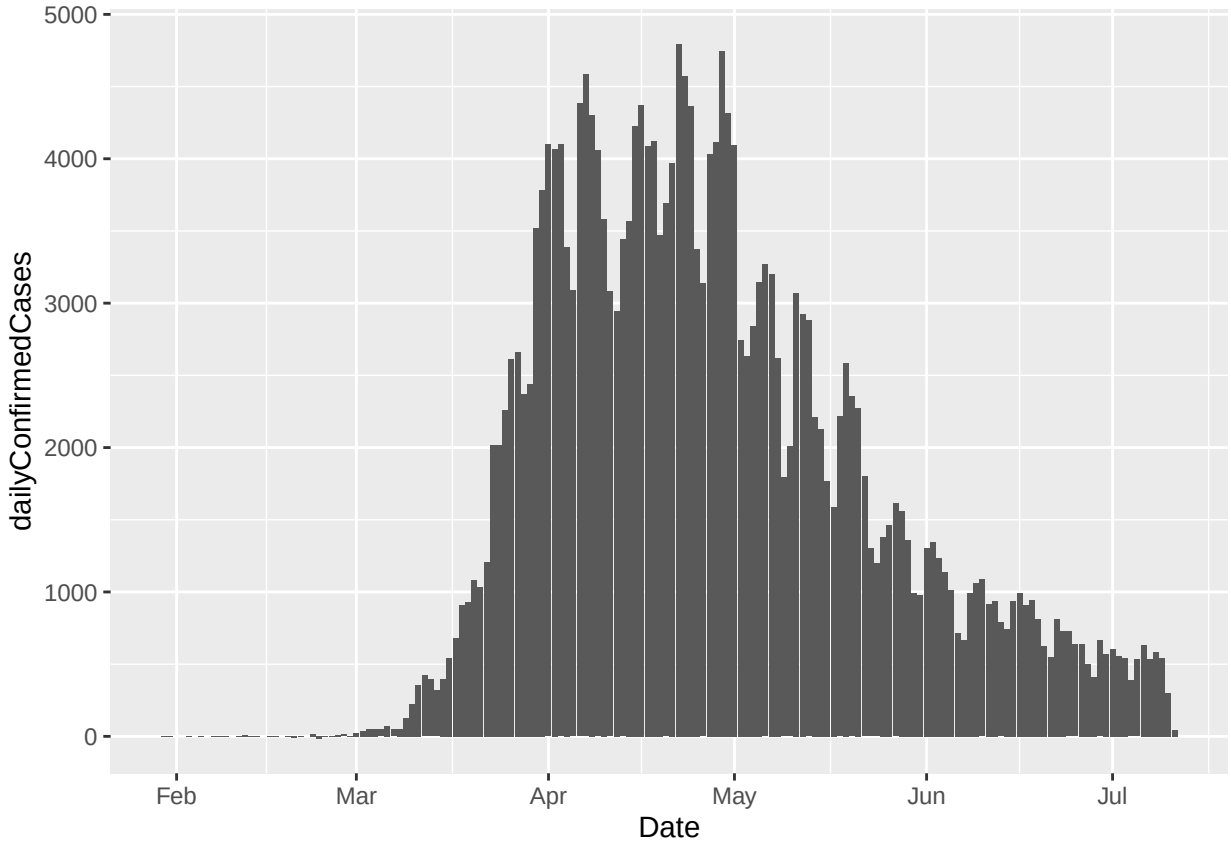
As we can see here this was presented as a cumulative case count:

```
ggplot(tom_white_cases,aes(x=Date,y=ConfirmedCases))+
  geom_bar(stat="identity")
```



To get a daily count we can do a diff:

```
tom_white_cases = tom_white_cases %>%
  mutate(dailyConfirmedCases = ConfirmedCases - lag(ConfirmedCases, default = 0))

ggplot(tom_white_cases,aes(x=Date,y=dailyConfirmedCases))+
  geom_bar(stat="identity")
```



There are several issues with this data set that need fixing before we can use `EpiEstim` which needs

- A complete time series (i.e. no missing days).
- Ordered in time.
- Includes the start of the outbreak.
- Count data is representative of incident infections.
- No missing and no negative values.

Dealing with more "real" data:

- Can you check for completeness and missing values?
- Can you fix the issues? (Hint: `tidyr::complete` and `tidyr::fill` are pretty useful here.)
- Any issues with your solution for missing values and what else could you do?

```
# The data is not conformant
if (any(is.na(tom_white_cases$dailyConfirmedCases))) print("missing values found")
if (any(tom_white_cases$dailyConfirmedCases < 0)) print("negatives found")
if (any(na.omit(lead(tom_white_cases$Date)-tom_white_cases$Date != 1))) print("missing dates")

# To get EpiEstim to work we will have to fill in missing dates and values, and
# ensure the case count is strictly positive.
tom_white_cases_2 = tom_white_cases %>%
  tidyr::complete(
    Date = tidyr::full_seq(Date,1), fill = list(dailyConfirmedCases = NA)
  ) %>%
  mutate(
    dailyConfirmedCases = ifelse(dailyConfirmedCases<0,NA,dailyConfirmedCases)
  ) %>%
  tidyr::fill(dailyConfirmedCases)
```

```
tom_white_cases_2 %>% glimpse()

if (!any(is.na(tom_white_cases_2$dailyConfirmedCases))) print("missing values fixed")
if (!any(tom_white_cases_2$dailyConfirmedCases < 0)) print("negatives fixed")
if (!any(na.omit(lead(tom_white_cases_2$Date)-tom_white_cases_2$Date != 1))) print("missing dates fixed
```

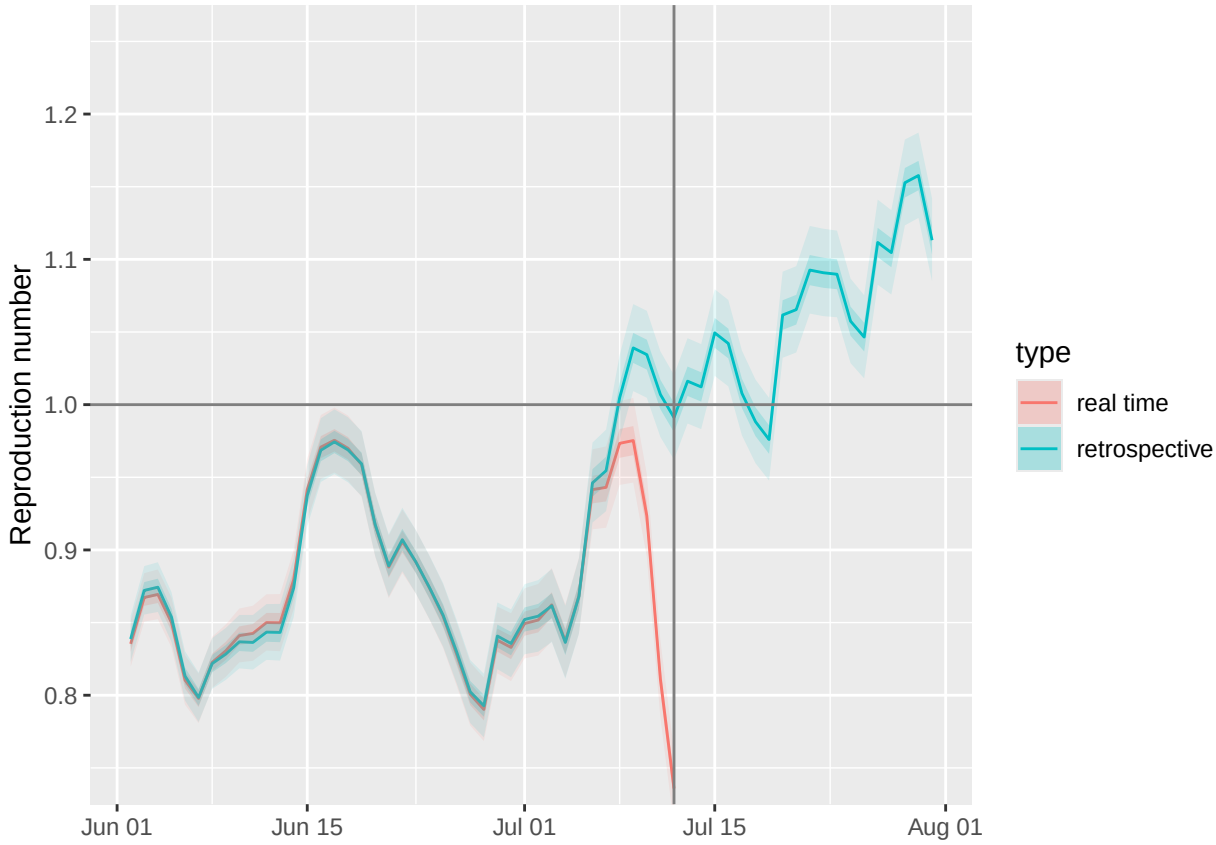
```
## [1] "missing values found"
## [1] "negatives found"
## [1] "missing dates"
## Rows: 165
## Columns: 5
## $ Date                <date> 2020-01-30, 2020-01-31, 2020-02-01, 2020-02-02, 2~
## $ Tests               <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ ConfirmedCases      <dbl> 2, 3, NA, NA, 8, NA, 3, NA, 6, 7, 8, NA, 9, 18, 19~
## $ Deaths              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ dailyConfirmedCases <dbl> 2, 1, 1, 1, 5, 5, 5, 5, 3, 1, 1, 1, 1, 9, 1, 1, 1,~
## [1] "missing values fixed"
## [1] "negatives fixed"
## [1] "missing dates fixed"
```

We now have a fixed data set as if it was the 12th July 2020 (a Sunday).

- Can you modify the code we used above to get a reproduction number estimate for this data and plot it?

```
window = 7
t_start_2 = 2:(nrow(tom_white_cases_2)-window)
t_end_2 = t_start_2 + window

simple_config_2 = EpiEstim::make_config(
  method = "parametric_si",
  mean_si = si_ganyani$mean_si_estimate,
  std_si = si_ganyani$std_si_estimate,
  mean_prior = 1,
  std_prior = 1,
  t_start = t_start_2,
  t_end = t_end_2
)

simple_ts_2 = tom_white_cases_2 %>%
  select(dates = Date, I = dailyConfirmedCases) %>%
  arrange(dates)

epiestim_out_2 = estimate_R(simple_ts_2,method = "parametric_si", config=simple_config_2)

plot_data_2 = epiestim_out_2$R %>%
  mutate(date = epiestim_out_2$dates[t_end]) %>%
  filter(date > "2020-06-01" & date < "2020-08-01")

combined_plot_data = bind_rows(
  plot_data %>% mutate(type = "retrospective"),
  plot_data_2 %>% mutate(type = "real time")
)

ggplot(combined_plot_data)+
```

```
    geom_line(mapping=aes(x=date, y=`Mean(R)`, colour=type))+
    geom_ribbon(mapping=aes(x=date, ymin=`Quantile.0.25(R)`,ymax=`Quantile.0.75(R)`, fill=type), colour=
    geom_ribbon(mapping=aes(x=date, ymin=`Quantile.0.025(R)`,ymax=`Quantile.0.975(R)`, fill=type), colou
    geom_hline(yintercept = 1, colour="grey50")+
    geom_vline(xintercept = as.Date("2020-07-12"), colour="grey50")+
    xlab(NULL)+
    ylab("Reproduction number")+
    coord_cartesian(ylim=c(0.75,1.25))
```



- Can you explain the differences? Why does it drop off at the end?
- What would you have advised if this was the data you saw?
- What do you think about the confidence intervals? Is there a source of uncertainty we are not accounting for.