How to Compute the Reproduction Rate of Covid-19 using the `EpiEstim`-Package*

A Short Practical Guide to Making Your Own Updates

Lukas Puschnig
June 12, 2020

Key Take-Aways and Skills: *Implement the Estimation of the Reproduction Factor of a Communicable Disease on your own System;* **Technologies:** *R incl. the EpiEstim-Package;* **Prerequisites:** *Fundamental R;*

1 Introduction

Since Covid-19 has arrived in our countries, the interest in epidemiologic issues has surged. Most of us have developed the habit of checking the news daily, to see whether the figures have gone up or down. One key indicator that is commonly listed and displayed amongst aggregate figures is the reproduction number (*effektive Reproduktionskennzahl*, R_{eff}). It is commonly interpreted as the average number of further infections one infected person causes, in other words, to how many other people one positively tested person spreads the disease (Robert-Koch-Institut 2020). It is clear why this number is of such high importance: Not only does it tell at a glance whether the number of active cases is increasing ($R_{\text{eff}} > 1$) or decreasing ($R_{\text{eff}} < 1$), it also captures, among other factors, the effectiveness of the policies and regulations in action (not in a causal way, though, only in terms of correlation).

One caveat of this number, as interesting as it may be, is that it is not as often updated as other figures. In Austria, for instance, this number is jointly provided by the AGES (2020) and the TU Graz, who update their figures on a weekly basis. If you are as curious and impatient like me — read through this guide to learn how to make a simple update on your own!

2 A Simplified Method

For anyone interested, the method applied by AGES (2020) and TU Graz is summarized in Richter, Schmid & Stadlober (2020), and is based on Cori et al. (2013). However, for this short application I will take a very pragmatic stance and start from the existing `EpiEstim` package, to elaborate on the inputs necessary to obtain the results (one might hold this against me, but after all, I'm not an epidemiologist, and we shall judge by the

* For the complete source codes and an online version of this guide, visit puschnig.eu.

outcome how well my method performs).

The function that provides us with the results is `estimate_R()`. It features several different methods of computing R_{eff}, but they all rely on basically two pieces of information: A time-series of daily new cases and the distribution of serial intervals.

2.1 Daily New Cases

The first (and for our purpose more important) source is a dataset on new daily cases. For Austria, this data can be downloaded from the Covid-19 Dashboard of the Minstry of Health BMGSPK (2020), or the Open Data Portal of the Ministry for Digital and Economic Affairs BMDW (2020).

2.2 Serial Interval Distribution

A serial interval describes the velocity with which a disease spreads from one person to the next. For example, if a person A contracts the disease at day 1, and spreads it to a person B four days later, the serial interval is 4 days (Robert-Koch-Institut 2020). Of course, this figure is a random variable and not a fixed parameter, and therefore needs to be estimated. However, due to a lack of data and for the sake of simplicity, we will not estimate this distribution itself, but rather take the estimates of the first two moments by Richter, Schmid, Chakeri, Maritschnik, Pfeiffer & Stadlober (2020), which yields a mean of 4.46 and a standard deviation of 2.63, and supply these to the `estimate_R()` function.

3 The Code

The code is quite short and simple. First we load the data we are going to use for our estimation, as well as the benchmark data for comparison later on.

```
# Load Epidemiological Data
Data <- read_delim("../Data/BMSGPK/Epikurve.csv",
                   ";", escape_double = FALSE,
                   col_types = cols(time = col_date(format = "%d.%m.%Y"),
                                    ‘tägliche Erkrankungen‘ = col_integer(),
                                    Timestamp = col_skip()), trim_ws = TRUE)

REffBenchmark <- read_delim("../Data/AGES/R_eff.csv",
                    ";", escape_double = FALSE,
                    col_types = cols(Datum = col_date(format = "%Y-%m-%d"),
                    R_eff = col_double()), locale = locale(decimal_mark = ",",
                            grouping_mark = "."), trim_ws = TRUE)
```

```
# Rename Columns according to the EpiEstim Package
colnames(Data) <- c("dates", "I")
colnames(REffBenchmark) <- c("Time", "REffBenchmark",
                             "REffBenchmarkLower",
                             "REffBenchmarkUpper")
```

Next, we supply the time-series of daily new cases and the serial interval mean and standard deviation to the `estimate_R()` function, to obtain the results. Note that the function returns several time series, of which we only extract the mean.

```
# Set Parameters of the Serial Interval Distribution
Mean = 4.46
StdDev = 2.63

# Estimation: EpiEstim::estimate_R()
ResultRaw <- estimate_R(Data,
                    method = "parametric_si",
                    config = make_config(method  = "parametric_si",
                                         mean_si = Mean,
                                         std_si  = StdDev))
```

Finally, we prepare the dataset and plot the results.

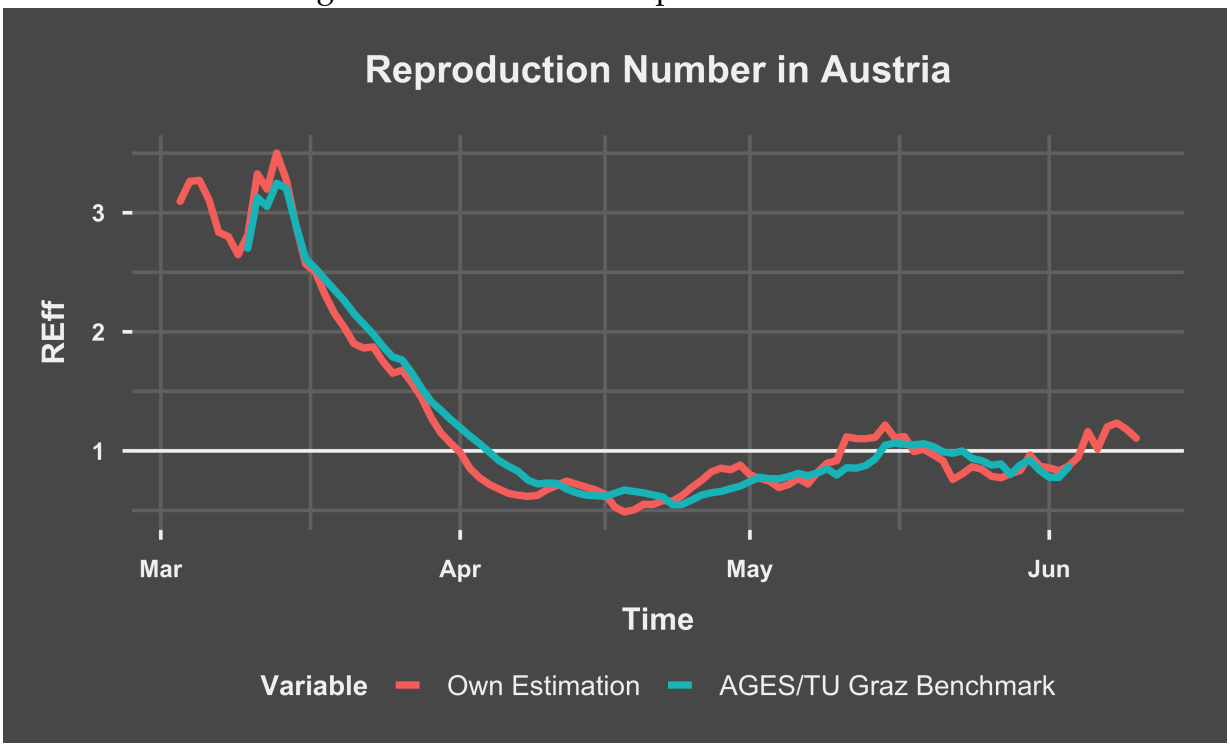
```
# Extract Estimates from Result
ResultDF <- data.frame(ResultRaw$dates[8:length(ResultRaw$dates)],
                       ResultRaw$R$`Mean(R)`)
colnames(ResultDF) <- c("Time", "REff")

# Combine with Benchmark Data and Save to Disk
ResultDF <- merge(ResultDF, REffBenchmark, all = T)
write.csv(ResultDF, file = "Results.csv")

# Reshape for Graphical Representation
ResultDFPlot <- ResultDF %>% select(Time, REff, REffBenchmark)
colnames(ResultDFPlot) <- c("Time", "Own Estimation", "AGES/TU Graz Benchmark")
ResultDFPlot <- melt(ResultDFPlot, id.vars = "Time")
colnames(ResultDFPlot) <- c("Time", "Variable", "Value")

# Make Graph and Save as PNG
REffPlot <- ggplot(ResultDFPlot, aes(x = Time, y = Value, color = Variable)) +
    geom_hline(yintercept = 1, size = 1, color = "gray95") +
    geom_line(size = 2, lineend = "round") +
    labs(title = "Reproduction Number in Austria",
         x = "Time",
```

```
         y = "REff") +
    thm_dark()
REffPlot
ggsave("Reproduction_Number.png",
       width = 25, height = 15, units = "cm", dpi = 400)
```

4 The Results

The results can be seen in Figure 1, which is automatically generated by the code. The turquoise line is the benchmark as published by AGES (2020), whereas the red line represents our own estimation.

Figure 1: Estimation of Reproduction Number



Note: Comparison of the estimation of the reproduction number R_{eff} by AGES (2020) and TU Graz and own estimation, based on data from BMGSPK (2020) and Richter, Schmid, Chakeri, Maritschnik, Pfeiffer & Stadlober (2020). Estimation was performed using the `EpiEstim` package in R.

How to explain the differences? It is possible that the authors behind the official figures estimate the parameters of the serial interval on a rolling basis — unlike the code here, where only mean and standard deviation of the latest study are used. Furthermore, there could be differences in the data. The main official database with much more

and more detailed information, the Epidemiological Reporting System (*Epidemiologisches Meldesystem*, EMS) is not publicly available and we have to make do with the data at hand. Also data revisions cannot be ruled out. Lastly, even though they refer to the `EpiEstim` package, they did not necessarily use it themselves — they might have adopted their own algorithm, or made subtle changes.

But even though the two lines do differ a bit in some places, they are fairly similar to each other, and our simplified estimation process seems to be a neat approximation of the professionals' work — and can be updated whenever a new data point is available!

References

AGES (2020), 'Epidemiologische Parameter des COVID19 Ausbruchs [Epidemiological Parameters of the COVID19 Onset in Austria]', https://www.ages.at/en/wissen-aktuell/publikationen/epidemiologische-parameter-des-covid19-ausbruchs-oesterreich-2020/.

BMDW (2020), 'COVID-19: Epidemiologische Kurve [COVID-19: Epidemiological Curve]', https://www.data.gv.at/katalog/dataset/fdff55f1-2157-49df-a351-b696ab3d471a.

BMGSPK (2020), 'Amtliches Dashboard COVID19 [Official Dashboard COVID19]', https://info.gesundheitsministerium.at/.

Cori, A., Ferguson, N. M., Fraser, C. & Cauchemez, S. (2013), 'A New Framework and Software to Estimate Time-Varying Reproduction Numbers During Epidemics', *American Journal of Epidemiology* **178**(9), 1505–1512.

Richter, L., Schmid, D., Chakeri, A., Maritschnik, S., Pfeiffer, S. & Stadlober, E. (2020), 'Epidemiologische Parameter des COVID19 Ausbruchs — Update 08.04.2020, Österreich, 2020 [Epidemiological Parameters of the COVID19 Onset — Update 08.04.2020, Austria, 2020]', https://www.ages.at/download/0/0/86de1e8a4d72a143280eca00fd5abc0a969d87c0/fileadmin/AGES2015/Wissen-Aktuell/COVID19/Update_Epidemiologische_Parameter_des_COVID19_Ausbruchs_2020-04-09.pdf.

Richter, L., Schmid, D. & Stadlober, E. (2020), 'Methodenbeschreibung für die Schätzung von epidemiologischen Parametern des COVID19 Ausbruchs, Österreich [Description of the Estimation Method of Epidemiologic Parameters of the COVID19 Onset, Austria]', https://www.ages.at/download/0/0/e03842347d92e5922e76993df9ac8e9b28635caa/fileadmin/AGES2015/Wissen-Aktuell/COVID19/Methoden_zur_Sch%C3%A4tzung_der_epi_Parameter.pdf.

Robert-Koch-Institut (2020), 'SARS-CoV-2 Steckbrief zur Coronavirus-Krankheit-2019 [SARS-CoV-2 Briefing on the Coronavirus Disease 2019]', https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Steckbrief.html.

Appendices

R-Script

```
1
2  # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
3  # # #                                                             # # #
4  # # #   Covid-19 Reproduction Factor Estimation                   # # #
5  # # #   Lukas Puschnig                                            # # #
6  # # #   June 2020                                                 # # #
7  # # #                                                             # # #
8  # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
9
10 rm(list = ls())
11
12 #   # ############################################### # #
13 #  # ##  Attach Packages                              ## #  #
14 # # ################################################ #   #
15
16 # NB: Some packages might need to be installed
17 # beforehand. You can use the following command:
18
19 #install.packages("NameOfPackage")
20
21 # For Estimation of Parameters of Interest
22 library(EpiEstim)
23
24 # For Reading CSV-Files
25 library(readr)
26
27 # For Producing Graphs
28 library(ggplot2)
29 source("Themes.R")
30
31 # To Reshape and Select Data
32 library(reshape2)
33 library(dplyr)
34
35 #   # ############################################### # #
36 #  # ##  Load Data                                    ## #  #
37 # # ################################################ #   #
38
39 # Load Epidemiological Data
40 Data <- read_delim("../Data/BMSGPK/Epikurve.csv",
```

```
                    ";", escape_double = FALSE,
                    col_types = cols(time = col_date(format = "%d.%
                       m.%Y"),
                                     `t gliche Erkrankungen` = col
                                        _integer(),
                                     Timestamp = col_skip()), trim_
                                        ws = TRUE)

REffBenchmark <- read_delim("../Data/AGES/R_eff.csv",
                     ";", escape_double = FALSE,
                     col_types = cols(Datum = col_date(format = "%Y
                        -%m-%d"),
                     R_eff = col_double()), locale = locale(decimal
                        _mark = ",",
                           grouping_mark = "."), trim_ws = TRUE)

# Rename Columns according to the EpiEstim Package
colnames(Data) <- c("dates", "I")
colnames(REffBenchmark) <- c("Time", "REffBenchmark",
                             "REffBenchmarkLower",
                             "REffBenchmarkUpper")

#   # ############################################ # #
#  # ##  Estimation                                ## #  #
# # ############################################## #   #

# Set Parameters of the Serial Interval Distribution
# Values taken from Richter et al. 2020
Mean = 4.46
StdDev = 2.63

# Estimation: EpiEstim::estimate_R()
ResultRaw <- estimate_R(Data,
                     method = "parametric_si",
                     config = make_config(method  = "parametric_si
                        ",
                                          mean_si = Mean,
                                          std_si  = StdDev))

#   # ############################################ # #
#  # ##  Save Results & Plot Graph                 ## #  #
# # ############################################## #   #

# Extract Estimates from Result
```

```
ResultDF <- data.frame(ResultRaw$dates[8:length(ResultRaw$dates)],
    ResultRaw$R$`Mean(R)`)
colnames(ResultDF) <- c("Time", "REff")

# Combine with Benchmark Data and Save to Disk
ResultDF <- merge(ResultDF, REffBenchmark, all = T)
write.csv(ResultDF, file = "Results.csv")

# Reshape for Graphical Representation
ResultDFPlot <- ResultDF %>% select(Time, REff, REffBenchmark)
colnames(ResultDFPlot) <- c("Time", "Own Estimation", "AGES/TU
   Graz Benchmark")
ResultDFPlot <- melt(ResultDFPlot, id.vars = "Time")
colnames(ResultDFPlot) <- c("Time", "Variable", "Value")

# Make Graph and Save as PNG
REffPlot <- ggplot(ResultDFPlot, aes(x = Time, y = Value, color =
   Variable)) +
    geom_hline(yintercept = 1, size = 1, color = "gray95") +
    geom_line(size = 2, lineend = "round") +
    labs(title = "Reproduction Number in Austria",
         x = "Time",
         y = "REff") +
    thm_dark()
REffPlot
ggsave("Reproduction_Number.png",
       width = 25, height = 15, units = "cm", dpi = 400)
```