**Estimating and modelling the change of R over time for common cold coronaviruses**

The paper uses data from the CDC to estimate how the reproduction number, R, for the four “common cold” coronaviruses has changed over time (from Fall 2014 to Spring 2019, in the US), and uses these estimates for R to estimate the impact of seasonality on transmission for these coronaviruses, the degree of immunity that develops to them, and the degree of cross-immunity between the two betacoronaviruses (HKU1 and OC43). Since SARS-CoV-2 is also a betacoronavirus, one might expect it to behave at least somewhat similarly to the two common cold betacoronaviruses, and for there to perhaps be some cross-immunity between SARS-CoV-2 and the other betacoronaviruses.

**Reproducing the estimates for R**

The procedure used in the paper to estimate R over time for each of these viruses has several steps:

* The incidence of infection each week with the common cold coronaviruses was estimated (up to an unknown scaling factor, relating to how likely sick people are to visit a doctor) by multiplying the weekly reports of physician visits for Influenza-Like Illness (ILI) by the weekly percentage of laboratory tests for the four coronaviruses that were positive for each of them.
* A spline-based procedure was used to interpolate daily incidence of each virus from these weekly numbers .
* From these daily incidence numbers, estimates of R for each day were obtained using a formula that looks at the incidence that day and the previous 19 days.
* The daily estimates for R were used to produce weekly estimates for R, by taking the geometric mean of the 21 daily values for the week in question and the previous and following weeks.

The first problem with reproducing these estimates is that although the data on physician visits for ILI is available from [here](https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html) (and included in the first repository above), the CDC allows access to only the *last two years* of data on positive tests for common cold coronaviruses (from [here](https://www.cdc.gov/surveillance/nrevss/coronavirus/natl-trends.html)). According to the README for the repository, “Full data used in paper is available through a data use agreement with the CDC”.

This sort of bullshit makes one wonder about the mentality of the people running the CDC. There is obviously *no reason whatever* for keeping this data under wraps. Patient confidentiality can’t be an issue, both due to the nature of the data, and to the fact that they do make it public for the last two years. Nor can it be a matter of minimizing work on the part of the CDC — it must take extra effort to keep removing older data so that only two years are available, not to mention the effort of processing data use agreements.

This CDC policy certainly resulted in extra work for the authors of this paper. They included the last two years of publicly-available data in the first repository above, along with R code that had been modified to work with only two years of data rather than five years. The results produced are of course not the same as in the paper.

Fortunately, the second repository above has a data file that in fact includes the full data that was omitted from the first repository. The data can be reformatted to the required form as follows:

dfi <- read.csv("../nCoV\_introduction-master/nrevssCDC\_ILI.csv",head=TRUE)

dfo < as.data.frame(

list(RepWeekDate=as.character(as.Date(dfi$WEEKEND),"%m/%d/%y"),

CoVHKU1=round(100\*dfi$HKU1,7),

CoVNL63=round(100\*dfi$NL63,7),

CoVOC43=round(100\*dfi$OC43,7),

CoV229E=round(100\*dfi$E229,7)))

write.table (dfo, "full-Corona4PP\_Nat.csv", sep=",",

row.names=FALSE, col.names=TRUE, quote=FALSE)

Now the remaining task is to modify the supplied R code in the first repository so it works with the full five years of data. Here are the crucial diffs needed to do this:

-# Data below shared by NREVSS team

-df.us\_cov\_national <- read.csv("Corona4PP\_Nat.csv") #2018-03-10 through 2020-02-29

+# Reconstruction of full dataset used in paper.

+# Data is for 2014-07-05 through 2019-06-29.

+df.us\_cov\_national <- read.csv("full-Corona4PP\_Nat.csv")

- Week\_start < "2018-07-01" ~ 0, # First season (and only complete season) in this dataset is 2018-19

- (Week\_start >= "2018-07-01") & (Week\_start < "2019-07-01") ~ 1,

- (Week\_start >= "2019-07-01") & (Week\_start < "2020-07-01") ~ 2)) # 2018-2019 is the last season in our data

+ Week\_start < "2014-07-06" ~ 0, # Before first season

+ Week\_start < "2015-07-05" ~ 1,

+ Week\_start < "2016-07-03" ~ 2,

+ Week\_start < "2017-07-02" ~ 3,

+ Week\_start < "2018-07-01" ~ 4,

+ Week\_start < "2019-06-30" ~ 5, # 2018-2019 is the last season, last data is for 2019-06-29

+ TRUE ~ 0)) # after last season

-for(s in 1:2){

- temp.df <- df.us\_all\_national\_withR %>% filter(season==s, epi\_week>=season\_start | epi\_week<=season\_end)

+for(s in 1:5){

+ temp.df <- df.us\_all\_national\_withR %>% filter(season==s, epi\_week>=season\_start | epi\_week<=(season\_end-(s==1))) # -(s==1) to fudge for 53 weeks in 2014

- season==1 ~ "2018-19",

- season==2 ~ "2019-20")) %>%

- mutate(season=factor(season, levels=c("1", "2"))) #Set season 1 as reference group in regression

-# Note: with this limited dataset, season 2 is incomplete. Full dataset has 5 complete seasons.

+ season==1 ~ "2014-15",

+ season==2 ~ "2015-16",

+ season==3 ~ "2016-17",

+ season==4 ~ "2017-18",

+ season==5 ~ "2018-19")) %>%

+ mutate(season=factor(season, levels=c("1", "2", "3", "4", "5"))) #Set season 1 as reference group in regression

I also added code to produce various plots and other output, some corresponding to plots in the paper or supplemental information, and some for my use in figuring out what the code does. The original code doesn’t come with an open-source license, so I won’t post my full modified source file, but some of the code that I added at the end is [here](https://radfordneal.files.wordpress.com/2020/05/plotcode.zip), and some of the plots that it produced are [here](https://radfordneal.files.wordpress.com/2020/05/data1.pdf) and [here](https://radfordneal.files.wordpress.com/2020/05/reff1.pdf).

**A digression about the R code**

I will, however, talk about one little snippet of the original program, whose behaviour is… interesting:

RDaily <- numeric()

for(u in 1:(length(week\_list)\*7)){ #Estimate for each day

sumt <- 0

for(t in u:(u+stop)){ #Look ahead starting at day u through (u+max SI)

suma <- 0

for(a in 0:(stop)){ #Calc denominator, from day t back through (t-max SI)

suma = daily\_inc[t-a,v]\*func.SI\_pull(a, serial\_int) + suma

}

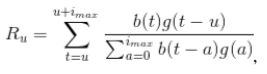
sumt = (daily\_inc[t,v]\*func.SI\_pull(t-u, serial\_int))/suma + sumt

}

RDaily[u] = sumt

}

This code computes daily estimates for R (putting them in RDaily), using the following formula from the supplemental information:



Notice that the loop for u starts at 1, the loop for t inside that starts at u, and the loop for a inside that starts at 0, and goes up to stop (*imax* in the formula), whose value is 19. For the first access to daily\_inc, the subscript t-a will be 1, the next time, it will be 0, then -1, -2, …, -18. All but the first of these index values seem to be out of bounds. But the program runs without producing an error, and produces reasonable-looking results. How can this be?

Well, R programmers will know that negative indexes are allowed, and extract all items *except* those identified by the negative subscript. So daily\_inc[-1,v] will create a long vector (1819 numbers) without error. It seems like an error should arise later, however, when this results in an attempt to store 1819 numbers into RDaily[u], which has space for only one.

But crucially, before a negative index gets used, there’s an attempt to access daily\_inc[0,v]. R programmers may also know that using a zero index is not an error in R, even though R vectors are indexed starting at 1 — zero indexes are just ignored. (I’ve previously written about [why this is a bad idea](https://radfordneal.wordpress.com/2008/09/21/design-flaws-in-r-3-%E2%80%94-zero-subscripts/).) When the subscript is a single zero index, ignoring it results in extraction of a zero-length vector.

Now, zero-length vectors also seem like the sort of thing that would lead to some sort of error later on. But R is happy (for good reason) to multiply a zero-length vector by a scalar, with the result being  another zero-length vector. The same is true for addition, so when t-a is 0, the effect is that suma in the innermost loop is set to a zero-length vector. (This is not the same as 0, which is what it was initialized to!)

Only *after* suma has been set to a zero-length vector does it get multiplied by a vector of length 1891, from accessing daily\_inc[-1,v]. R is also happy to multiply a zero-length vector by a vector of length greater than one (though this is rather dubious), with the result being a zero-length vector. So suma stays a zero-length vector for the rest of the inner loop, as daily\_inc is accessed with indexes of -1, -2, …, -18. After this loop completes, suma is used to compute a term to add to sumt, with R’s treatment of arithmetic on zero-length vectors resulting in sumt being set to a zero-length vector, and remaining a zero-length vector even when t becomes large enough that accesses with indexes less than one are no longer done.

But it still seems we should get an error! After the loop over t that computes an estimate for R at time u, this estimate is stored with the assignment RDaily[u]=sumt. Since sumt is a zero-length vector, we’d expect an error — we get one with code like x=c(10,20);x[2]=numeric() for example (note that numeric()) creates a zero-length numeric vector). Now, the code is actually extending RDaily, rather than replacing an existing element, but that doesn’t explain the lack of an error, since code like x=c(10,20);x[3]=numeric() also gives an error.

The final crucial point is that all these “out-of-bounds” accesses occur at the *beginning* of the procedure, when RDaily is itself a zero-length vector. For no clear reason, R does *not* signal an error for code like x=numeric();x[3]=numeric(), but simply leaves x as a zero-length vector. And so it is in this code, with the result that RDaily is still a zero-length vector after all operations with zero and negative out-of-bounds accesses have been done. At that point, when u is 20, a sensible value for R will be computed, and stored in RDaily[20]. R will automatically extend RDaily from length zero to length 20, with the first 19 values set to NA, and later computations will proceed as expected.

So in the end, the result computed is sensible, with the estimates for R on days for which data on 19 previous days is not available being set to NA, albeit by a mechanism that I’m pretty sure was not envisioned by the programmer. Later on, there are also out-of-bounds accesses past the end of the vector, which also result in NA values rather than errors. All these out-of-bounds references can be avoided by changing the loop over u as follows:

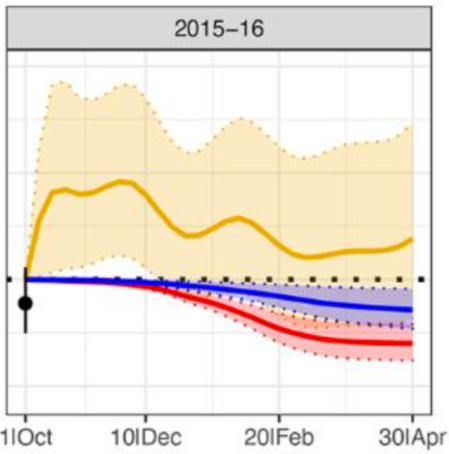
for (u in (stop+1):(length(week\_list)\*7-stop)) { #Estimate for each day

**Modeling the effects on R of immunity and seasonality**

The code produces estimates of R for each week of the cold season for all four coronaviruses, but attention focuses mainly on the two betacoronaviruses, HKU1 and OC43. A regression model is built for the R values of these viruses in terms of a seasonal effect (modelled as a spline, common to both viruses) and the effects of immunity from exposure to the same virus and of cross-immunity from exposure to the other of the two betacoronaviruses (four coefficients). The immunity effects can only be estimated up to some unknown scaling factor, with the assumption that the sum of weekly incidence numbers up to some point in the season is proportional to the fraction of the population who have been exposed to that virus.

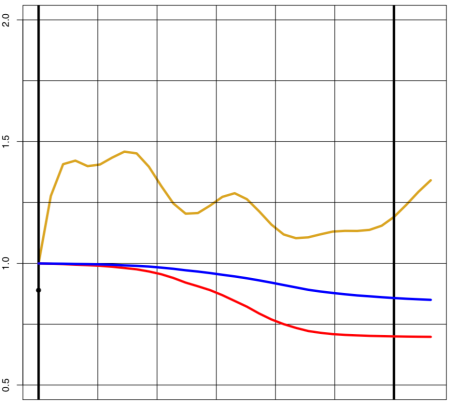
The [results I get](https://radfordneal.files.wordpress.com/2020/05/corona-coef.pdf) match the regression coefficients in Table S1 of the paper’s supplemental information, and some [additional plots](https://radfordneal.files.wordpress.com/2020/05/model1.pdf) are also as expected given results in the paper.

The seasonal and immunity effects over time are summarized in Figure 1 of the paper. Here is the part of that figure pertaining to HKU1 and the 2015-2016 cold season:



The orange curve shows the estimated multiplicative seasonal effect on R (horizonal dots are at one), the red curve is the estimated effect on R from immunity to HKU1, and the blue curve is the estimated effect from cross-immunity to OC43.

Here is my reproduction of this figure (without attempting to reproduce the error bands):



This seems to perfectly match the plot in the paper, except that the plot in the paper shows only 30 weeks, whereas the model is fit to data for 33 weeks, which is also time span of the spline used to model the seasonal effect. As one can see in my plot, after week 30 (at the bold vertical bar), the modelled seasonal effect on R rises substantially. But this feature of the model fit is not visible in the figures in the paper.

Researchers at Harvard really ought to know that they should not to do this. The rise after week 30 that is not shown in their plots is contrary to the expectation that R will decrease in summer, and is an indication that their modelling procedure may not be good. In particular, after seeing this rise at the end of the season, one might wonder whether the sharp rise in the seasonal effect on R seen at the beginning of the season is actually real, or is instead just an artifact of their spline model.

**An ODE model for betacoranavirus incidence**

The second major topic of the paper is the fitting of an ODE (Ordinary Differential Equation) model for the incidence of the two common cold betacoronavirues. The data used is the same as for the first part of the paper, but rather than directly estimate R at each time point, an underlying model of the susceptible-exposed-infected-recovered-susceptible (SEIRS) type is used, from which incidence numbers can be derived, and compared to the data.

According the paper and supplemental information, the parameters of the SEIRS model (eg, the degree of seasonal variation, and the rate at which immunity wanes) were fit by a procedure combining latin hypercube sampling (implemented in R) and Nelder-Mead optimization (implemented in Mathematica). The code for these procedures has not been released, however. Hence reproducing this part of the paper is not possible.

The second repository above does contain code to run the SEIRS model, with parameters set to values that are fixed in the code (presumably to the values found by the optimization procedure that they ran).

This SEIRS model produces values for R for each virus and time point, which can be compared to the estimates from the first part of the paper. To do this, the R code for this part of the paper needs to read the estimates for R produced by the R code for the first part. These estimates can be written out as follows:

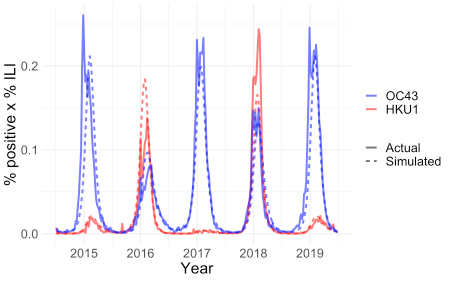
rmv <- c(1:3,nrow(Reff.CoV\_ili\_x\_pos\_pct\_SARS):(nrow(Reff.CoV\_ili\_x\_pos\_pct\_SARS)-2))

write.table (Reff.CoV\_ili\_x\_pos\_pct\_SARS[-rmv,],

"R\_ili\_x\_pos\_pct\_SARS.csv", row.names=FALSE, col.names=TRUE, quote=FALSE, sep=",")

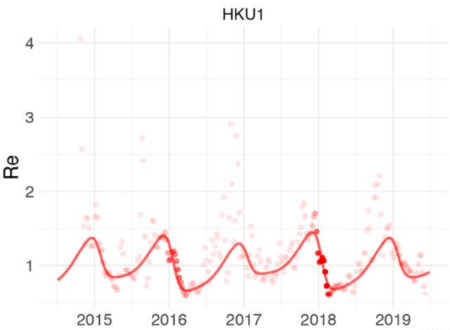
My modified version of the figuremaker.R R source file provided in the second repository above is [here](https://radfordneal.files.wordpress.com/2020/05/figuremaker.zip). It has small modifications to read the data as written out above, and to enable production of plots.

One of the plots produced by running this code is an exact reproduction of Figure 2A in the paper:

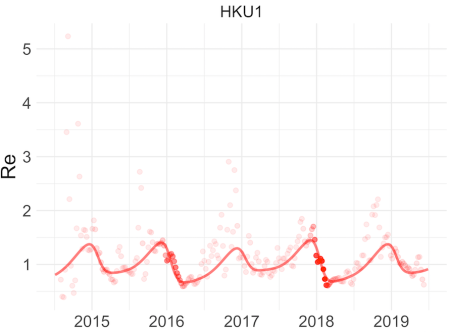


This plot shows the actual and simulated incidence of the two common cold betacoronaviruses over five cold seasons.

Running the code also produces what should be reproductions of Figures 2B and 2C, in which the values for R produced by the best-fit SEIRS model (the curve) are compared to the weekly estimates for R from the first part of the paper. But these reconstructions do not match the paper. Here is Figure 2B from the paper:

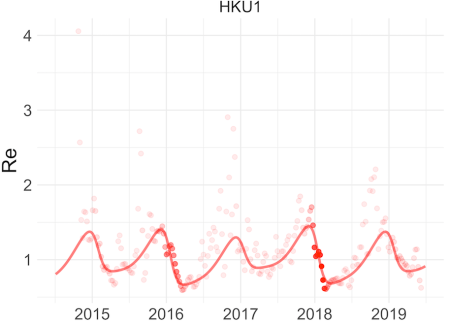


And here is what the code produces:



The curves are the same (apart from vertical scale), but the figure in the paper is missing the first 12 estimates for R, and the first few estimates that follow those are noticeably different.

I found that an exact reproduction of Figures 2B and 2C in the paper *can* be obtained by re-running the code for estimating R using a data file in which the first eleven estimates for R have been deleted, and in which the code has been changed to say that the first season starts on 2014-09-28 (rather than 2014-07-06). Here is the perfectly-matching result:



Perhaps Figures 2B and 2C in the paper were inadvertently produced using a file of R estimates created using preliminary code that for some reason treated the start of the season differently than the final version of the code. It’s unfortunate that the published figure is somewhat misleading regarding the match between the R estimates from the first part of the paper and the R estimates from the SEIRS model, since this match is significantly worse for the missing data points than for the others.

**Projecting the future course of SARS-CoV-2 infection**

The final part of the paper extends the ODE model for the two common cold betacoronaviruses to include SARS-CoV-2, considering various possibilities for the characteristics of SARS-CoV-2, such as degree of seasonality and duration of immunity, as well as various interventions such as social distancing.