JBF $R_e(t)$ for NL

Jan van Rongen

2021-07-26

Explanation

JBF emulates the RIVM data quite well, Can it do better? And can we clean up the messy code and make a proper error estimate?

Well, yes and no.

The JBF method

It follows two steps: first smooth the input data. The smoother happens to create only positive values, so if X is the current data and Y is the data of the Serial interval, then X/Y is the $R_e(t)$. It happens to work best with a Gaussian smoother with a span of 11 and with a non stochastic serial interval of 4.

The error margin is calculated using some non-theoretical hocus pocus.

JBF is theoretically flawed.

But it is (unfortunately) a flawed method. After smoothing we have an estimate of a trend and that estimate has an error margin. We can assume that the error is more or less Gaussian, so we have a time series of normal distributions $N(\mu(t), \sigma(t))$ and $R_e(t-4) = N(\mu(t), \sigma(t))/N(\mu(t-4), \sigma(t-4))$ for all relevant t. The $\mu(t)$ is estimated as the smoothed value at day t, and here comes the problem. We divide two random variables X and Y, but the $E(X/Y) = E(X).E(1/Y)$ if we assume them independent, but $E(1/Y) \neq 1/E(Y)$ in general.

Because JBF was such a quick hack I missed this point completely (even though I knew it very well as the central problem in a previous project).

The solution

To solve this problem we can use a Taylor expansion of (X/Y) around $\mu(Y)$, but it is easier to use a simulation. See later in this report.

Load RIVM Data

```
rivm_ts <- make_cases(repro_rivm[, c(1,3)])
FROM= as.Date("2020-08-01"); TO= as.Date("2021-06-25")
rivm_ts<- rivm_ts[FROM <= rivm_ts$date & rivm_ts$date < TO,]
cases_all<- make_all_cases()
```

Try other smoothing methods

As an example we use a Golay filter.

```
for (m in 8:15)  {
  JBF_Rt <- Rt_JBF(cases_all, SMOOTH_DATA=m, SMOOTH_Method="Golay")
  JBF_ts <- make_cases( JBF_Rt[, 1:2])
  JBF_ts<- JBF_ts[FROM <= JBF_ts$date & JBF_ts$date < TO,]
  cat("\nResemblance:", m, rev_rmse(JBF_ts$cases, rivm_ts$cases), "%")
}
```

```
##
## Resemblance: 8 92.81 %
## Resemblance: 9 95.36 %
## Resemblance: 10 95.36 %
## Resemblance: 11 97.39 %
## Resemblance: 12 97.39 %
## Resemblance: 13 96.48 %
## Resemblance: 14 96.48 %
## Resemblance: 15 94.84 %
```

We have tried various other filters, but the Gaussian remains the closest with 98.51% and a span of 11, followed by the moving average centered with a span of 7 days and a score of 97.49%. Third and very close is the Golay filter with a span again of 11 days and a score of 97.39%.

We have also tried to change the dates of DON and DPL records in the source data (as RIVM says it's doing), but that has almost no effect.

improving the error estimates

The initial code of JBF was just a messy hack. It worked, but does it mean anything? Lets find out some details

Uncertainties after a kernel approximation

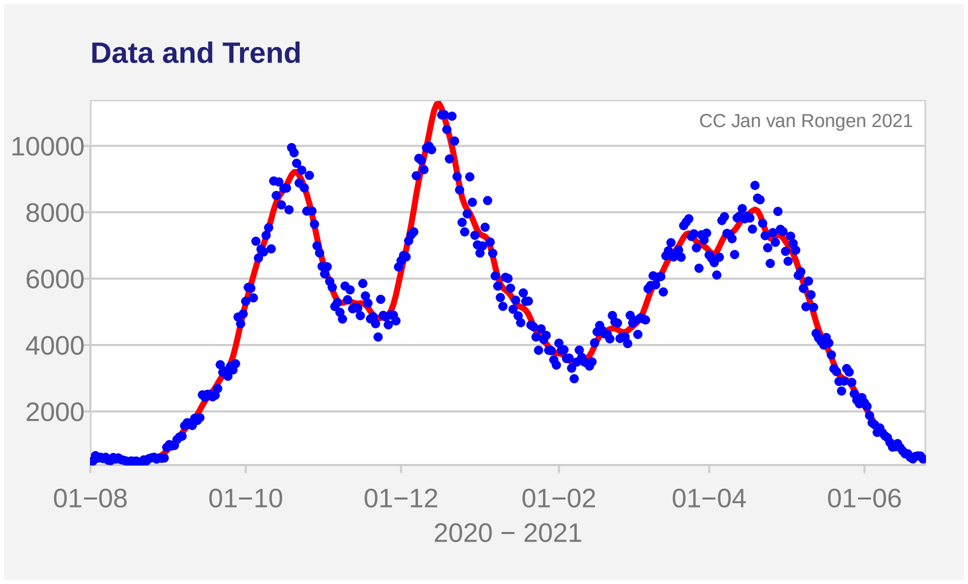
However we smooth, we calculate a trend in a time series. We fit the best possible trend (in our class of models), but how certain can we be about that trend?

The first objection is that time series are always autocorrelated, and we know to correct the error term for that – so I looked into it. But our smoothing threw most of it out.

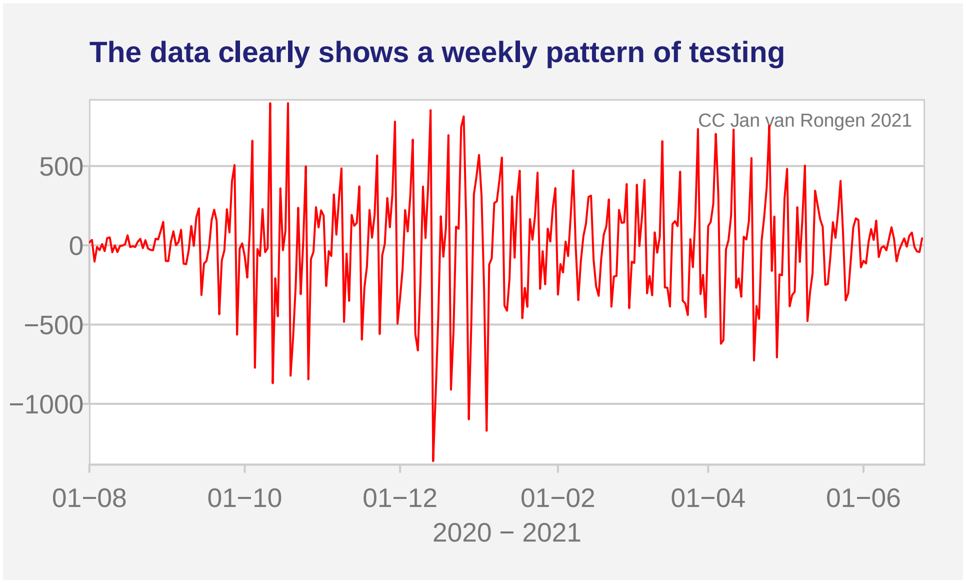
But still, the error terms or $data - trend$ are not the white noise they are supposed to be.

With t the date variable, we start with the model that $Y(t) = T(t) + E(t)$ wher T is the trens, E is the error (noise) and Y are the daily observations. This is a familiar regression model and we will immediately discover that $E(t)$ is not so randomly distributed.

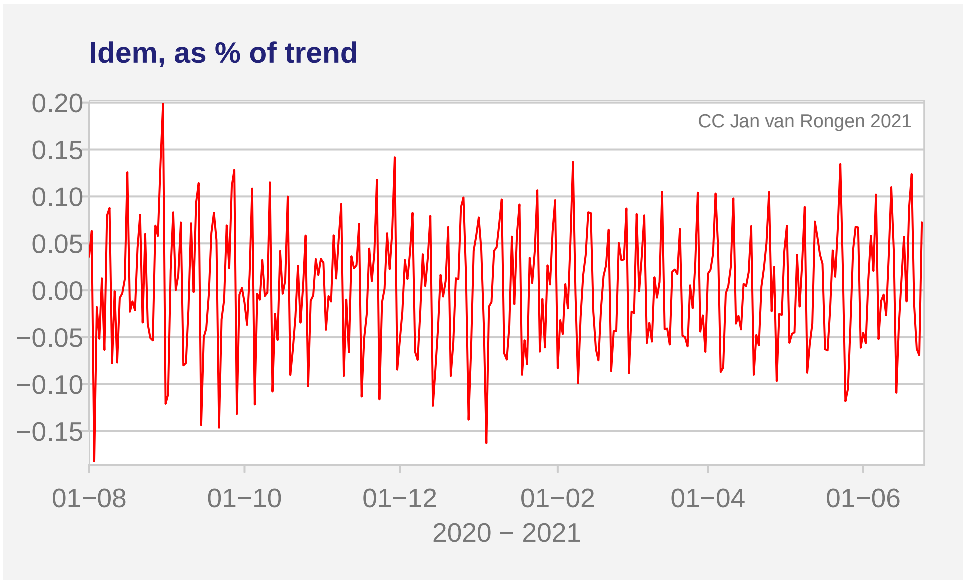
```
TO= as.Date("2021-06-25")
# change to mid july to see how absurd the Dutch situation was end june

# data
cases_all<- make_all_cases()
trend_all<- smoothed(cases_all, 11)
cases_ts<- cases_all[FROM <= cases_all$date & cases_all$date < TO, ]
trend_ts<- trend_all[FROM <= trend_all$date & trend_all$date < TO, ]

pretty_date(trend_ts, lwd=3, start=FROM, end=TO, main="Data and Trend")
pretty_date(add=T, type="p", CEX=0.9, cases_ts, kleur=2)
```



```
a<- trend_ts; a$cases<- a$cases- cases_ts$cases
pretty_date(a, start=FROM, end=TO, main="The data clearly shows a weekly pattern of test
```



```
a$cases<- ifelse(trend_ts$cases>0, a$cases/trend_ts$cases, 0)
pretty_date(a, start=FROM, end=TO, main="Idem, as % of trend")
```



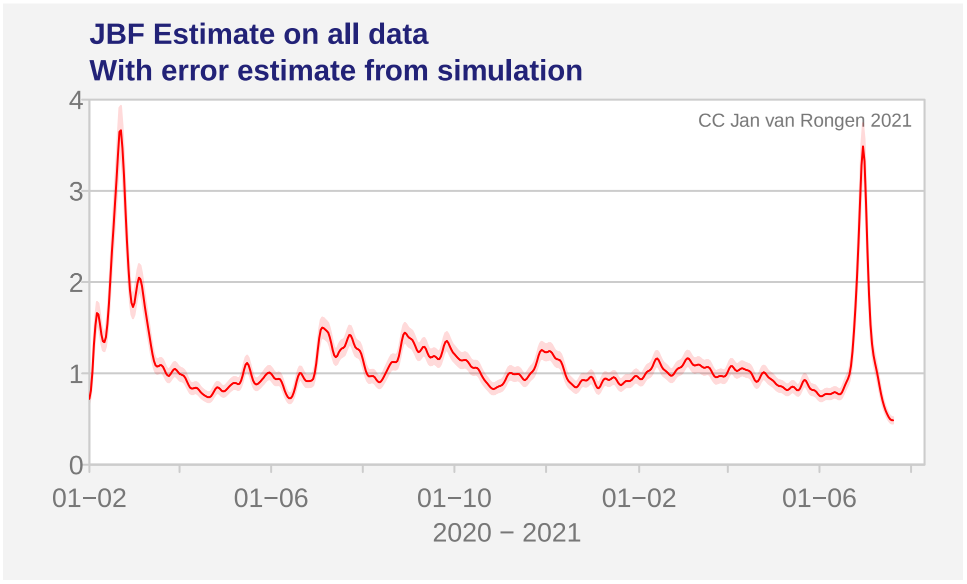
```
cat("\nStandard error as % of the mean:", mean(abs(a$cases)/1.96))
```

```
##
## Standard error as % of the mean: 0.02631366
```

So we can conclude that locally the trend might be deviated by 5%, so with a std of 2.5%. So when we want to know the uncertainties, we might sample from N(trend(t), 0.0263*trend(t)) in a simulation.

That is exactly what we do in the next piece. We also add prior bounds to the simulated $R_e(t)$, to remove real outliers.

```
Rt_JBF_2 <- function(data_set)
  {
  N= 7500

  LO<- 0.4
  HI<- 10

  trend_all <- smoothed(data_set, 11)
  M<- matrix(0,nrow=nrow(trend_all), ncol=N)
  for ( m in 1:nrow(trend_all))
    M[m,] <- rnorm(N, trend_all[m,2], 0.0263*trend_all[m,2])

  M1<- head(M, -4)
  M<- ifelse (M1 >0, tail(M, -4)/M1, 0)

  R<-  apply(M, 1, function(x){
    x<- x[x<HI & x > LO]
    mean(x)
  })
  STD<- apply(M, 1, function(x){
    x<- x[x<HI & x > LO]
    sd(x)
  })

  result<- data.frame(date= head(trend_all$date, -4),
                      R=R, lo= R-1.96*STD, hi=R+1.96*STD)
  return(result)
}

## look at it
result<- Rt_JBF_2(cases_all)
pretty_date(start= as.Date("2020-02-01"),ylim=c(0,4), result[, -2], type="a",
            main="JBF Estimate on all data\nWith error estimate from simulation")
pretty_date(add = TRUE,  result[, 1:2])
```



How good is it?

Almost as good. Without the prior in the above routine it is 2 points lower.

```
##
## Resemblance: 98.49 %
```