COVID 19 US Total Case and Death Estimates & Forecasts

Viole, Fred

April 15, 2020

1 Install latest NNS (0.5.1)

```
# Install NNS >= 0.5.1, uncomment next line if NNS >= 5.1 not installed already
# library(devtools); install_github('OVVO-Financial/NNS', ref = "NNS-Beta-Version")
library(NNS)
library(data.table)
library(xtable)
```

2 Read and convert data

NY Times covid 19 data repository: `https://github.com/nytimes/covid-19-data`

```
data <- read.csv("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv",
                  header = TRUE, sep = ",")
data <- data.table(data)
# US Totals
totals <- data[, lapply(.SD, sum), .SDcols = c("cases","deaths"), by = date]
# NY NJ Specific
ny_nj_totals <- data[state%in%c("New York", "New Jersey"), lapply(.SD, sum),
                      .SDcols = c("cases","deaths"), by = date]
```

3 Create growth rate for total cases

In this step, we are doing a univariate time-series estimation of the observed growth rate in **total cases**. We use **NNS.ARMA** on cross-validated parameters via the **NNS.ARMA.optim** function.

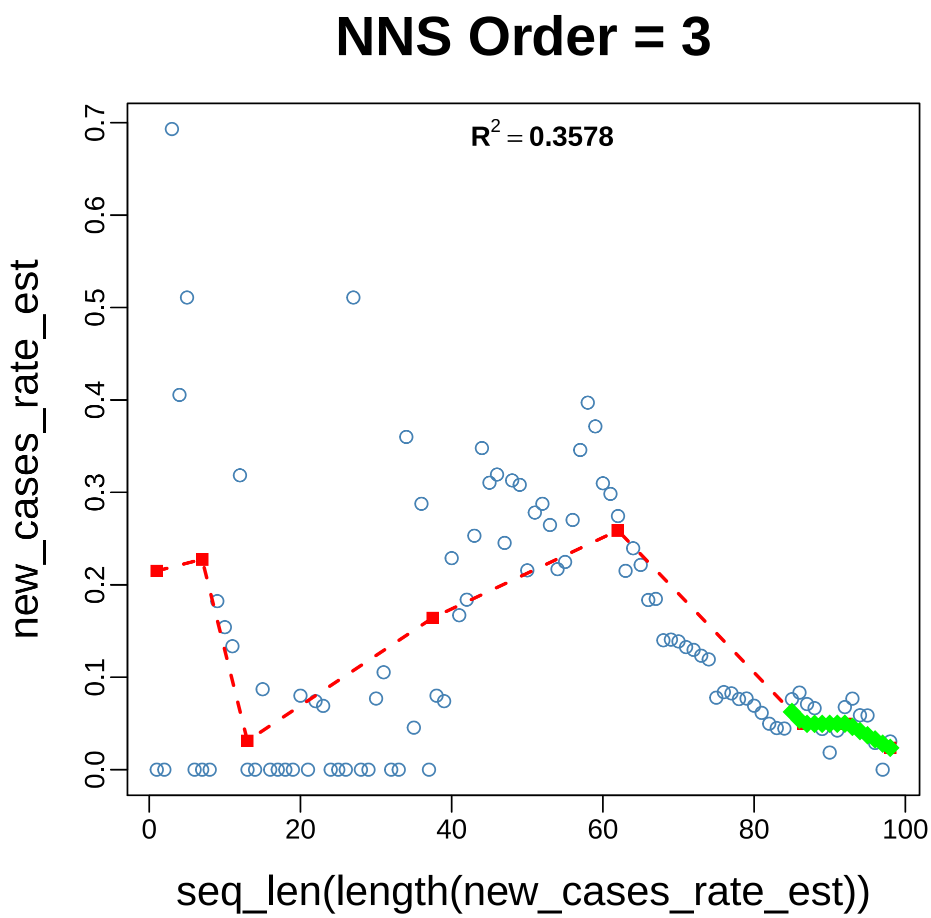
```
new_cases_rate <- log(totals$cases/shift(totals$cases,1))
new_cases_rate <- new_cases_rate[-1]
l <- length(new_cases_rate)
a <- NNS.seas(new_cases_rate, modulo = 7)
b <- NNS.ARMA.optim(new_cases_rate, training.set = l-14, seasonal.factor = a$periods,
                    print.trace = FALSE)
new_cases_rate_est <- NNS.ARMA(new_cases_rate, h = 14, seasonal.factor = b$periods,
                                weights = b$weights) + b$bias.shift
# NY NJ Specific
ny_nj_new_cases_rate <- log(ny_nj_totals$cases/shift(ny_nj_totals$cases,1))
ny_nj_new_cases_rate <- ny_nj_new_cases_rate[-1]
l <- length(ny_nj_new_cases_rate)
a <- NNS.seas(ny_nj_new_cases_rate, modulo = 7)
b <- NNS.ARMA.optim(ny_nj_new_cases_rate, training.set = l-14, seasonal.factor = a$periods,
                    print.trace = FALSE)
```

```
# Test 60 days out to find 0 intercept
ny_nj_new_cases_rate_est <- NNS.ARMA(ny_nj_new_cases_rate, h = 14, seasonal.factor = b$periods,
                                weights = b$weights) + b$bias.shift
```

3.1 Smooth growth rate point estimates and generate CI

With this new **extended growth rate** time series, we smooth the point estimates provided and generate a 0.95 confidence interval using $\pm 2\sigma$ of the observed growth rates. The image below is the regression of the **extended growth rate**. 0 is the critical point to achieve.

```
new_cases_rate_est_raw <- c(new_cases_rate, new_cases_rate_est)
new_cases_rate_est <- c(new_cases_rate, new_cases_rate_est)
new_cases_rate_est <- NNS.reg(seq_len(length(new_cases_rate_est)), new_cases_rate_est,
                              point.est = tail(1:length(new_cases_rate_est),14))$Point.est
new_cases_rate_est <- new_cases_rate_est + 1
# NY NJ Specific
ny_nj_new_cases_rate_est_raw <- c(ny_nj_new_cases_rate, ny_nj_new_cases_rate_est)
ny_nj_new_cases_rate_est <- c(ny_nj_new_cases_rate, ny_nj_new_cases_rate_est)
ny_nj_new_cases_rate_est <- NNS.reg(seq_len(length(ny_nj_new_cases_rate_est)),
                                    ny_nj_new_cases_rate_est,
                              point.est = tail(1:length(ny_nj_new_cases_rate_est), 14),
                              plot = FALSE)$Point.est
# US Lower CI Case rate
new_cases_rate_est_lower <- pmax(0, new_cases_rate_est_raw - 2 * sd(new_cases_rate_est_raw))
new_cases_rate_est_lower <- c(new_cases_rate, new_cases_rate_est_lower)
new_cases_rate_est_lower <- NNS.reg(seq_len(length(new_cases_rate_est_lower)),
                                    new_cases_rate_est_lower,
                              point.est = tail(1:length(new_cases_rate_est_lower), 14),
                              plot = FALSE)$Point.est
new_cases_rate_est_lower <- new_cases_rate_est_lower + 1
# US Upper CI Case rate
new_cases_rate_est_upper <- new_cases_rate_est_raw + 2 * sd(new_cases_rate_est_raw)
new_cases_rate_est_upper <- c(new_cases_rate, new_cases_rate_est_upper)
new_cases_rate_est_upper <- NNS.reg(seq_len(length(new_cases_rate_est_upper)),
                                    new_cases_rate_est_upper,
                              point.est = tail(1:length(new_cases_rate_est_upper), 14),
                              plot = FALSE)$Point.est
new_cases_rate_est_upper <- new_cases_rate_est_upper + 1
```



4 Apply growth rates to estimate total cases

Using the growth rates from the above step, we can now apply them to the last known value of **total cases** and extrapolate. We do this for the **lower growth rate** as well as the **upper growth rate**.

```
cases_est <- numeric()
for(i in 1:length(new_cases_rate_est)){
   if(i >1){
     cases_est[i] <- cases_est[i-1]*new_cases_rate_est[i]
     } else {
     cases_est[i] <- tail(totals$cases,1)*new_cases_rate_est[i]
     }
 }
lower_cases_est <- numeric()
for(i in 1:length(new_cases_rate_est_lower)){
   if(i >1){
     lower_cases_est[i] <- lower_cases_est[i-1]*new_cases_rate_est_lower[i]
     } else {
     lower_cases_est[i] <- tail(totals$cases,1)*new_cases_rate_est_lower[i]
     }
 }
```

```
upper_cases_est <- numeric()
for(i in 1:length(new_cases_rate_est_upper)){
   if(i >1){
     upper_cases_est[i] <- cases_est[i-1]*new_cases_rate_est_upper[i]
     } else {
     upper_cases_est[i] <- tail(totals$cases,1)*new_cases_rate_est_upper[i]
     }
 }
```

5 Assign death rate and CI

We will use the last observed **death rate**, and apply $\pm 2\sigma$ from the observed death rates to determine a **lower death rate** and an **upper death rate**.

```
death_rate <- totals$deaths/totals$cases
death_rate <- death_rate[-1]
death_rate_est <- tail(death_rate,1)
lower_death_rate_est <- max(0, death_rate_est - 2*sd(death_rate))
upper_death_rate_est <- death_rate_est + 2*sd(death_rate)
```

6 Estimate total deaths from estimated total cases

Using the death rates from the above step, we can now apply them to the estimated total cases in section 4, creating a lower and upper interval for **estimated total deaths**.

```
new_deaths <- c(totals$deaths,death_rate_est*cases_est)
forecast_points <- tail(1:length(new_deaths), 14)
deaths_est <- NNS.reg(seq_len(length(new_deaths)), new_deaths,
                       point.est = forecast_points)$Point.est
new_deaths_lower <- c(totals$deaths,lower_death_rate_est*cases_est)
lower_deaths_est <- NNS.reg(seq_len(length(new_deaths_lower)), new_deaths_lower,
                             point.est = forecast_points)$Point.est
lower_deaths_est <- pmax(lower_deaths_est, tail(totals$deaths,1))
new_deaths_upper <- c(totals$deaths,upper_death_rate_est*cases_est)
upper_deaths_est <- NNS.reg(seq_len(length(new_deaths_upper)), new_deaths_upper,
                             point.est = forecast_points)$Point.est
date <- seq(as.Date(tail(totals$date,1)), by = "day", length.out = 15)[-1]
estimates <- cbind.data.frame(as.character(date),
                               cases_est, lower_cases_est, upper_cases_est,
                               deaths_est, lower_deaths_est, upper_deaths_est)
colnames(estimates)[1] <- "date"
```

7 Results

7.1 Last known values:

	date	cases	deaths
1	2020-04-09	463684	16674
2	2020-04-10	496912	18712
3	2020-04-11	528395	20575
4	2020-04-12	555325	22056
5	2020-04-13	580878	23607
6	2020-04-14	607318	26081

7.2 Estimates:

	date	cases_est	lower_cases_est	upper_cases_est	deaths_est	lower_deaths_est	upper_deaths_est
1	2020-04-15	645261	608070	820870	26898	26081	40970
2	2020-04-16	680069	608449	866400	28716	26081	45709
3	2020-04-17	713779	608454	907073	30534	26081	50449
4	2020-04-18	749176	608454	950715	32352	26081	55189
5	2020-04-19	786344	608454	996544	34170	26081	59928
6	2020-04-20	825374	608454	1044602	35988	26081	64668
7	2020-04-21	866358	608454	1094998	37807	26081	69408
8	2020-04-22	909397	608454	1147848	39625	26081	74147
9	2020-04-23	951508	608454	1203271	41202	26081	77616
10	2020-04-24	991244	608454	1257316	42530	26081	79776
11	2020-04-25	1028135	608454	1308080	43859	26081	81936
12	2020-04-26	1061725	608454	1354954	45225	26081	84405
13	2020-04-27	1091588	608454	1397355	46604	26081	86978
14	2020-04-28	1117329	608454	1434738	47983	26081	89552

7.3 Yesterday's Forecasted and Actual Values:

To compare yesterday's forecasted versus the actual reported values (*plus any revisions to existing data*), we can repeat all of the previous steps on the modified dataset by removing the last entry and creating the object **yesterday**.

```
yesterday <- totals[-.N,]
print(xtable(tail(yesterday)))
```

	date	cases	deaths
1	2020-04-08	429319	14803
2	2020-04-09	463684	16674
3	2020-04-10	496912	18712
4	2020-04-11	528395	20575
5	2020-04-12	555325	22056
6	2020-04-13	580878	23607

```
yesterday_new_cases_rate <- log(yesterday$cases/shift(yesterday$cases,1))
yesterday_new_cases_rate <- yesterday_new_cases_rate[-1]
l <- length(yesterday_new_cases_rate)
a <- NNS.seas(yesterday_new_cases_rate, modulo = 7)
b <- NNS.ARMA.optim(yesterday_new_cases_rate, training.set = l-14, seasonal.factor = a$periods,
```

```r
                      print.trace = FALSE)
yesterday_new_cases_rate_est <- NNS.ARMA(yesterday_new_cases_rate, h = 14,
                                          seasonal.factor = b$periods,
                                weights = b$weights) + b$bias.shift
yesterday_new_cases_rate_est <- c(new_cases_rate, yesterday_new_cases_rate_est)
yesterday_new_cases_rate_est <- NNS.reg(seq_len(length(yesterday_new_cases_rate_est)),
                                         yesterday_new_cases_rate_est,
                           point.est = tail(1:length(yesterday_new_cases_rate_est),14))$Point.est
yesterday_new_cases_rate_est <- yesterday_new_cases_rate_est + 1
yesterday_cases_est <- numeric()
for(i in 1:length(yesterday_new_cases_rate_est)){
   if(i >1){
     yesterday_cases_est[i] <- yesterday_cases_est[i-1]*yesterday_new_cases_rate_est[i]
     } else {
     yesterday_cases_est[i] <- tail(yesterday$cases,1)*yesterday_new_cases_rate_est[i]
     }
 }
yesterday_death_rate <- yesterday$deaths/yesterday$cases
yesterday_death_rate <- yesterday_death_rate[-1]
yesterday_death_rate_est <- tail(yesterday_death_rate,1)
yesterday_new_deaths <- c(yesterday$deaths, yesterday_death_rate_est*yesterday_cases_est)
yesterday_forecast_points <- tail(1:length(yesterday_new_deaths), 14)
yesterday_deaths_est <- NNS.reg(seq_len(length(yesterday_new_deaths)), yesterday_new_deaths,
                       point.est = yesterday_forecast_points)$Point.est
yesterday_date <- seq(as.Date(tail(yesterday$date,1)), by = "day", length.out = 15)[-1]
yesterday_estimates <- cbind.data.frame(as.character(yesterday_date),
                               yesterday_cases_est,
                               yesterday_deaths_est)
colnames(yesterday_estimates)[1] <- "date"
```

7.3.1 Yesterday's Forecast:

```r
head(yesterday_estimates, 1)
```

```
        date yesterday_cases_est yesterday_deaths_est
1 2020-04-14            620221.6             25123.35
```

7.3.2 Yesterday's Actual:

```r
tail(totals, 1)
```

```
         date  cases deaths
1: 2020-04-14 607318  26081
```

7.3.3 Yesterday's Forecast Percentage Error:

Yesterday NNS Total Cases Percentage Error:	2.1247%
Yesterday NNS Total Deaths Percentage Error:	-3.6718%

7.3.4 Baseline nls Estimate:

We will use the nls function in R to estimate a nonlinear model and compare to **NNS** on yesterday's values. We will use the following nonlinear parametric form:

$$y = \alpha e^{\beta x} + \theta$$

```
theta.0 <- 0 #min(yesterday$cases) * 0.5
# Estimate the rest parameters using a linear model
model.0 <- lm(log(yesterday$cases - theta.0) ~ seq_len(length(yesterday$cases)))
alpha.0 <- exp(coef(model.0)[1])
beta.0 <- coef(model.0)[2]
# Starting parameters
start <- list(alpha = alpha.0, beta = beta.0, theta = theta.0)
model <- nls(yesterday$cases ~ alpha * exp(beta * seq_len(length(yesterday$cases))) + theta ,
              data = yesterday, start = start)
coef(model)
```

```
       alpha          beta         theta
1.913589e+02  9.701778e-02 -1.029787e+04
```

Now forecast the nls model for the next period's **total cases** and apply the same **death rate** estimate used in the **NNS** model.

```
nls_cases_forecast <- coef(model)[1]*exp(coef(model)[2]*(length(yesterday$cases)+1)) + coef(model)[3]
as.numeric(nls_cases_forecast)
```

```
[1] 719601.3
```

```
nls_deaths_forecast <- nls_cases_forecast * yesterday_death_rate_est
as.numeric(nls_deaths_forecast)
```

```
[1] 29244.74
```

Yesterday nls Total Cases Percentage Error:	18.4884%
Yesterday nls Total Deaths Percentage Error:	12.1305%

8 Best Guess When NY/NJ Total Cases Peak

Looking at the `NY/NJ new case rate estimate`, we see it does equal 0. This curve flattening occurs on the following date:

```
ny_nj_new_cases_rate_est
```

```
 [1] 0.0262911119 0.0219153785 0.0175396452 0.0131639119 0.0087881786
 [6] 0.0044124452 0.0000367119 0.0000000000 0.0000000000 0.0000000000
[11] 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

```
as.Date(tail(ny_nj_totals$date,1)) + which(ny_nj_new_cases_rate_est==0)[1]
```

```
[1] "2020-04-22"
```

9 Best Guess When US Total Cases Peak

Looking at the `new case rate est`, there's nothing on the immediate horizon signalling no growth (a rate equal to 1).

```
new_cases_rate_est
```

```
 [1] 1.062476 1.053944 1.049569 1.049591 1.049612 1.049634 1.049656 1.049678
 [9] 1.046307 1.041761 1.037216 1.032671 1.028126 1.023581
```

We solve for the number of days by dividing the last `new case rate est - 1` by its last gradient.[1]

```
as.Date(tail(totals$date, 1)) + length(new_cases_rate_est) +
   round(abs((tail(new_cases_rate_est, 1) - 1) / tail(diff(new_cases_rate_est - 1), 1)))
```

```
[1] "2020-05-03"
```

[1] The NY/NJ case rate was never applied to estimate a total case estimate, only to estimate the growth rate properties, hence no need to subtract 1.