

Covid 19 US Total Case and Death Estimates

Viola, Fred
fviola@fordham.edu

April 9, 2020

1 Install latest NNS (0.5.1)

```
# Install NNS >= 0.5.1
require(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)
totals <- data[, 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 = 1-14, 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
```

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.

```
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(1:length(new_cases_rate_est), new_cases_rate_est,
```

```

        point.est = tail(1:length(new_cases_rate_est),14))$Point.est
new_cases_rate_est <- new_cases_rate_est + 1
# Lower CI Case rate
new_cases_rate_est_lower <- pmax(0, new_cases_rate_est_raw - 2 * sd(new_cases_rate_est_raw))
new_cases_rate_est_lower <- c(new_cases_rate, new_cases_rate_est_lower)
new_cases_rate_est_lower <- NNS.reg(1:length(new_cases_rate_est_lower), new_cases_rate_est_lower,
        point.est = tail(1:length(new_cases_rate_est_lower), 14))$Point.est
new_cases_rate_est_lower <- new_cases_rate_est_lower + 1
# 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(1:length(new_cases_rate_est_upper), new_cases_rate_est_upper,
        point.est = tail(1:length(new_cases_rate_est_upper), 14))$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$deaths,death_rate_est*cases_est)
forecast_points <- tail(1:length(new_deaths), 14)
deaths_est <- NNS.reg(1:length(new_deaths), new_deaths,
                     point.est = forecast_points)$Point.est
new_deaths_lower <- c(totals$deaths,lower_death_rate_est*cases_est)
lower_deaths_est <- NNS.reg(1:length(new_deaths_lower), new_deaths_lower,
                           point.est = forecast_points)$Point.est
new_deaths_upper <- c(totals$deaths,upper_death_rate_est*cases_est)
upper_deaths_est <- NNS.reg(1:length(new_deaths_upper), new_deaths_upper,
                           point.est = forecast_points)$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:

```
print(xtable(tail(totals)))
```

| | date | cases | deaths |
|---|------------|--------|--------|
| 1 | 2020-04-03 | 276501 | 7121 |
| 2 | 2020-04-04 | 311536 | 8499 |
| 3 | 2020-04-05 | 336774 | 9655 |
| 4 | 2020-04-06 | 366238 | 10959 |
| 5 | 2020-04-07 | 397754 | 12956 |
| 6 | 2020-04-08 | 429319 | 14827 |

7.2 Estimates:

```
print(xtable(estimates, digits = 0))
```

| | date | cases_est | lower_cases_est | upper_cases_est | deaths_est | lower_deaths_est | upper_deaths_est |
|----|------------|-----------|-----------------|-----------------|------------|------------------|------------------|
| 1 | 2020-04-09 | 478397 | 432688 | 601356 | 17136 | 2546 | 29681 |
| 2 | 2020-04-10 | 529355 | 435822 | 665654 | 19158 | 2572 | 34542 |
| 3 | 2020-04-11 | 582474 | 438716 | 731638 | 21179 | 2591 | 39403 |
| 4 | 2020-04-12 | 640371 | 441364 | 800885 | 23201 | 2119 | 44265 |
| 5 | 2020-04-13 | 703416 | 443762 | 880276 | 25222 | 1648 | 49126 |
| 6 | 2020-04-14 | 772001 | 445905 | 966704 | 27244 | 1176 | 53987 |
| 7 | 2020-04-15 | 846541 | 447789 | 1060700 | 29265 | 704 | 58848 |
| 8 | 2020-04-16 | 927476 | 449411 | 1162831 | 31287 | 233 | 63709 |
| 9 | 2020-04-17 | 1015269 | 450767 | 1273694 | 34279 | 0 | 70204 |
| 10 | 2020-04-18 | 1110410 | 451855 | 1393919 | 38273 | 0 | 78383 |
| 11 | 2020-04-19 | 1213415 | 452674 | 1524171 | 42266 | 0 | 86561 |
| 12 | 2020-04-20 | 1324824 | 453220 | 1665150 | 46204 | 0 | 94625 |
| 13 | 2020-04-21 | 1445206 | 453494 | 1817590 | 50122 | 0 | 102651 |
| 14 | 2020-04-22 | 1575157 | 453494 | 1982263 | 54041 | 0 | 110676 |

7.3 Yesterday's Forecasted and Actual Values:

To compare yesterday's forecasted versus the actual reported values, we can repeat all of the previous steps on the modified dataset by removing the last entry and creating the object **yesterday**.

```
yesterday <- totals[1:N,]  
print(xtable(tail(yesterday)))
```

| | date | cases | deaths |
|---|------------|--------|--------|
| 1 | 2020-04-02 | 244423 | 6232 |
| 2 | 2020-04-03 | 276501 | 7121 |
| 3 | 2020-04-04 | 311536 | 8499 |
| 4 | 2020-04-05 | 336774 | 9655 |
| 5 | 2020-04-06 | 366238 | 10959 |
| 6 | 2020-04-07 | 397754 | 12956 |

```
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 = 1-14, seasonal.factor = a$periods,
                    print.trace = FALSE)
yesterday_new_cases_rate_est <- NNS.ARMA(yesterday_new_cases_rate, h = 14,
                                         seasonal.factor = b$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(1: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$deaths, yesterday_death_rate_est*yesterday_cases_est)
yesterday_forecast_points <- tail(1:length(yesterday_new_deaths), 14)
yesterday_deaths_est <- NNS.reg(1:length(yesterday_new_deaths), yesterday_new_deaths,
                              point.est = yesterday_forecast_points)$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:

```

head(yesterday_estimates, 1)

      date yesterday_cases_est yesterday_deaths_est
1 2020-04-08          445099.4          14878.73

```

7.3.2 Yesterday's Actual:

```

tail(totals,1)

      date  cases deaths
1: 2020-04-08 429319  14827

```

7.3.3 Yesterday's Forecast Percentage Error:

| | 1 | 2 |
|---|--|---------|
| 1 | Yesterday Total Cases Percentage Error: | 3.6757% |
| 2 | Yesterday Total Deaths Percentage Error: | 0.3489% |