UCI limitations

Brew, Chaccour, García-Basteiro, others?

2020-03-20

```
# ## Read first: Implicit non-exclusivity agreement
#
# - I (Joe) am working right now only under conditions of non-exclusivity.
# - This means that if a collaborator (you) is working with me, you understand that I may publish, shar
# - The condition goes both ways: the collaborator (you) also can share publicly anything I do with you
# - The reason for this non-exclusivity pact is principle: the COVID-19 crisis requires extremely fast
# - This non-exclusivity principile applies to everything we work on together, unless explicitly stated
# - If you don't like non-exclusivity (that is, you want me to "keep secret" any ideas we generate or w
# - Just to be clear: If you are working with me, this means all of our code / data is public and can b
```

This document

This document seeks to answer the question: how soon until Spain runs out of ICU beds?

Click "Code" to see code snippets.

The question?

How soon until Spain runs out of UCI beds?

Methods

We define some parameters at the state level, generate a simple log-linear predictive model for daily UCI admissions weighted by days ago, and compare the capacity "ceiling" with prediction.

Here are the parameters:

```
  beds_public = 3508,
  beds_private = 896,
  normal_occupancy = 50,
  need_hospitalization = 15, # percent which requires hospitalization
  need_uci = 5, # percent which requires uci
  beds = 4404,
  average_days_in_uci = 10 # total rough estimate
```

Here is pseudo-code of model definition:

```
model = log(daily UCI admission) ~ days, weights = 1 / (1 + today - date)
```

Code

```
## Load libraries
library(covid19)  #devtools::install_github('databrew/covid19')
library(ggplot2)
library(lubridate)
library(dplyr)
library(ggplot2)
library(sp)
library(raster)
library(viridis)
library(ggthemes)
library(sf)
library(rnaturalearth)
library(rnaturalearthdata)
```

```
# Define parameters
# Basing off: https://www.niusdiario.es/multimedia/nius-te-explica/colpaso-sistema-sanitario-espana-uci
p <- list(
  beds_public = 3508,
  beds_private = 896,
  normal_occupancy = 50,
  need_hospitalization = 15, # percent which requires hospitalization
  need_uci = 5, # percent which requires uci
  beds = 4404,
  average_days_in_uci = 10 # total rough estimate
)
options(scipen = '999')


make_prediction <- function(data,
                            n_start = 5,
                            cumulative = FALSE,
                            time_ahead = 7,
                            var = 'uci'){

  sub_data <- data
  # Get the var
  the_var <- paste0(var, ifelse(cumulative, '', '_non_cum'))
  sub_data$var <- as.numeric(unlist(sub_data[,the_var]))
  # narrow down
  sub_data <- sub_data %>%
    dplyr::select(date, var)

  pd <- sub_data %>%
        filter(!is.na(var)) %>%
      mutate(start_date = min(date[var >= n_start])) %>%
      mutate(days_since = date - start_date) %>%
      filter(days_since >= 0) %>%
      mutate(days_since = as.numeric(days_since)) %>%
      mutate(the_weight = 1/(1 + (as.numeric(max(date) - date))))
  fit <- lm(log(var) ~ days_since,
              weights = the_weight,
              data = pd)
```

```
    # Predict days ahead
    day0 <- pd$date[pd$days_since == 0]
    fake <- tibble(days_since = seq(0, max(pd$days_since) + time_ahead, by = 1))
    fake <- fake %>%mutate(date = seq(day0, day0+max(fake$days_since), by = 1))
    fake <- left_join(fake, pd %>% dplyr::select(days_since, var, date))
    fake$predicted <- exp(predict(fit, newdata = fake))
    # fake$predictedlo <- predict(fitlo, newdata = fake)
    ci <- exp(predict(fit, newdata = fake, interval = 'prediction'))
    # cilo <- predict(fitlo, newdata = fake, interval = 'prediction')

    fake$lwr <- ci[,'lwr']
    fake$upr <- ci[,'upr']
    # fake$lwrlo <- ci[,'lwr']
    # fake$uprlo <- ci[,'upr']
    # Doubling time
    dt <- log(2)/fit$coef[2]
    fake %>% mutate(doubling_time = dt)
}

plot_prediction <- function(data, ylog = F,
                            ci = FALSE){
  long <- data %>%
    tidyr::gather(key, value, var:predicted) %>%
    mutate(key = ifelse(key == 'var', 'Observed', key)) %>%
    mutate(key = Hmisc::capitalize(key))
  g <- ggplot()
  if(ci){
    g <- g +
      geom_ribbon(data = data %>% filter(date > max(long$date[!is.na(long$value) & long$key == 'Observe
                aes(x = date,
                    ymax = upr,
                    ymin = lwr),
                alpha =0.6,
                fill = 'darkorange')
  }
  g <- g +
    geom_line(data = long,
              aes(x = date,
                  y = value,
                  group = key,
                  lty = key)) +
    geom_bar(data = long %>% filter(key == 'Observed'),
             stat = 'identity',
             alpha = 0.6,
               aes(x = date,
                   y = value)) +
    theme_simple() +
    theme(legend.position = 'right',
          legend.title = element_blank())
  if(ylog){
    g <- g + scale_y_log10()
  }
```

```
  return(g)
}


spain_data <-
  esp_df %>% group_by(date) %>%
                                summarise_at(.vars = vars(uci, deaths, confirmed_cases,
                                                          uci_non_cum,
                                                          deaths_non_cum,
                                                          confirmed_cases_non_cum),
                                                          .fun = function(x){sum(x, na.rm = TRUE)})

pd <- make_prediction(data = spain_data,
                      n_start = 20,
                      cumulative = FALSE,
                      time_ahead = 10,
                      var = 'uci')
# Get the number of daily admissions to "spillover" for the number of days they need to be in UCI
# this function not estimating confidence bounds at this point
spill_over <- function(data,
                       days = p$average_days_in_uci){
  out <- data
  out$predicted_spilled_over <- NA
  out_list <- list()
  for(i in 1:nrow(out)){
    # Get the sub data for up to days days before
    sub_data <- out %>%
      filter(date >= out$date[i] -( days-1),
             date <= out$date[i])
    # Get the sum of ingresado people during that window
    sum_ingresado <- sum(sub_data$var, na.rm = T)
    # Get the predicted sum too
    sum_predicted <- mean(sub_data$predicted, na.rm = TRUE) * nrow(sub_data)
    # Manually replace with observed
    out_predicted <- ifelse(!is.na(sub_data$var[i]),
                            sum_ingresado,
                            sum_predicted)
    message(i, ": ", round(out_predicted))
    # pop back into dataframe
    # out_list[[i]] <- out_predicted
    out$predicted_spilled_over[i] <- out_predicted
  }
  # out <- unlist(out_list)
  return(out)
}
# preds <- spill_over(pd)
pd <- pd %>% spill_over(days = p$average_days_in_uci)
```

Results

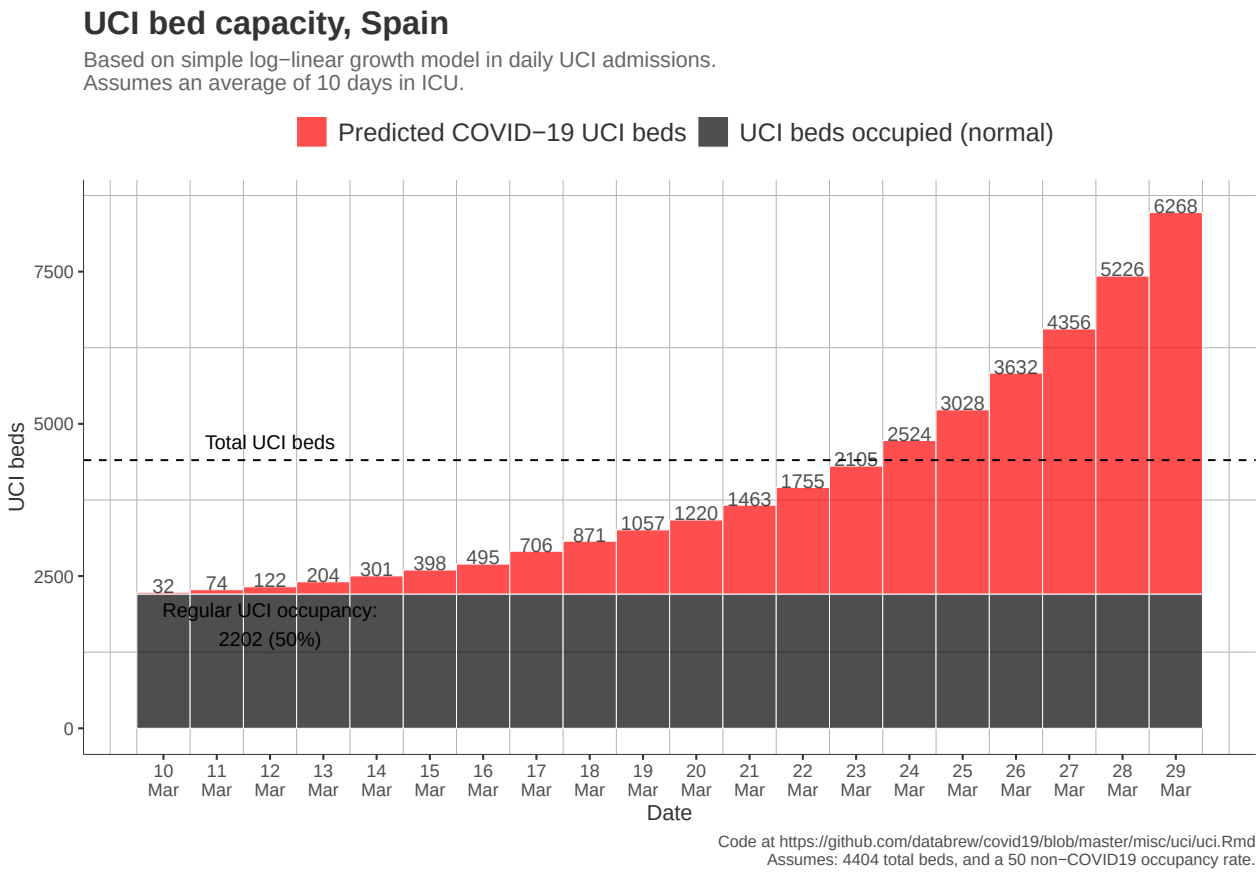
```r
already_occupied <- (p$normal_occupancy/100) * p$beds
total_beds <- p$beds

# Shape data for plotting
plot_data <- pd %>%
  dplyr::select(date, var, predicted, predicted_spilled_over) %>%
  mutate(already_occupied = already_occupied) %>%
  tidyr::gather(key, value, var:already_occupied) %>%
  mutate(key = ifelse(key == 'already_occupied',
                      'UCI beds occupied (normal)',
                      ifelse(key == 'predicted_spilled_over',
                             'Predicted COVID-19 UCI beds',
                             key)))

basic_data <- plot_data %>%
              filter(key %in% c('UCI beds occupied (normal)',
                                'Predicted COVID-19 UCI beds'))
# basic_data$key <- factor(basic_data$key,
#                          levels = rev(c('UCI beds occupied (normal)',
#                                'Predicted COVID-19 UCI beds')))

ggplot() +
  geom_bar(data = basic_data,
           aes(x = date,
               y = value,
               fill = key),
           stat = 'identity',
           position = position_stack(),
           alpha = 0.7, width = 1,
           color = 'white',
           lwd = 0.1) +
  geom_text(data = basic_data %>% filter(key == 'Predicted COVID-19 UCI beds'),
           aes(x = date,
               y = value + already_occupied * 1.05,
               label = round(value, digits = 0)),
           # stat = 'identity',
           position = position_stack(),
           alpha = 0.7) +
  # scale_y_log10() +
  geom_hline(yintercept = total_beds,
             lty = 2) +
  geom_text(data = tibble(x = min(plot_data$date) +2,
                          y = total_beds + 300,
                          label = 'Total UCI beds'),
            aes(x = x,
                y = y,
                label = label)) +
  geom_text(data = tibble(x = min(plot_data$date) +2,
                          y = already_occupied - 500,
                          label = paste0('Regular UCI occupancy:\n', round(already_occupied), ' (', p$n
            aes(x = x,
                y = y,
                label = label)) +
```

```
  theme_simple() +
  theme(legend.position = 'top') +
  scale_fill_manual(name = '',
                    values = c('red', 'black')) +
  labs(x = 'Date',
       y = 'UCI beds',
       title = 'UCI bed capacity, Spain',
       subtitle = paste0('Based on simple log-linear growth model in daily UCI admissions.\nAssumes an a
       caption = paste0('Code at https://github.com/databrew/covid19/blob/master/misc/uci/uci.Rmd\nAssu
  scale_x_date(breaks = sort(unique(plot_data$date)),
               labels = format(sort(unique(plot_data$date)), '%d\n%b'))
```




```
ggsave('~/Desktop/uci.png',
       height = 7, width = 10)
```

Table of results:

Note: `Predicted including stay-over time` means that amount of people in the UCI taking into account the fact that they stay for `p$average_days_in_uci` days.

```
pd %>% dplyr::select(date, `Observed daily admissions` = var,
    `Predicted daily admissions based on model` = predicted,
    `Predicted including stay-over time` = predicted_spilled_over) %>%
    kable
```

date	Observed daily admissions	Predicted daily admissions based on model	Predicted including stay-over time
2020-03-10	32	39.26825	32.000

date	Observed daily admissions	Predicted daily admissions based on model	Predicted including stay-over time
2020-03-11	42	47.10136	74.000
2020-03-12	48	56.49699	122.000
2020-03-13	82	67.76682	204.000
2020-03-14	97	81.28473	301.000
2020-03-15	97	97.49915	398.000
2020-03-16	97	116.94796	495.000
2020-03-17	211	140.27636	706.000
2020-03-18	165	168.25824	871.000
2020-03-19	186	201.82185	1057.000
2020-03-20	NA	242.08062	1219.534
2020-03-21	NA	290.37008	1462.803
2020-03-22	NA	348.29217	1754.598
2020-03-23	NA	417.76836	2104.600
2020-03-24	NA	501.10345	2524.418
2020-03-25	NA	601.06194	3027.981
2020-03-26	NA	720.95984	3631.993
2020-03-27	NA	864.77458	4356.491
2020-03-28	NA	1037.27702	5225.510
2020-03-29	NA	1244.18969	6267.878