Coronavirus Second Waves

Tyler DeGroff

Import New York Times Coronavirus Data

```r
setwd("/Users/tylerdegroff/Documents/Github/NYTimes\ COVID-19\ Data")
nytimes <- read.csv("us-counties.csv")
setwd("/Users/tylerdegroff/Documents/Github/Coronavirus")
```

Source: New York Times, The, Smith, M., Yourish, K., Almukhtar, S., Collins, K., Ivory, D., & Harmon, A. (2020, January–July). *Coronavirus (Covid-19) Data in the United States* [Cumulative counts of coronavirus cases in the United States, at the county level, over time (daily frequency).]. Github. https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties.csv

```r
data <- nytimes %>%
  filter(county != "Unknown") %>%
  mutate(date = as.Date(date)) %>%
  within(., fips[county == "New York City"] <- 36999) # treats NYC as a county
```

Import Census Bureau FIPS Code Metadata

```r
fips.state <- read_excel("fips.state.xlsx", skip = 4)
```

Source: United States Census Bureau. (2020, May 20). *2019 Census Bureau Region and Division Codes and State FIPS Codes* [Reference file for vintage 2019 Census Bureau state-level FIPS codes.]. United States Department of Commerce. https://www2.census.gov/programs-surveys/popest/geographies/2019/state-geocodes-v2019.xlsx

```r
fips.state <- fips.state %>%
  rename(fips.state = "State (FIPS)", state = "Name") %>%
  select(fips.state, state) %>%
  filter(fips.state != "00") %>%
  arrange(fips.state)
```

```r
fips.granular <- read_excel("fips.granular.xlsx", skip = 4)
```

Source United States Census Bureau. (2020, May 20). *2019 State, County, Minor Civil Division, and Incorporated Place FIPS Codes* [Reference file for vintage 2019 Census Bureau county-level FIPS codes.]. United States Department of Commerce. https://www2.census.gov/programs-surveys/popest/geographies/2019/all-geocodes-v2019.xlsx

```r
fips.county <- fips.granular %>%
  rename(
    fips.state = "State Code (FIPS)",
    fips.county = "County Code (FIPS)",
    county = "Area Name (including legal/statistical area description)"
  ) %>%
  mutate(fips = as.numeric(paste0(
    as.character(fips.state),
    as.character(fips.county)
  ))) %>%
  filter(fips.county != "000") %>%
  select(fips, fips.state, fips.county, county)
```

```r
fips <- merge(x = fips.county, y = fips.state, by = "fips.state", all.x = TRUE)
fips <- fips[complete.cases(fips[, "state"]), ]
```

Import Census Bureau Population Estimates

```r
pop <- read_excel("pop.xlsx", skip = 3)
```

```
## New names:
## * `` -> ...1
```

Source: United States Census Bureau. (2010–2019, April 1–July 1). *County Population Totals: 2010-2019* [Annual estimates of the county-level resident population, over time (annual frequency).]. United States Department of Commerce. https://www2.census.gov/programs-surveys/popest/tables/2010-2019/counties/totals/co-est2019-annres.xlsx

```r
pop <- pop %>%
  rename(countyState = "...1", population = "2019") %>%
  filter(countyState != "United States") %>%
  select(countyState, population)
```

```r
fips <- fips %>% mutate(countyState = paste0(county, ", ", state))
pop <- merge(x = pop, y = fips, by = "countyState", all.x = TRUE)
```

Aggregate and Treat NYC as Its Own County

```r
pop.nyc <- pop %>%
  filter(
    fips == 36005 | # Bronx County (Bronx)
    fips == 36047 | # Kings County (Brooklyn)
    fips == 36061 | # New York County (Manhattan)
    fips == 36081 | # Queens County (Queens)
    fips == 36085   # Richmond County (Staten Island)
  )

pop <- rbind(
  pop,
  data.frame(
    fips.state = "36",    # actual New York State state-level FIPS code
    fips.county = "999", # synthetic county-level FIPS code
    fips = "36999",       # synthetic FIPS code
    county = "New York City",
    state = "New York",
    countyState = "New York, New York",
    population = sum(pop.nyc$population)
  )
)
```

```r
data <- merge(
  x = data,
  y = pop %>% select(fips, population),
  by = "fips",
  all.x = TRUE
)

data <- data %>%
  mutate(
    cases.percap = cases / population,
    deaths.percap = deaths / population
  )

data <- data[order(data$date, data$state, data$county), ]

data <- data %>%

  # mutate across dates by unique county/state combinations

  mutate(countyState = paste0(county, ", ", state)) %>%
  group_by(countyState) %>%

  mutate(

    cases.new  = c(cases[1], diff(cases)),
    deaths.new = c(deaths[1], diff(deaths)),

    cases.new.7dsma  = rollmean(cases.new, k = 7, fill = NA, align = "right"),
    deaths.new.7dsma = rollmean(deaths.new, k = 7, fill = NA, align = "right")

  )
```

```r
data.state <- data %>%

  # aggregate across counties by unique date/state combination

  mutate(dateState = paste0(date, ", ", state)) %>%
  group_by(dateState) %>%
  summarize(
    date = date[1],
    state = state[1],
    cases = sum(cases),
    deaths = sum(deaths),
    pop = sum(population)
  ) %>%
  mutate(
    cases.perCap = cases / pop,
    deaths.perCap = deaths / pop
  ) %>%

  # mutate across individual states, exclusively
  group_by(state) %>%

  mutate(
    cases.new = c(cases[1], diff(cases)),
    deaths.new = c(deaths[1], diff(deaths)),

    cases.new.perCap = c(cases.perCap[1], diff(cases.perCap)),
    deaths.new.perCap = c(deaths.perCap[1], diff(deaths.perCap)),

    cases.new.perM = cases.new.perCap * 1000000,
    deaths.new.perM = deaths.new.perCap * 1000000
  ) %>%

  mutate(
    cases.new.7dsma = rollmean(cases.new, k = 7, fill = 0, align = "right"),
    deaths.new.7dsma = rollmean(deaths.new, k = 7, fill = 0, align = "right"),

    cases.new.perCap.7dsma =
      rollmean(cases.new.perCap, k = 7, fill = 0, align = "right"),
    deaths.new.perCap.7dsma =
      rollmean(deaths.new.perCap, k = 7, fill = 0, align = "right"),

    cases.new.perM.7dsma =
      rollmean(cases.new.perM, k = 7, fill = 0, align = "right"),
    deaths.new.perM.7dsma =
      rollmean(deaths.new.perM, k = 7, fill = 0, align = "right"),
  ) %>%

  mutate(
    cases.active = rollsum(cases.new, k = 9, fill = 0, align = "right"),
    cases.active.perCap = rollsum(cases.new.perCap, k = 9, fill = 0, align = "right"),
    cases.active.perM = rollsum(cases.new.perM, k = 9, fill = 0, align = "right")
  )
```

```r
write_csv(data, "data.csv")
write_csv(data.state, "data.state.csv")
```

Analysis

```r
guests <- data.frame(
  state = c("Connecticut", "Wisconsin", "South Dakota", "Indiana", "Arizona", "Nebraska"),
  guests = c(1, 4, 8, 3, 4, 200 - 8 - 3 - (4 * 2) - 1)
)

wedding <- merge(
  x = guests,
  y = data.state %>%
    filter(
      date == max(date),
      state %in% unique(guests$state)
    ) %>%
  select(cases.active.perM),
  by = "state"
) %>% arrange(desc(cases.active.perM))

kable(
  wedding,
  col.names = c("State", "Guests", "Active Cases per Million")
)
```

State	Guests	Active Cases per Million
South Dakota	8	6661.324
Wisconsin	4	5061.801
Nebraska	180	3649.178
Indiana	3	2336.377
Connecticut	1	1233.842
Arizona	4	1011.717

```r
summary <- data.frame(
  state = "Summary",
  guests = sum(wedding$guests),
  cases.active.perM = weighted.mean(
    x = wedding$cases.active.perM,
    w = wedding$guests
  )
)

kable(
  summary,
  col.names = c("", "Total Guests", "W.Avg. Active Cases/Mln.")
)
```

	Total Guests	W.Avg. Active Cases/Mln.
Summary	200	3713.399

Binomial Probability Distriubtion Cumulative Density Function

$$P(q > x|n, p) = \binom{n}{x} p^x \left(1 - p\right)^{(n-x)} \tag{1}$$

$$1 - P(x > 0|n = 200, p = \frac{3810}{10^7}) = 1 - \frac{200!}{0!(200-0)!}\frac{3810}{10^7}^0 \left(1 - \frac{3810}{10^7}\right)^{200-0} \tag{2}$$

```r
pbinom(
  q = 0,
  size = sum(wedding$guests),
  prob = weighted.mean(x = wedding$cases.active.perM, w = wedding$guests) / 1000000,
  lower.tail = FALSE
)
```

```
## [1] 0.5248202
```