Neil M Ferguson, Daniel Laydon, Gemma Nedjati-Gilani et al. Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand. Imperial College London (16-03-2020), doi: https://doi.org/10.25561/77482.

The article cited above models the transmission of COVID-19 and the progression of the epidemic across the United States and Great Britain with a variety of mitigation policies in place. This article, published in 2020, was one of the earliest to model the pandemic and predict the efficacy of and influenced policy makers in both studied countries in the early weeks of the pandemic (Mathematical Models for COVID-19 Pandemic: A Comparative Analysis | SpringerLink). The method of creating the transmission model in this article started with adapting a previously published individual-based simulation model (Strategies for mitigating an influenza pandemic - PMC (nih.gov). High-resolution population density data was used to define the areas in which individuals reside. Additional data was used to model with whom individuals come into contact.

Data wrangling occurs in this research when researchers use census data of the two countries to define the age and household distribution size of individuals. Additionally, researchers pull information on schools and workplaces, including average class sizes, staff-and-student ratios, population densities, distributions of workplace size, and commuting distance data. This data was used to create a synthetic population of schools and workplaces and geographically visualize the data. After collecting this data and preparing for use in the modelling, the researchers assigned individuals to locations created from the school and workplace data. The data wrangling steps of the research may take a considerable amount of time because of the variety of data needed from large data sets; however, it may not take as long as the visualization and exploration steps. Visualization and exploration may take several iterations to ensure the data is accurately represented and organized before being passed through the model. Additional data wrangling may be required before this step is complete.

To create the model, transmission events had to be appropriately defined. This required additional data from social mixing surveys to appropriately estimate contact patterns. Additionally, the model was created using assumptions coronavirus research in China and incubation period, infectiousness of symptomatic and asymptomatic persons, and basic reproduction number. The model is used to determine the number of hospitalizations and fatalities over time as the virus spreads, and it is repeatedly modified adding in different mitigation policies, such as, social distancing, home isolation, and school closures. The different results from each iteration of the model are presented as graphs and tables. For this research, the modelling step likely does not take a lot of time because the researchers are repurposing a previously used model; however, they may have add several iterations to evaluate the model with new parameters and assumptions.

R will be sufficient at all stages of the analysis. R has a variety of data wrangling packages, including readr, dplyr, and tidyr. Data visualization can be done in R with ggplot2 and ggplot. R packages include a large collection of statistical models and methods. Finally, RMarkdown and knitr allow researchers create reproducible reports for publishing.

Arithmetic ——————————————————

```
2+2
3 + 3
10 + 17 + 5
5 - 3
8 - 1
5 * 5
2 * 4
1 / 2
2 / 7
2^8
10^3
2**8
10**3
2+3-6 / 2
10 * 2 - 3 / 5^2
2 / (3 * 2)
(2 / 3) * 2
-2 * 3
10 ^ -3
```

Variables and assignments ——————————

```
x <- (12/3.5)^2 + (1/2.5)^3 + (1+2+3)^0.33
x # returns 13.6254
y <- x^2
y # returns 185.6516
y <- x * 3.6
y # returns 49.05145
```

Vectors ——————————————————-

```
primes <- c(2, 3, 5, 7, 11, 13, 17, 19, 23)
primes + 1 # performs operation on every value in vector
primes / 2
primes ^ 2

primes[1] # returns the first value in vector
primes[5]

primes[c(7,5,3)] # returns the 7th, 5th, and 3rd elements in primes
primes[2:5] # returns 2nd, 3rd, 4th, and 5th elements of primes vector
primes[-1] # returns all but the first element of primes
primes[-2]
primes[-c(7,5,3)] # returns all but the 7th, 5th, and 3rd elements of primes

x[1] # single values are considered vectors with one element

class(primes) # returns "numeric"

nation <- c('ireland', 'england', 'scotland', 'wales', 'france', 'italy')
```

```
class(nation) # returns "character"

nation[1] # returns 'ireland'
nation[-2] # returns all values except england
nation[4:6] #returns 4th, 5th, and 6th elements of nation

nation + 2 # returns an error
nation * 2 # returns an error

is_male <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
class(is_male) # returns logical

is_male[2] # returns FALSE
is_male[c(1,2)] # returns TRUE FALSE

result <- is_male * 2
class(result) # returns "numeric"

TRUE & FALSE # returns FALSE
TRUE & TRUE # returns TRUE
c(T, F, T) & c(T, T, F) # returns TRUE FALSE FALSE
TRUE | FALSE # return TRUE
TRUE | TRUE # returns TRUE
!TRUE #returns FALSE
!FALSE #return TRUE
(TRUE | !TRUE) & FALSE # returns FALSE
(TRUE | !TRUE) & !FALSE # returns TRUE

primes == 7 # returns logical value for each element in primes against the
condition
12 == (6*2)

12 != (6*2)
primes != 3 # returns logical value for each element checking if it is not
equal to 3

primes < 5
primes > 8
primes <= 7
primes >= 3

nation == 'england'
nation != 'england'

nation >= 'italy'

coerced_num <- c(TRUE, FALSE, 3, 2, -1) #logical values will convert to
numeric
coerced_str <- c(2.75, 11.3, TRUE, FALSE, 'dog', 'cat') #all values will
become character strings
```

```
c(primes, primes^2, primes^3)

ages <- c(bob = 27, bill = 19, charles = 24)
ages
ages['bob'] # returns 27
ages['bill'] # returns 19

ages_again <- c(27, 19, 24)
names(ages_again) <- c('bob', 'bill', 'charles')
ages_again # equivalent to ages

w <- c(1, 2, NA, 4, 5)
w
z <- c('be', NA, 'afeard')
z
c(1, 2, NA, 4, 5) # numeric vector
c(1, 2, 'NA', 4, 5) # character vector
```

Data Frames ——————————————————

```
data_df <- data.frame(name = c('billy', 'joe', 'bob'), age = c(21, 29, 23))
data_df[3,2] # returns value at row 3 column 2
data_df[c(1,3), 2] # returns values of 1st and 3rd row in 2nd column
data_df[3,] # returns all elements in third row
data_df[,2] # returns all values of the 2nd column
data_df$age
data_df[['age']]
data_df['age'] #returns subset of data_df that is itself a data frame
```

#Other data structures ——————————————–

```
example_list <- list(A = TRUE, B = c(1, 2, 3), C = c('cat', 'dog'))
example_list
example_list[['B']]
example_list$B
example_list[[1]]

matrix(c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29), nrow=2, ncol=5)
matrix(c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29), byrow = T, nrow=2, ncol=5)

rbind(c(1, 2), c(3, 4)) # binds by row
cbind(c(1, 2), c(3, 4)) # binds by column

primes_m <- matrix(c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29), byrow = T, nrow=2,
ncol=5)
primes_m
primes_m[1, 3] # row 1, column 3
primes_m[2, ] # row 2, all cols
primes_m[, 3] # column 3, all rows

example_array <- array(c(1, 2, 3, 4, 5, 6, 7, 8), dim=c(2,2,2))
example_array[1,2,]
```

```
example_array[,1,]
example_array
```

Functions ————————————————-

```
length(primes)
sum(primes)
mean(primes)
median(primes)
sd(primes)
var(primes)

round(sqrt(mean(primes)))

mean(primes, trim = 0.1) # trims the 10% highest and lowest values

my_mean <- function(x){ sum(x)/length(x) } # custom function
my_mean(primes)
```

Read File ————————————————

```
data <- read_csv("C:\\Users\\janna\\OneDrive\\Documents\\DSE5001\\Week
1\\weight.csv")
```