CPSC 375 Project 1

Kenn Son, Hamid Suha, Vivian Truong

4/7/2022

```
setwd("~/Documents/CPSC-375")

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(class)
library(ggplot2)
```

1) Data preparation/wrangling to get all the data into one table that can be used for linear modeling

   a)reading the data files using read_csv()

```
covid <- read_csv("https://raw.githubusercontent.com/govex/COVID-19/master/data_tables/vaccine_data/glo
```

```
## Rows: 831 Columns: 512
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (5): iso2, iso3, Province_State, Country_Region, Combined_Key
## dbl (505): UID, code3, Lat, Long_, Population, 2020-12-12, 2020-12-13, 2020-...
## lgl   (2): FIPS, Admin2
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bed <- read_csv("hospitalbed.csv")
```

```
## Rows: 1770 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Country
## dbl (2): Year, Hospital beds (per 10 000 population)
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
demo <- read_csv("demographics.csv")
```

```
## Rows: 3885 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Country Name, Country Code, Series Name, Series Code
## dbl (1): YR2015
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

b) Removing unneeded rows (e.g., countries like Brazil and India report Province_State-level data that is not needed as we are studying only country-level rates) and columns.

```
covid <- covid %>% filter(Population >= 0, is.na(Province_State))
covid <- covid %>% select(-Admin2, -FIPS, -Province_State, -UID, -iso2, -iso3, -code3, -Lat, -Long_, -Co
covid
```

```
## # A tibble: 134 x 502
##    Country_Region Population `2020-12-12` `2020-12-13` `2020-12-14` `2020-12-15`
##    <chr>               <dbl>        <dbl>        <dbl>        <dbl>        <dbl>
##  1 Afghanistan      38928341           NA           NA           NA           NA
##  2 Albania           2877800           NA           NA           NA           NA
##  3 Algeria          43851043            0            0            0            0
##  4 Andorra             77265            0            0            0            0
##  5 Angola           32866268           NA           NA           NA           NA
##  6 Antigua and B~      97928           NA           NA           NA           NA
##  7 Argentina        45195777            0            0            0            0
##  8 Australia        25459700           NA           NA           NA           NA
##  9 Austria           9006400            0            0            0            0
## 10 Azerbaijan       10139175           NA           NA           NA           NA
## # ... with 124 more rows, and 496 more variables: `2020-12-16` <dbl>,
## #   `2020-12-17` <dbl>, `2020-12-18` <dbl>, `2020-12-19` <dbl>,
## #   `2020-12-20` <dbl>, `2020-12-21` <dbl>, `2020-12-22` <dbl>,
## #   `2020-12-23` <dbl>, `2020-12-24` <dbl>, `2020-12-25` <dbl>,
## #   `2020-12-26` <dbl>, `2020-12-27` <dbl>, `2020-12-28` <dbl>,
## #   `2020-12-29` <dbl>, `2020-12-30` <dbl>, `2020-12-31` <dbl>,
## #   `2021-01-01` <dbl>, `2021-01-02` <dbl>, `2021-01-03` <dbl>, ...
```

```
demo <- demo %>% select(-`Series Name`, -`Country Code`)
```

c) tidying tables, as needed. For example, the vaccinations data is not tidy.

```r
covid <- covid %>% pivot_longer(-c(Country_Region, Population), names_to = "Date", values_to = "shots")
covid <- covid %>% filter(shots > 0) %>% view()

#bed <- bed %>% arrange(Year) %>% pivot_wider( names_from = "Year",
#              values_from = "Hospital beds (per 10 000 population)")

demo <- demo %>% pivot_wider(names_from = "Series Code", values_from = YR2015)
```

d) Calculate the vaccination rate: vaccinations/population

```r
covid <- covid %>% mutate(vacRate = shots/Population) %>% view()
covid
```

```
## # A tibble: 59,998 x 5
##    Country_Region Population Date       shots  vacRate
##    <chr>               <dbl> <chr>      <dbl>    <dbl>
##  1 Afghanistan      38928341 2021-02-28  8200 0.000211
##  2 Afghanistan      38928341 2021-03-01  8200 0.000211
##  3 Afghanistan      38928341 2021-03-02  8200 0.000211
##  4 Afghanistan      38928341 2021-03-03  8200 0.000211
##  5 Afghanistan      38928341 2021-03-04  8200 0.000211
##  6 Afghanistan      38928341 2021-03-05  8200 0.000211
##  7 Afghanistan      38928341 2021-03-06  8200 0.000211
##  8 Afghanistan      38928341 2021-03-07  8200 0.000211
##  9 Afghanistan      38928341 2021-03-08  8200 0.000211
## 10 Afghanistan      38928341 2021-03-09  8200 0.000211
## # ... with 59,988 more rows
```

e) Since the most important factor affecting vaccination rate is the number of days since vaccination began (vaccination rate always increases), calculate a variable that is: number of days since first non-zero vaccination number. This variable will be important for modeling.

```r
covid <- covid %>% group_by(Country_Region) %>% mutate(daysSinceStart = 1:n())
covid <- covid %>% select(-Date)
covid
```

```
## # A tibble: 59,998 x 5
## # Groups:   Country_Region [134]
##    Country_Region Population shots  vacRate daysSinceStart
##    <chr>               <dbl> <dbl>    <dbl>          <int>
##  1 Afghanistan      38928341  8200 0.000211              1
##  2 Afghanistan      38928341  8200 0.000211              2
##  3 Afghanistan      38928341  8200 0.000211              3
##  4 Afghanistan      38928341  8200 0.000211              4
##  5 Afghanistan      38928341  8200 0.000211              5
##  6 Afghanistan      38928341  8200 0.000211              6
##  7 Afghanistan      38928341  8200 0.000211              7
##  8 Afghanistan      38928341  8200 0.000211              8
##  9 Afghanistan      38928341  8200 0.000211              9
## 10 Afghanistan      38928341  8200 0.000211             10
## # ... with 59,988 more rows
```

f) Discard data that is not needed. For example, only the number of hospital beds from the most recent year is necessary.

```
bed.new <- bed %>% group_by(Country) %>% summarise(Year = max(Year))
bed <- inner_join(bed.new, bed)
```

```
## Joining, by = c("Country", "Year")
```

```
bed <- bed %>% mutate(beds = `Hospital beds (per 10 000 population)`) %>%
  select(-Year, -`Hospital beds (per 10 000 population)`)

bed
```

```
## # A tibble: 178 x 2
##    Country              beds
##    <chr>               <dbl>
##  1 Afghanistan           3.9
##  2 Albania              28.9
##  3 Algeria              19
##  4 Angola                8
##  5 Antigua and Barbuda  28.9
##  6 Argentina            49.9
##  7 Armenia              41.6
##  8 Australia            38.4
##  9 Austria              72.7
## 10 Azerbaijan           48.2
## # ... with 168 more rows
```

g) You can ignore sex-related differences in demographics in this project, so add the male/female population numbers together (already done in HW #5).

```
demo.total <- demo %>% mutate(SP.POP.80UP=SP.POP.80UP.FE+SP.POP.80UP.MA) %>%
  mutate(SP.POP.1564.IN=SP.POP.1564.MA.IN+SP.POP.1564.FE.IN) %>%
  mutate(SP.POP.0014.IN=SP.POP.0014.MA.IN+SP.POP.0014.FE.IN) %>%
  mutate(SP.DYN.AMRT=SP.DYN.AMRT.MA+SP.DYN.AMRT.FE) %>%
  mutate(SP.POP.TOTL.IN=SP.POP.TOTL.FE.IN+SP.POP.TOTL.MA.IN) %>%
  mutate(SP.POP.65UP.IN=SP.POP.65UP.FE.IN+SP.POP.65UP.MA.IN) %>%
  select(-contains(".FE")) %>% select(-contains(".MA"))

demo <- demo.total %>% group_by(`Country Name`) %>% summarise(SP.DYN.LE00.IN = sum(SP.DYN.LE00.IN, na.r

demo
```

```
## # A tibble: 259 x 10
##    `Country Name`      SP.DYN.LE00.IN SP.URB.TOTL SP.POP.TOTL SP.POP.80UP
##    <chr>                        <dbl>       <dbl>       <dbl>       <dbl>
##  1 Afghanistan                   63.4     8535606    34413603       85552
##  2 Albania                       78.0     1654503     2880703       66965
##  3 Algeria                       76.1    28146511    39728025      453741
##  4 American Samoa                 0         48689       55812           0
##  5 Andorra                        0         68919       78011           0
##  6 Angola                        59.4    17691524    27884381       69363
```

```
##  7 Antigua and Barbuda           76.5      23392      93566       1571
##  8 Arab World                    71.2  229821020  396028278    2689793
##  9 Argentina                     76.1   39467043   43131966    1095211
## 10 Armenia                       74.5    1845585    2925553      77292
## # ... with 249 more rows, and 5 more variables: SP.POP.1564.IN <dbl>,
## #   SP.POP.0014.IN <dbl>, SP.DYN.AMRT <dbl>, SP.POP.TOTL.IN <dbl>,
## #   SP.POP.65UP.IN <dbl>
```

h) Merge all tables (Hint: Join using the country name)

```r
#Unifying Country Names
demo <- demo %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Republic of Korea",
                                  "South Korea")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Egypt, Arab Rep.",
                                  "Egypt")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Gambia, The",
                                  "Gambia")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "St. Vincent and the Grenadines",
                                  "Saint Vincent and the Grenadines")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "St. Lucia",
                                  "Saint Lucia")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Lao PDR",
                                  "Laos")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Slovak Republic",
                                  "Slovakia")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Bahamas, The",
                                  "Bahamas")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Iran, Islamic Rep.",
                                  "Iran")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "Venezuela, RB",
                                  "Venezuela")) %>%
  mutate(`Country Name` = replace(`Country Name`, `Country Name` == "St. Kitts and Nevis  ",
                                  "Saint Kitts and Nevis"))

bed <- bed %>%
  mutate(Country = replace(Country, Country == "Venezuela (Bolivarian Republic of)",
                           "Venezuela")) %>%
  mutate(Country = replace(Country, Country == "Viet Nam",
                           "Vietnam")) %>%
  mutate(Country = replace(Country, Country == "United States",
                           "United States of America")) %>%
  mutate(Country = replace(Country, Country == "Iran (Islamic Republic of) ",
                           " Iran")) %>%
  mutate(Country = replace(Country, Country == "Lao People's Democratic Republic",
                           " Laos")) %>%
  mutate(Country = replace(Country, Country == "Bolivia",
                           "Bolivia (Plurinational State of)"))

covid <- covid %>% mutate(Country_Region = replace(Country_Region,
                                                   Country_Region == "US", "United States of America"))

big_data <- covid %>% inner_join(bed, by=c(Country_Region = "Country")) %>%
  inner_join(demo, by=c(Country_Region = "Country Name"))
```

```r
big_data <- big_data %>% rename(Country = Country_Region) %>%
  relocate(shots, .after = Country) %>% relocate(vacRate, .after = Country)

big_data
```

```
## # A tibble: 52,447 x 15
## # Groups:   Country [117]
##    Country      vacRate shots Population daysSinceStart  beds SP.DYN.LE00.IN
##    <chr>          <dbl> <dbl>      <dbl>          <int> <dbl>          <dbl>
##  1 Afghanistan 0.000211  8200   38928341              1   3.9           63.4
##  2 Afghanistan 0.000211  8200   38928341              2   3.9           63.4
##  3 Afghanistan 0.000211  8200   38928341              3   3.9           63.4
##  4 Afghanistan 0.000211  8200   38928341              4   3.9           63.4
##  5 Afghanistan 0.000211  8200   38928341              5   3.9           63.4
##  6 Afghanistan 0.000211  8200   38928341              6   3.9           63.4
##  7 Afghanistan 0.000211  8200   38928341              7   3.9           63.4
##  8 Afghanistan 0.000211  8200   38928341              8   3.9           63.4
##  9 Afghanistan 0.000211  8200   38928341              9   3.9           63.4
## 10 Afghanistan 0.000211  8200   38928341             10   3.9           63.4
## # ... with 52,437 more rows, and 8 more variables: SP.URB.TOTL <dbl>,
## #   SP.POP.TOTL <dbl>, SP.POP.80UP <dbl>, SP.POP.1564.IN <dbl>,
## #   SP.POP.0014.IN <dbl>, SP.DYN.AMRT <dbl>, SP.POP.TOTL.IN <dbl>,
## #   SP.POP.65UP.IN <dbl>
```

2) Linear modeling the Covid vaccination rate Make a list of all predictor variables that are available. The challenge is to identify which combination of these predictors will give the best predictive model. You should also try transforming some of the variables (e.g., transforming population counts to proportion of total population). Run linear regression with at least 5 different combinations of predictor variables. Note: each day becomes one data point, i.e., the vaccination rate is calculated for each day for each country. The number of vaccinations should not be used as an independent variable as this is essentially what you are predicting.

```r
combo1 <- lm(vacRate~SP.POP.0014.IN+SP.POP.1564.IN+beds, data = big_data)
combo2 <- lm(vacRate~SP.DYN.AMRT+beds, data = big_data)
combo3 <- lm(vacRate~SP.POP.65UP.IN+beds+SP.POP.80UP, data = big_data)
combo4 <- lm(vacRate~SP.POP.80UP+beds, data = big_data)
combo5 <- lm(vacRate~Population+beds+daysSinceStart, data = big_data)

summary(combo1)
```

```
##
## Call:
## lm(formula = vacRate ~ SP.POP.0014.IN + SP.POP.1564.IN + beds,
##     data = big_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4162 -0.5805 -0.1556  0.5060  2.0604
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.535e-01  5.225e-03  125.07   <2e-16 ***
```

```
## SP.POP.0014.IN -6.597e-09  2.346e-10  -28.12   <2e-16 ***
## SP.POP.1564.IN  2.248e-09  7.994e-11   28.12   <2e-16 ***
## beds            3.688e-03  1.341e-04   27.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6636 on 52443 degrees of freedom
## Multiple R-squared:  0.04034,    Adjusted R-squared:  0.04029
## F-statistic: 734.9 on 3 and 52443 DF,  p-value: < 2.2e-16
```

```
summary(combo2)
```

```
##
## Call:
## lm(formula = vacRate ~ SP.DYN.AMRT + beds, data = big_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.15269 -0.52168 -0.06899  0.45815  1.89809
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.090e+00  7.489e-03 145.476  < 2e-16 ***
## SP.DYN.AMRT -1.408e-03  1.736e-05 -81.096  < 2e-16 ***
## beds         8.468e-04  1.309e-04   6.471 9.81e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6304 on 52444 degrees of freedom
## Multiple R-squared:  0.1341, Adjusted R-squared:  0.1341
## F-statistic:  4063 on 2 and 52444 DF,  p-value: < 2.2e-16
```

```
summary(combo3)
```

```
##
## Call:
## lm(formula = vacRate ~ SP.POP.65UP.IN + beds + SP.POP.80UP, data = big_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5546 -0.5897 -0.1543  0.5101  2.0911
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.294e-01  5.112e-03  123.12   <2e-16 ***
## SP.POP.65UP.IN -1.467e-08  1.084e-09  -13.53   <2e-16 ***
## beds            3.534e-03  1.487e-04   23.76   <2e-16 ***
## SP.POP.80UP     9.868e-08  6.395e-09   15.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6667 on 52443 degrees of freedom
## Multiple R-squared:  0.03139,    Adjusted R-squared:  0.03134
## F-statistic: 566.6 on 3 and 52443 DF,  p-value: < 2.2e-16
```

```r
summary(combo4)
```

```
##
## Call:
## lm(formula = vacRate ~ SP.POP.80UP + beds, data = big_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3235 -0.5919 -0.1600  0.5082  2.0685
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.097e-01  4.909e-03  124.20   <2e-16 ***
## SP.POP.80UP 1.368e-08  1.190e-09   11.49   <2e-16 ***
## beds        4.484e-03  1.314e-04   34.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6679 on 52444 degrees of freedom
## Multiple R-squared:  0.02801,    Adjusted R-squared:  0.02798
## F-statistic: 755.7 on 2 and 52444 DF,  p-value: < 2.2e-16
```

```r
summary(combo5)
```

```
##
## Call:
## lm(formula = vacRate ~ Population + beds + daysSinceStart, data = big_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.33445 -0.22383 -0.01228  0.24680  1.27957
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -2.644e-01  4.323e-03 -61.159   <2e-16 ***
## Population     -1.372e-11  9.554e-12  -1.436    0.151
## beds            3.905e-03  7.993e-05  48.853   <2e-16 ***
## daysSinceStart  4.004e-03  1.375e-05 291.148   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4134 on 52443 degrees of freedom
## Multiple R-squared:  0.6276, Adjusted R-squared:  0.6276
## F-statistic: 2.946e+04 on 3 and 52443 DF,  p-value: < 2.2e-16
```

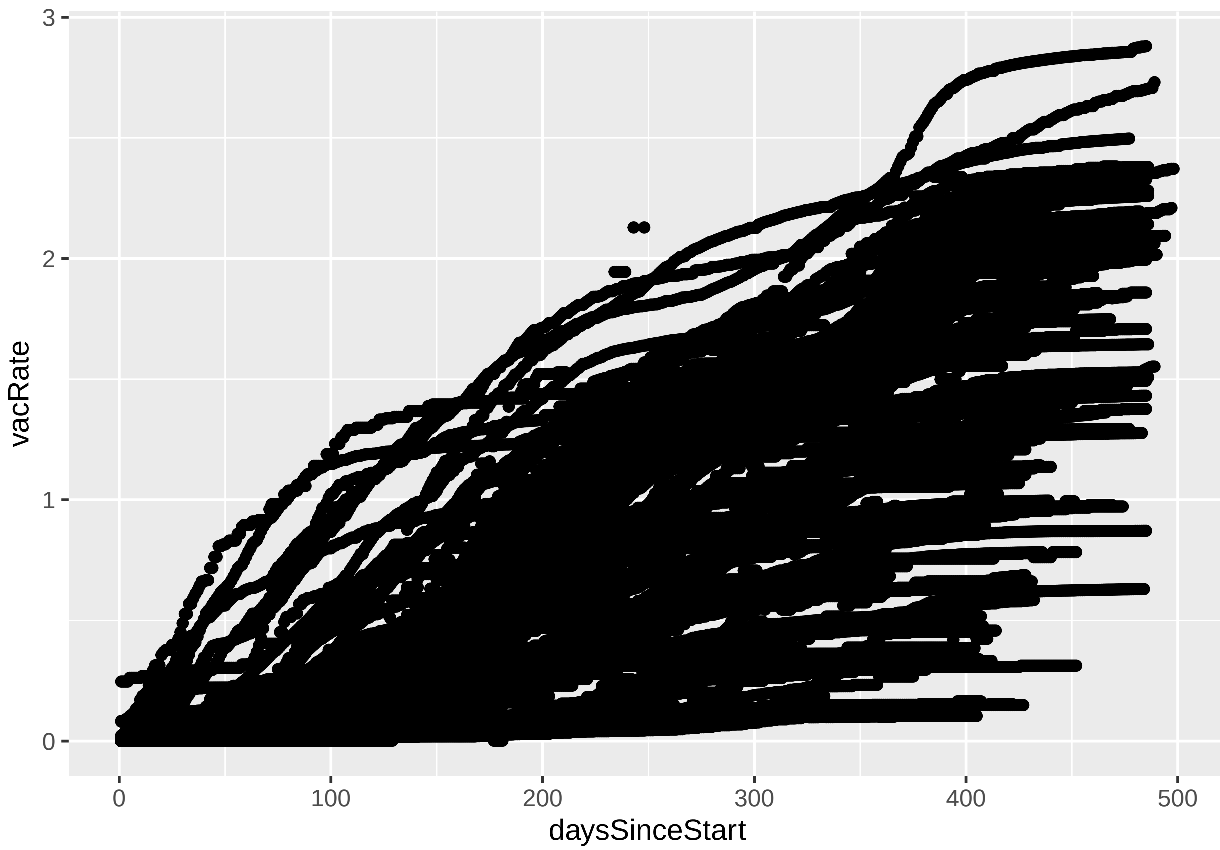
```r
cf1 <- coef(combo1)
cf2 <- coef(combo2)
cf3 <- coef(combo3)
cf4 <- coef(combo4)
cf5 <- coef(combo5)
```

Write a short report describing your data wrangling steps and the different combinations of predictor variables you tried, and any variable transforms. [A PDF file]

The report should include the following plots:

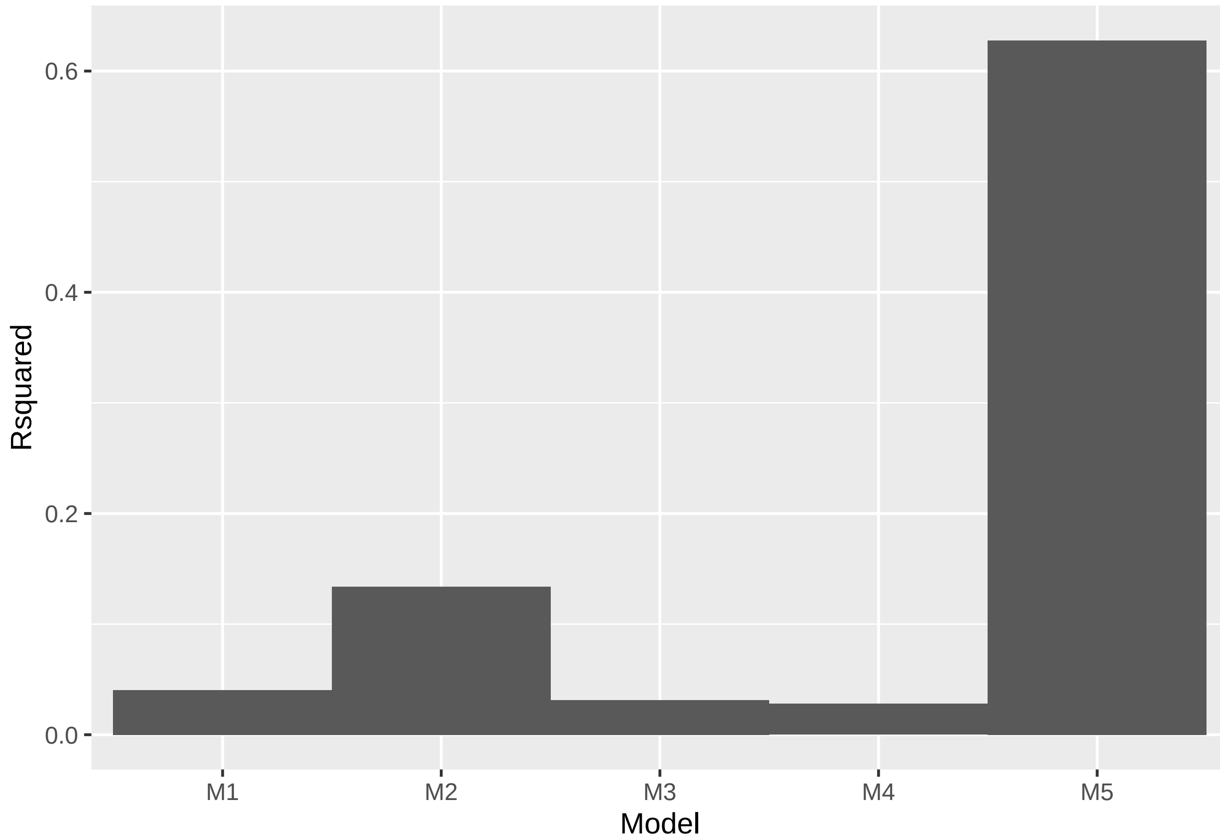
i) a scatterplot of only the most recent vaccination rate for every country and the number of days since first vaccination

```
ggplot(big_data) + geom_point(mapping = aes(x = daysSinceStart, y = vacRate))
```



ii) a summary bar graph with the R2 values on the y-axis and a corresponding model name on the x-axis (include all the different models you tried).

```
R2 = c(0.04034,0.134,0.03136,0.028,0.6278)
big_data.R2 <- data.frame (Model  = c("M1","M2","M3","M4","M5"),
                  Rsquared = R2)
ggplot(big_data.R2, aes(x = Model, y=Rsquared)) + geom_bar(stat = "identity", width=1)
```


Rsquared
0.6
0.4
0.2
0.0
M1
M2
M3
M4
M5
Model