Covid Case Modeling

Project Description

The intent of this project is to predict next day new covid-19 cases based on the previous week of cases, the population, and the population that can be infected. All data is taken as a csv file download from the website ***Our World in Data*** (OWID) on April 17 2020. A secondary purpose of this project is to show my thought process, so rather than give what I believe to be the best model I can make, I will be going through what I've tried and why I made the decisions I did.

Part 1 Setting Up the Data

The OWID data set has a ton of data most of which will not be used in this project. The first step is to filter by the countries ISO code in this document we are looking at USA.

```
covid_data = read.csv("owid-covid-data.csv")
covid_data = covid_data[covid_data$iso_code == params$iso_code,]
```

It is assumed for simplicity that any time there is a missing entry it is 0.

```
covid_data[is.na(covid_data)] = 0
```

Next a data column is created which keeps track of the infectable population on any day, I will assume that someone that has covid can't contract the disease again, and that once someone is fully vaccinated will have full immunity to the virus. The deaths will be ignored for this analysis because that's a bit depressing.

$$infectable_{day} = population - total.cases_{day} - people.fully.vacinated_{day}$$

```
covid_data$infectable_pop = covid_data$population-
  covid_data$total_cases-covid_data$people_fully_vaccinated
```

The value that will be predicted is similar to the logit ($ln(\frac{p}{1-p})$) of new cases but in order to be more accurate to real life is a little bit more complicated. Because the upper bound of people that can be new infections on any given day is equal to the number of people that are infectable the predicated value will be:

$$ln(\frac{new.cases_{day}}{infectable_{day-1} - new.cases_{day}})$$

This is essentially the logit of the probability an uninfected person will become infected. To better keep track of things I will make two functions to convert between the new cases and the modified logit.

```
to.logit.modified <- function(new.cases,infectable){
  log(new.cases/(infectable-new.cases))
}

from.logit <- function(infectable,logit){
  el = exp(logit)
  infectable*el/(1+el)
}
```

Now a new data frame will be made to house the expected values and the possible predictors for each day.

```
length = nrow(covid_data)
prediction.data = data.frame(
  result = to.logit.modified(covid_data$new_cases[8:length],
                             covid_data$infectable_pop[7:(length-1)]),
  cases_back_7 = covid_data$new_cases[1:(length-7)],
  cases_back_6 = covid_data$new_cases[2:(length-6)],
  cases_back_5 = covid_data$new_cases[3:(length-5)],
  cases_back_4 = covid_data$new_cases[4:(length-4)],
  cases_back_3 = covid_data$new_cases[5:(length-3)],
  cases_back_2 = covid_data$new_cases[6:(length-2)],
  cases_back_1 = covid_data$new_cases[7:(length-1)],
  infectable = covid_data$infectable_pop[7:(length-1)]
)
```

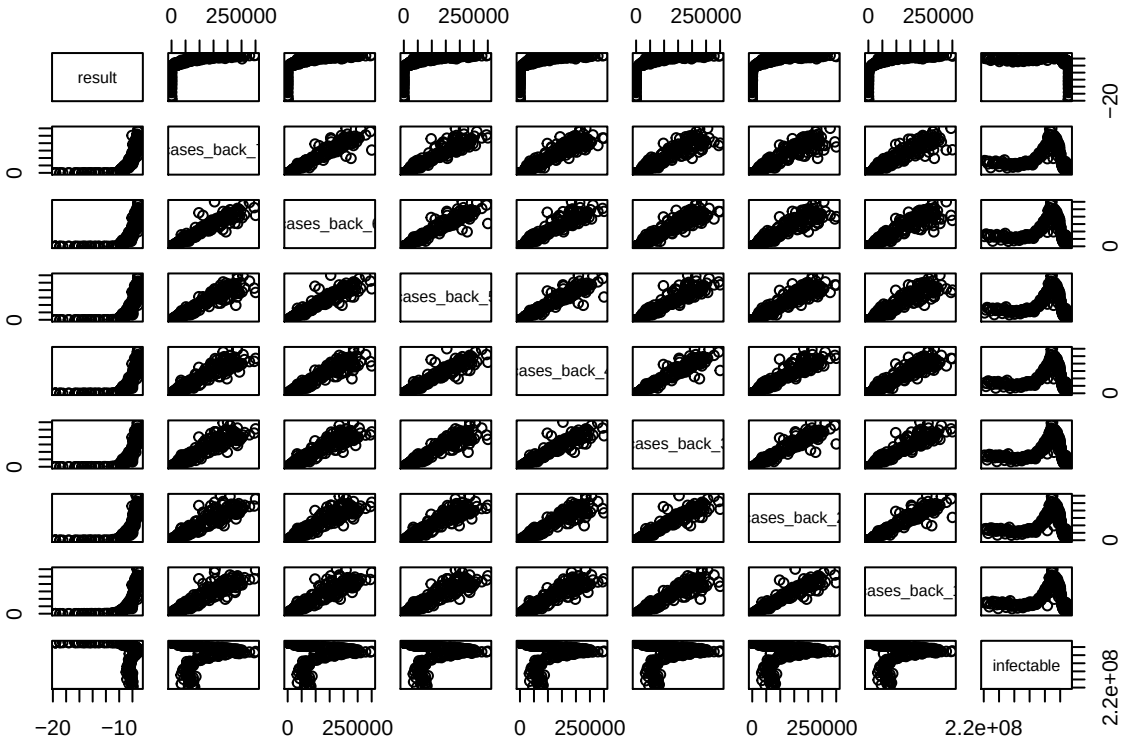
Finally a training set and test set will be made. 75% of the data will be used for training set, and 25% of the data will be used for the test set. Before the split a seed is set to 100.

```
n = nrow(prediction.data)
train.size = floor(params$trainSize * n)
test.size = n - train.size
set.seed(params$seed)

train.index = sample(1:n,size = train.size,replace=F)
train.index = sort(train.index)
test.index = setdiff(1:n,train.index)

train.data = prediction.data[train.index,]
test.data = prediction.data[test.index,]
```

Part 2 Building a Model

I think a good way to start is looking at all the predictors plotted against eachother to get a good idea if the model chosen is even a good one.

```
plot(prediction.data)
```

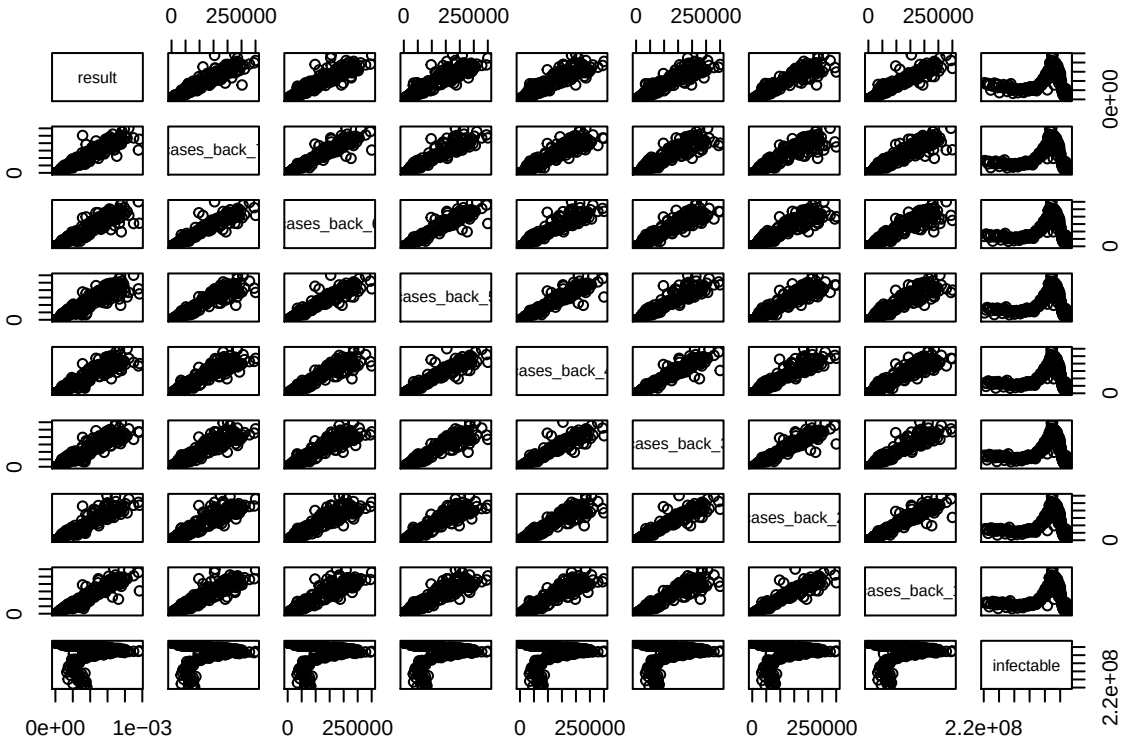


From looking at these plots it seems the new cases have an exponential relationship with the modified logit. In order to remedy this the modified logit will be changed to $\frac{new.cases_{day}}{infectable_{day-1}-new.cases_{day}}$ Also the case day ranges appear to be very related and may be redundant, this will be tested when the model is built.

```
to.odds.infected <- function(new.cases,infectable){
  new.cases/(infectable-new.cases)
}

from.odds <- function(odds,infectable){
  infectable*odds/(1+odds)
}

prediction.data$result = to.odds.infected(covid_data$new_cases[8:length],
                          covid_data$infectable_pop[7:(length-1)])
train.data = prediction.data[train.index,]
test.data = prediction.data[test.index,]
```

Lets look at the new plot of predictors.

```
plot(prediction.data)
```



There is now a clear linear relationship between the odds of being infected and the previous days cases. The range of previous days cases still seem redundant, and the infectable population still doesn't seem to have a relationship with the odds. Again this will be tested in the model.

The Models

We'll look at three models at first, compare them based on their residuals.

```
residuals <- function(model,data){
  predict(model,newdata=data)-data$result
}
RSS <- function(model,data){
  residuals = (predict(model,newdata=data)-data$result)^2
  sum(residuals)
}
```

Model 1: This model is made from the interaction of the new cases range with the infectable population.

$$odds(day) = infectable_{day-1} * \sum_{i=1}^{7} new.cases_{day-i}$$

```
model.1 = lm(result~0+cases_back_1:infectable+cases_back_2:infectable+
               cases_back_3:infectable+cases_back_4:infectable+
               cases_back_5:infectable+cases_back_6:infectable+
```

```
              cases_back_7:infectable,data = train.data)
RSS.1 = RSS(model.1,test.data)
summary(model.1)$coef
```

```
##                              Estimate   Std. Error    t value     Pr(>|t|)
## cases_back_1:infectable  3.137219e-18 6.384810e-19  4.9135669 1.418459e-06
## infectable:cases_back_2  1.851982e-18 6.711714e-19  2.7593276 6.119327e-03
## infectable:cases_back_3  9.824650e-19 6.646181e-19  1.4782399 1.403091e-01
## infectable:cases_back_4 -5.655512e-19 6.952350e-19 -0.8134676 4.165437e-01
## infectable:cases_back_5  2.023761e-19 6.796377e-19  0.2977706 7.660679e-01
## infectable:cases_back_6  8.170299e-19 7.744085e-19  1.0550373 2.921895e-01
## infectable:cases_back_7  4.025196e-18 7.519739e-19  5.3528396 1.636317e-07
```

Model 2: This one is based entirely on the range of new cases, if there is any reason to have a range of cases this model will show it.

$$odds(day) = \sum_{i=1}^{7} new.cases_{day-i}$$

```
model.2 = lm(result~0+cases_back_1+cases_back_2+cases_back_3+cases_back_4+
              cases_back_5+cases_back_6+cases_back_7,data = train.data)
RSS.2 = RSS(model.2,test.data)
summary(model.2)$coef
```

```
##                   Estimate   Std. Error    t value     Pr(>|t|)
## cases_back_1  1.074384e-09 1.778081e-10  6.0423811 4.145482e-09
## cases_back_2  6.246915e-10 1.863206e-10  3.3527771 8.941924e-04
## cases_back_3  2.975196e-10 1.846734e-10  1.6110578 1.081349e-01
## cases_back_4 -1.746069e-10 1.921999e-10 -0.9084650 3.643037e-01
## cases_back_5  5.852767e-11 1.883639e-10  0.3107159 7.562152e-01
## cases_back_6  2.633087e-10 2.143288e-10  1.2285273 2.201352e-01
## cases_back_7  1.104102e-09 2.082000e-10  5.3030846 2.105039e-07
```

Model 3: This is the simplest model designed to test the importance of the infectable population value on its own.

$$odds(day) = infectable_{day-1}$$

```
model.3 = lm(result~0+infectable,data = train.data)
RSS.3 = RSS(model.3,test.data)
```

Analysis

Model 1 RSS	Model 2 RSS	Model 3 RSS
2.8044446×10^{-7}	2.303237×10^{-7}	4.9009859×10^{-6}

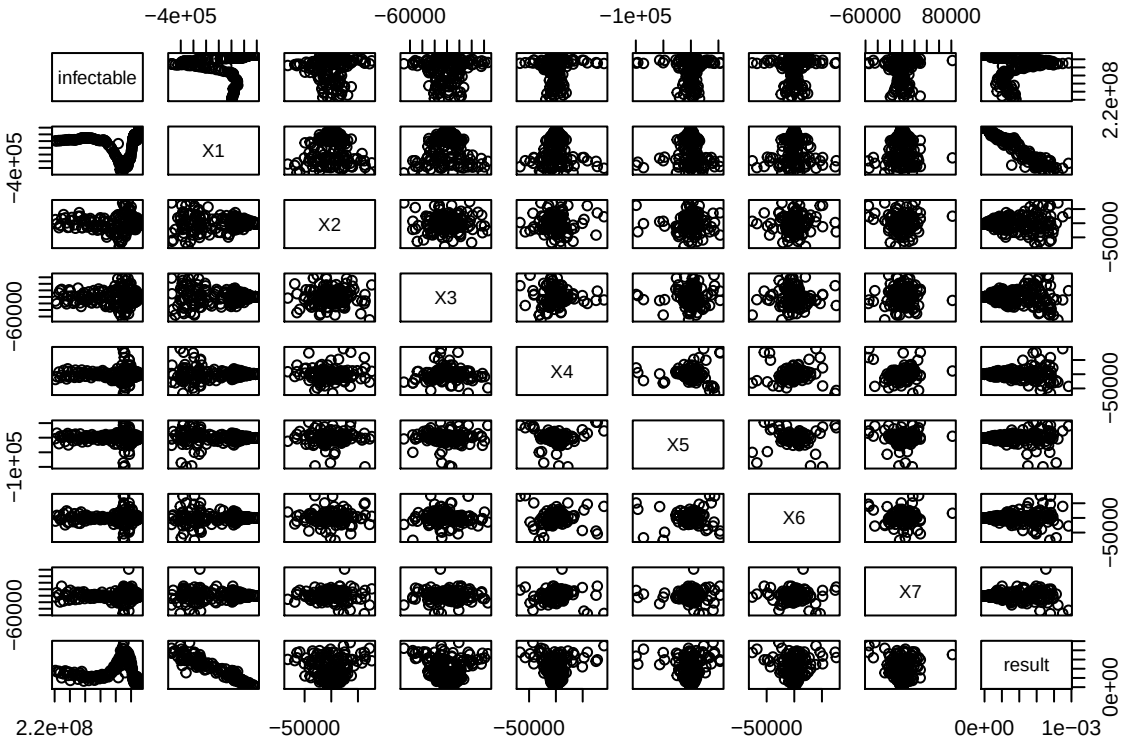
Model 3 has the largest RSS out of all 3 of the models so it does not appear to be as good of a model as 1 or 2, but it has the benefit of only using one predictor, which likely is the cause of its high RSS value. Model 1 and 2 have very similar RSS but model 2 has a slightly better RSS. I'd expect that as the infectable population shrinks in the future model 1, the infectable/new_cases interaction model would be more accurate. I will

continue with models 1 and 2 because they seem the most promising.

Building a Model with Principle Components

The Correlation between the day range of cases is an issue. In order to tackle this Principle Component Analysis will be preformed on the training data

```
means = c(mean(train.data$cases_back_1),mean(train.data$cases_back_2),mean(train.data$cases_back_3),mean

newCases = cbind((train.data$cases_back_1-means[1]),(train.data$cases_back_2-means[2]),(train.data$cases

Var = cov(newCases)
ev = eigen(Var)
eVecs = ev$vectors
eVals = ev$values

transformdata <- function(data){
  nc = cbind((data$cases_back_1-means[1]),(data$cases_back_2-means[2]),(data$cases_back_3-means[3]),(da
  nc.transformed = nc%*%eVecs
  transformed.data = data.frame(infectable = data$infectable)

  transformed.data$X1 = nc.transformed[,1]
  transformed.data$X2 = nc.transformed[,2]
  transformed.data$X3 = nc.transformed[,3]
  transformed.data$X4 = nc.transformed[,4]
  transformed.data$X5 = nc.transformed[,5]
  transformed.data$X6 = nc.transformed[,6]
  transformed.data$X7 = nc.transformed[,7]
  transformed.data
}

train.transformed.data = transformdata(train.data)
train.transformed.data$result = train.data$result
plot(train.transformed.data)
```



Now that Principle Component Analysis has been preformed on the data, and it has been transformed all the predictors are independent. This can be seen when looking at the plots of predictors against each other, for X2 - X7 there are clumps of data that center around a certain point but don't seem to have any real correlation. When comparing X1 to X2-X7 the data seems to be a funnel shape, this is probably because when we have very high case numbers there will be more variance associated with that area. X1 also seems very correlated with the predicted new cases.

Initial models Lets make some initial models based on the previous models. model 3 will not be used, because none of its data was transformed and wasn't as good as the other two when comparing RSS values.

```
model.1.transformed = lm(result~0+X1:infectable+X2:infectable+X3:infectable+X4:infectable+
              X5:infectable+X6:infectable+ X7:infectable,data = train.transformed.data)
summary(model.1.transformed)$coef
```

```
##                    Estimate   Std. Error      t value     Pr(>|t|)
## X1:infectable -3.728610e-18 2.465879e-19 -15.12081638 1.645591e-39
## infectable:X2  1.159898e-18 2.302591e-18   0.50373571 6.147873e-01
## infectable:X3 -3.316157e-18 2.631849e-18  -1.26001059 2.085668e-01
## infectable:X4 -2.701725e-19 3.111465e-18  -0.08683128 9.308589e-01
## infectable:X5  4.115382e-19 3.163498e-18   0.13008960 8.965758e-01
## infectable:X6  5.272699e-19 3.286162e-18   0.16045159 8.726247e-01
## infectable:X7 -8.504031e-19 4.105833e-18  -0.20712073 8.360449e-01
```

```
model.2.transformed = lm(result~0+X1+X2+X3+X4+X5+X6+X7,data = train.transformed.data)
summary(model.2.transformed)$coef
```

```
##         Estimate   Std. Error     t value     Pr(>|t|)
## X1 -1.201885e-09 7.695969e-11 -15.6170672 1.939665e-41
## X2  3.785136e-10 6.959084e-10   0.5439129 5.868732e-01
## X3 -1.087814e-09 7.942790e-10  -1.3695621 1.717663e-01
## X4 -1.209604e-10 9.486712e-10  -0.1275051 8.986192e-01
## X5  1.366238e-10 9.642582e-10   0.1416880 8.874140e-01
## X6  1.514589e-10 9.980781e-10   0.1517505 8.794776e-01
## X7 -2.407262e-10 1.246092e-09  -0.1931849 8.469345e-01
```

Refined Models It seems pretty clear based on the P value of coefficients that all except X1 can be removed from the models, because some of the predictors may be shadowing others I will remove them one by one until X1 is left, or others are become significant.

```
model.1.transformed.2 = lm(result~0+X1:infectable,data = train.transformed.data)
summary(model.1.transformed.2)$coef
```

```
##                     Estimate   Std. Error  t value     Pr(>|t|)
## X1:infectable -3.729862e-18 2.450423e-19 -15.2213 4.664065e-40
```

```
anova(model.1.transformed,model.1.transformed.2)
```

```
## Analysis of Variance Table
##
## Model 1: result ~ 0 + X1:infectable + X2:infectable + X3:infectable +
##     X4:infectable + X5:infectable + X6:infectable + X7:infectable
## Model 2: result ~ 0 + X1:infectable
##   Res.Df        RSS Df   Sum of Sq      F Pr(>F)
## 1    326 2.0346e-05
## 2    332 2.0467e-05 -6 -1.2072e-07 0.3224 0.9251
```

```
model.2.transformed.2 = lm(result~0+X1,data = train.transformed.data)
summary(model.2.transformed.2)$coef
```

```
##         Estimate   Std. Error   t value    Pr(>|t|)
## X1 -1.201885e-09 7.652595e-11 -15.70558 5.89424e-42
```

```
anova(model.2.transformed,model.2.transformed.2)
```

```
## Analysis of Variance Table
##
## Model 1: result ~ 0 + X1 + X2 + X3 + X4 + X5 + X6 + X7
## Model 2: result ~ 0 + X1
##   Res.Df        RSS Df   Sum of Sq     F Pr(>F)
## 1    326 1.9800e-05
## 2    332 1.9938e-05 -6 -1.3776e-07 0.378 0.8928
```

After removing all the Xs none were significant except X1 for both models.

Now I will make a full interaction model between X1 and the infectable population because it seems like the next logical step.

```
interaction.model= lm(result~0+X1*infectable,data = train.transformed.data)
summary(interaction.model)$coef
```

```
##                    Estimate   Std. Error   t value      Pr(>|t|)
## X1             4.988934e-09 6.953140e-10  7.175081  4.795900e-12
## infectable     8.160019e-13 1.251873e-14 65.182494 1.548584e-190
## X1:infectable -1.965768e-17 2.198576e-18 -8.941099  2.801987e-17
```

Analysis of New Models In order to get a better idea of which model is good we'll compare them by thier test RSS values again

```
test.transformed = transformdata(test.data)
test.transformed$result = test.data$result

RSS.transformed.1 = RSS(model.1.transformed.2,test.transformed)
RSS.transformed.2 = RSS(model.2.transformed.2,test.transformed)
RSS.interaction.model = RSS(interaction.model,test.transformed)
```

Transformed Model 1 RSS	Transformed Model 2 RSS	Interaction Model RSS
6.398158×10^{-6}	6.2078528×10^{-6}	3.2013926×10^{-7}

Surprisingly the interaction model has a comparable RSS value as the original model 1 and model 2, whose RSS values are 2.8044446×10^{-7} and 2.303237×10^{-7} respectively, even though it has 5 fewer predictors. Both of the transformed models 1 and 2 have a much higher RSS value than their counterparts. I can think of two reasons for this, it could be that the test data is so similar to the training data that the larger model is more effective in predicting the test data. The other possibility is that there is some extra bit of predictability that can be picked up from the other principle components. In either case adding the infectable population to the model makes up for the loss in precision of taking the first principle component, so that will be the model I choose, with one modification. The full interaction model includes the X1 term by itself which means that it would be possible for there to be no infectable people and the model still predict that there would be more cases, the simple fix is to remove the X1 term. It would be unlikely that the X1 term has much of an effect on the model considering that in all the previous versions comparing the interaction with the cases and the cases by themselves did not have a big difference.

```
interaction.model.2 = lm(result~0+ X1:infectable +infectable,
                         data = train.transformed.data)
summary(interaction.model.2)$coef
```

```
##                    Estimate   Std. Error   t value      Pr(>|t|)
## infectable     7.635912e-13 1.091450e-14  69.96120 2.168915e-200
## X1:infectable -3.888134e-18 6.180655e-20 -62.90812 3.508865e-186
```

```
anova(interaction.model,interaction.model.2)
```

```
## Analysis of Variance Table
```

```
##
## Model 1: result ~ 0 + X1 * infectable
## Model 2: result ~ 0 + X1:infectable + infectable
##   Res.Df        RSS Df   Sum of Sq      F    Pr(>F)
## 1    330 1.1215e-06
## 2    331 1.2964e-06 -1 -1.7496e-07 51.482 4.796e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
RSS.interaction.model.2 = RSS(interaction.model.2,test.transformed)
```

Interaction Model 1 RSS	Interaction Model 2 RSS
3.2013926×10^{-7}	3.065355×10^{-7}

The anova analysis seems to say that the X1 predictor is statistically significant, but the test RSS between the two models is almost identical. For this reason I will use the second version of the model, the one without X1. When using the test data to compare they are very similar, and the benefit of there not being anymore cases once there's no one left to infect is a big plus.

Part 3: Predicting Into the Future

Vaccination Model In order to make future predictions the model needs two main pieces of data, the new cases over the last seven days, and the infectable population of the day before. Because the model is predicting new cases those data points are not a problem to obtain. The infectable population is more complicated, it is made of three parts the total population, the total cases, and the number of people fully vaccinated. The total number of cases can be calculated, but the number of people vaccinated need to be predicted.

A simple model using the logit scale will be used for this, this essentially makes the assumption that once the vaccine roll out starts there are no hiccups along the way, of course we know this wasn't entirely true. It is assumed that 25% are hessitant to the vaccine and will not recieve it even if given the chance.

```
toVaccineLogit <- function(peopleVacinated,totalPop){
  log(((totalPop*(1-params$fractionVaccineHessitant))/peopleVacinated)-1)
}

fromVaccineLogit <- function(logit,totalPop){
  (totalPop*(1-params$fractionVaccineHessitant))/(exp(logit) + 1)
}

not_zero = covid_data$people_fully_vaccinated > 0;

total_people_vacinated = covid_data[not_zero,]$people_fully_vaccinated;

vaccine.prediction.data = data.frame(
  result = toVaccineLogit(total_people_vacinated,
                  covid_data$population[not_zero]),
  index = (1:length(not_zero))[not_zero]
);
vaccine.predictor = lm(result~index, data = vaccine.prediction.data)
```

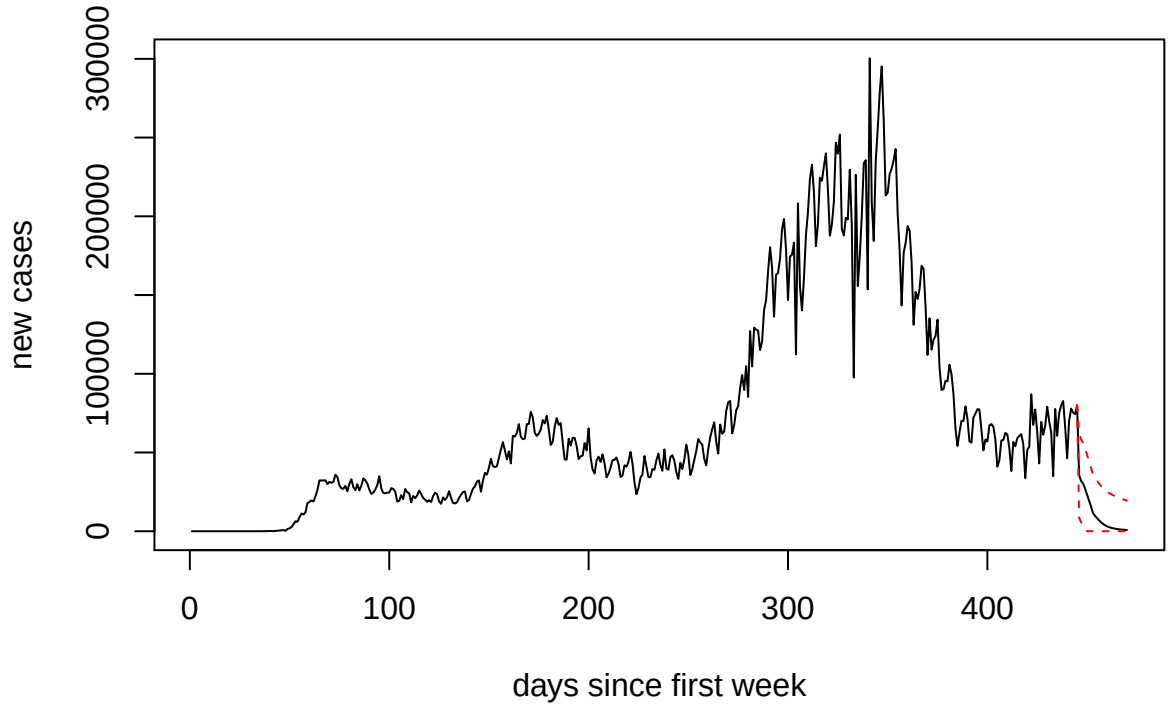
Because predicting the amount of vaccines handed out is not the purpose of this project, not alot of time is spent on creating this model.

Prediction Function These functions predict one day into the future, using the predicted value, and the upper and lower bounds. Applying this function repeatedly will yield a prediction n days into the future, though the more it is applied the less likely that it will be accurate.

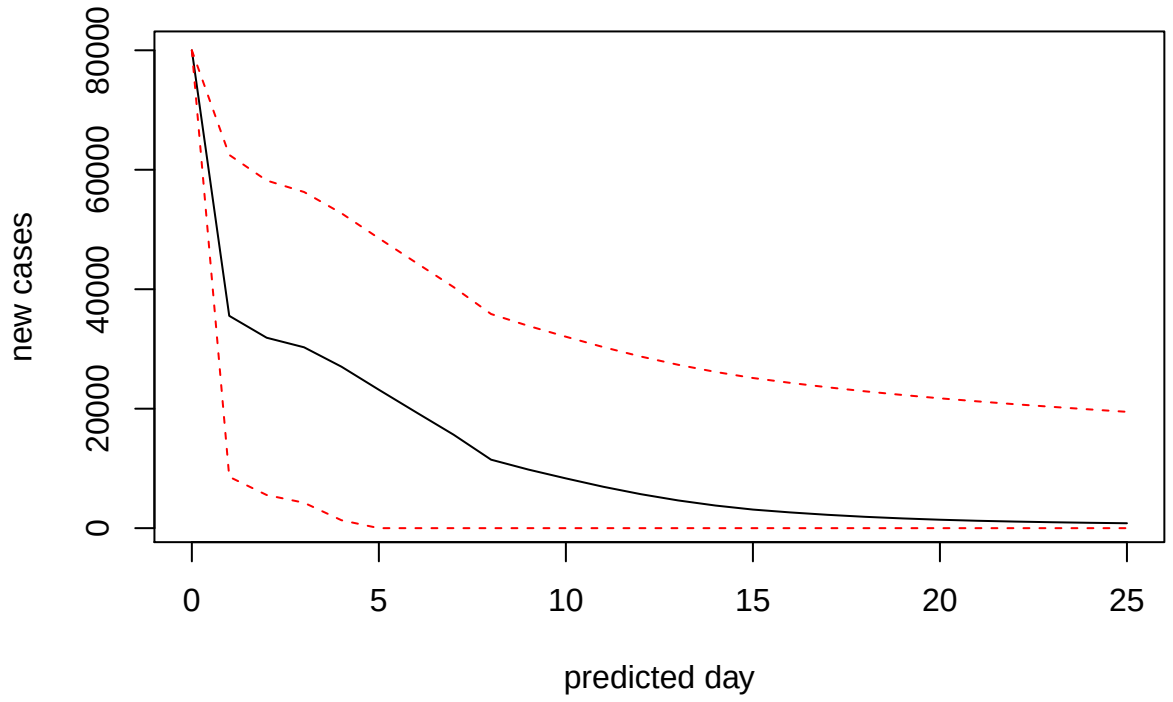
```
appendNewCases <- function(data, oddsNewCases, vaccine.pred){
  prev = data[nrow(data),]
  new_cases = max(c(from.odds(oddsNewCases,prev$infectable),0));
  total_cases = prev$total_cases + new_cases;
  infectable = prev$total_pop - total_cases - vaccine.pred;

  cases_back_1 = new_cases;
  cases_back_2 = prev$cases_back_1;
  cases_back_3 = prev$cases_back_2;
  cases_back_4 = prev$cases_back_3;
  cases_back_5 = prev$cases_back_4;
  cases_back_6 = prev$cases_back_5;
  cases_back_7 = prev$cases_back_6;

  newrow = c(prev$total_pop,total_cases,infectable,new_cases,cases_back_1,
             cases_back_2,cases_back_3,cases_back_4,cases_back_5, cases_back_6,
             cases_back_7);
  data[nrow(data)+1,] = newrow;
  data
}

nextday.data <- function(lower,upper,expected){
  expect.transformed = transformdata(expected);
  low.transformed = transformdata(lower);
  up.transformed = transformdata(upper);

  expect.pred = predict(interaction.model.2,newdata =
            expect.transformed[nrow(expect.transformed),])
  low.pred = predict(interaction.model.2,newdata =
            expect.transformed[nrow(low.transformed),],interval = "predict")[2]
  up.pred = predict(interaction.model.2,newdata =
            expect.transformed[nrow(up.transformed),],interval = "predict")[3]

  vaccine.pred = predict(vaccine.predictor,newdata =
          data.frame(index = c(nrow(expected) + 1)))

  vaccine.pred = fromVaccineLogit(vaccine.pred,
                                  expected[nrow(expect.transformed),]$total_pop)

  list(expected = appendNewCases(expected,expect.pred,vaccine.pred),
    lower = appendNewCases(lower,low.pred,vaccine.pred),
    upper = appendNewCases(upper,up.pred,vaccine.pred));
}
```

Predict Into The Future By applying the model to itself 25 times we can get a possible idea of what the next 25 days look like in terms of covid cases.

```
expected = data.frame(
          total_pop = covid_data$population[7:length],
          total_cases = covid_data$total_cases[7:length],
          infectable = covid_data$infectable_pop[7:length],
          new_cases = covid_data$new_cases[7:length],
          cases_back_1 = covid_data$new_cases[7:length],
          cases_back_2 = covid_data$new_cases[6:(length-1)],
          cases_back_3 = covid_data$new_cases[5:(length-2)],
          cases_back_4 = covid_data$new_cases[4:(length-3)],
          cases_back_5 = covid_data$new_cases[3:(length-4)],
          cases_back_6 = covid_data$new_cases[2:(length-5)],
          cases_back_7 = covid_data$new_cases[1:(length-6)])

realDays = nrow(expected)

lower = expected;
upper = expected;

for (d in (1:params$futureDays)){
  a = nextday.data(lower,upper,expected)
  lower = a$lower;
  expected = a$expected;
  upper = a$upper;
}

len = nrow(expected)
X1 = 1:len;
X2 = realDays:len;
plot(X1,expected$new_cases,type = "l",lty = 1,xlab = "days since first week",ylab = "new cases")
lines(X2,lower$new_cases[X2],type = "l",lty = 2,col = "red")
lines(X2,upper$new_cases[X2],type = "l",lty = 2,col = "red")
```




```
Xf = 0:params$futureDays;

plot(Xf,expected$new_cases[X2],type = "l",lty = 1,xlab = "predicted day",ylab = "new cases")
lines(Xf,lower$new_cases[X2],type = "l",lty = 2,col = "red")
lines(Xf,upper$new_cases[X2],type = "l",lty = 2,col = "red")
```



In my personal opinion this model may be a bit too optimistic, I doubt that we will hit about 0 cases per day in about two weeks. While the upper bound seems more realistic I still think that is too optimistic.

Possible Future Improvments

While working on this I have thought of several future improvements that could be done to this model. Which I will discuss here

1) New Cases Vs. New Cases Smoothed Perhaps the most naive mistake of mine was using the raw new case numbers rather than the smoothed version, which is readily provided in the OWID data-set. If I were to redo this I would use the new cases smoothed, because it removes a level of randomness that is in the new cases data.

2) Data Processing A big problem that i didn't think would have much of an effect was that I never standardized any of the data. Because the numbers were so big, and the infectable population was so much greater than the new cases, comparing different models was difficult, it also made it difficult to see which predictors the model was pushing towards zero. Standardizing the data would have fixed this.

3) The Model I assumed that there would be some sort of pattern within the previous week of case that would have helped predict the next day, I was wrong about that, and when I did PCA on the data the first eigen vector was almost just an average of the week, each entry was about -0.37. This was also the only statistically significant predictor after PCA. If I were to redo this I would take an average over the past 3-7 days, and then go further back to the next section of 3-7 days, and repeat that a few times. By doing this there might be a more significant trend between multi-day averages that the model would pick up on

4) Missing Vaccine Data There are quite a few data points missing when it comes to the vaccine data, this could be remedied by creating a more accurate vaccine prediction model, and filling in those blanks before any further data processing.

5) The First Month The first month or so of the covid pandemic only had 1 or 2 new cases each day, so the data is different to more recent data, where we tend to have thousands of new cases each day. For this reason The first month or two of data should be cut from the model, or somehow reduced in importance.