245 Project

SIR and SIRV models

Stanley Whitehouse

Throughout this section of the document the following differential equations are used:

$$\frac{dx}{dt} = -\beta xy$$
$$\frac{dy}{dt} = \beta xy - \gamma y$$
$$\frac{dz}{dt} = \gamma y$$

These are represented by the following code:

```
dxdt <- function(t, x, y) {
  return(-beta * x * y)
}
dydt <- function(t, x, y) {
  return(beta * x * y - gamma * y)
}
dzdt <- function(t, y) {
  return(gamma * y)
}
```

Part 1:

Question 1:

Three Assumptions are:

1. The model assumes that an individual cannot be reinfected after recovery. This doesn't align with biology where we know that immunity isn't always a permanent thing.
2. The SIR Model assumes that everyone has the same probability of being infected. This doesn't account for variations within the population. This isn't true in reality as somebody with an autoimmune disease will be more susceptible to the disease.
3. Since the model only allows you to be susceptible, infected or recovered there is no way to track fatalities which are crucial when dealing with real diseases.

Question 2:

We know that $\frac{dx}{dt} + \frac{dy}{dt} + \frac{dz}{dt} = 0$, therefore we can assume that $x(0), +y(0), +z(0) = x(t) + y(t) + z(t) = N$, where N is the total population size and $0 \leq t \leq T$. Rearranging this three-dimensional system of ODEs $\frac{dx}{dt} + \frac{dy}{dt} + \frac{dz}{dt} = 0$ we get $\frac{dz}{dt} = -\left(\frac{dx}{dt} + \frac{dy}{dt}\right)$.

Therefore:

$$\gamma y = -\left(\frac{dx}{dt} + \frac{dy}{dt}\right)$$
$$\frac{dy}{dt} = -\frac{dx}{dt} - \gamma y$$
$$\frac{dy}{dt} = \beta xy - \gamma y$$

From this, we obtain the 2-dimensional system:

$$\frac{dx}{dt} = -\beta xy \tag{1}$$
$$\frac{dy}{dt} = \beta xy - \gamma y \tag{2}$$

as required. □

Question 3:

a:

I have used the RK4 method to solve this system of ODE's because it offers a good balance between order of accuracy and cost of computation. The RK4 method is the highest order explicit Runge-Kutta method that requires the same number of steps as the order of accuracy. The main weakness of the RK4 method is it's inability to solve "rigid" equations. This isn't a problem with our SIR model which has a lot of flexibility. The local truncation error of the RK4 method is $O(h^3)$, the step error is $O(h^5)$ with the total error $O(h^4)$.

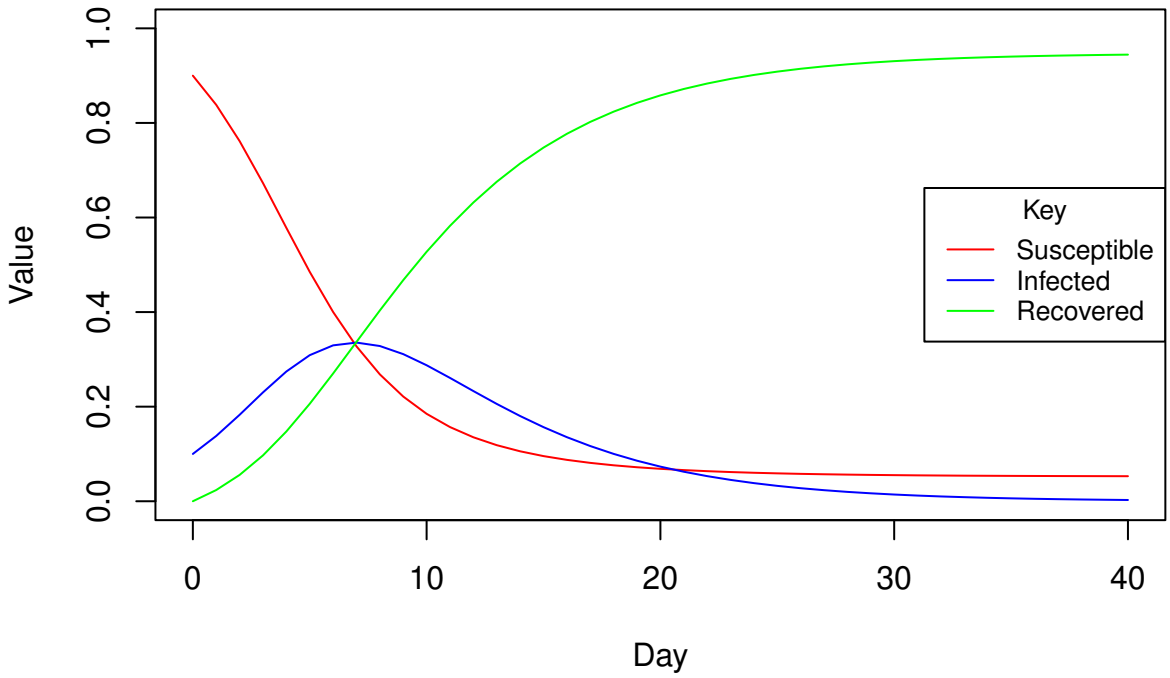
This code was based off a blog post[1] and I have been influenced by it throughout this project.

```
RK4SIR <- function(T, beta, gamma, x0, y0, h) {

  N <- x0 + y0  # Fixed population

  W <- floor(T/h) # Total number of steps needed

  x <- c(x0, rep(0, W)) # Generating empty vectors with x[1]=x0 and y[1]=y0
  y <- c(y0, rep(0, W))
  z <- c(rep(0, W+1))

  for (i in 1:W) {
    # Setting our values from the vector
    xi <- x[i]
    yi <- y[i]
     #zi <- z[i]

     # Generating K_0 terms
    x.k0 <- h * dxdt(i, xi, yi)
    y.k0 <- h * dydt(i, xi, yi)
     #z.k0 <- h * dzdt(i, yi)

     # Generating K_1 terms
    x.k1 <- h * dxdt(i + h / 2, xi + 1 / 2 * x.k0, yi + 1 / 2 * y.k0)
    y.k1 <- h * dydt(i + h / 2, xi + 1 / 2 * x.k0, yi + 1 / 2 * y.k0)
```

```
    #z.k1 <- h * dzdt(i + h / 2, yi + 1 / 2 * y.k0)

    # Generating K_2 terms
   x.k2 <- h * dxdt(i + h / 2, xi + 1 / 2 * x.k1, yi + 1 / 2 * y.k1)
   y.k2 <- h * dydt(i + h / 2, xi + 1 / 2 * x.k1, yi + 1 / 2 * y.k1)
    #z.k2 <- h * dzdt(i + h / 2, yi + 1 / 2 * y.k1)

   # Generating K_3 terms
   x.k3 <- h * dxdt(i + h, xi + x.k2, yi + y.k2)
   y.k3 <- h * dydt(i + h, xi + x.k2, yi + y.k2)
    #z.k3 <- h * dzdt(i + h, yi + y.k2)

   # Applying Simpsons rule to calculate the next entry in the vector
   x[i + 1] <- xi + 1 / 6 * (x.k0 + 2 * x.k1 + 2 * x.k2 + x.k3)
   y[i + 1] <- yi + 1 / 6 * (y.k0 + 2 * y.k1 + 2 * y.k2 + y.k3)
    #z[i + 1] <- zi + 1 / 6 * (z.k0 + 2 * z.k1 + 2 * z.k2 + z.k3)

    # This for loop runs W times and generates vectors of x and y

    # activating the code for z above generates independent vector
 }

  # Using the fixed population to calculate z
 z <- N - x - y


 xh <- x[c(seq(1,W+1,1/h))] # the values that correspond to each day
 yh <- y[c(seq(1,W+1,1/h))]
 zh <- z[c(seq(1,W+1,1/h))]


  # Plotting the calculated values at each day
 plot( 0:T, xh, type="l", col = "red",ylim = c( 0, 1), ylab = "Value",
       xlab = "Day")
 lines( 0:T, yh, col = "blue")
 lines( 0:T, zh, col = "green")

 # Adding a key
 legend( "right", legend = c( "Susceptible", "Infected","Recovered"),
         col=c( "red", "blue", "green"), lty = rep( 1, 3), cex = 0.8,
         title ="Key")

 # Creating a dataframe rounding to 5.d.p
 results <- data.frame("Day" = 0:T, "Susceptible" = round(xh, digits=5),
                       "Infectious" = round(yh, digits=5),
                       "Recovered" = round(zh, digits=5))

 # Returning every 5th day
 return(results[c(seq(1,T+1,5)),])
}
```

b:

Letting T=40, $\beta = 0.6$, $\gamma = 0.2$, x0=0.9, y0=0.1 and $h = 0.1$ we generate the following plot and data frame:

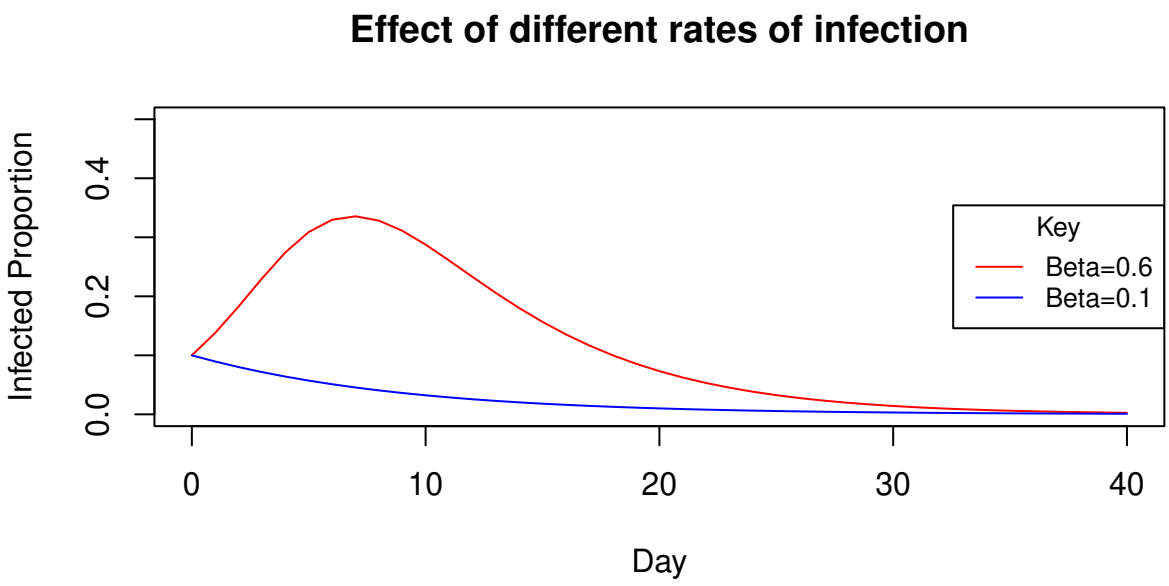



```
##    Day Susceptible Infectious Recovered
## 1    0     0.90000    0.10000   0.00000
## 6    5     0.48505    0.30890   0.20605
## 11  10     0.18491    0.28758   0.52750
## 16  15     0.09548    0.15669   0.74783
## 21  20     0.06855    0.07319   0.85825
## 26  25     0.05896    0.03253   0.90851
## 31  30     0.05517    0.01419   0.93064
## 36  35     0.05360    0.00614   0.94025
## 41  40     0.05294    0.00265   0.94441
```

Question 4:

a:

Using the same T, γ, x0, y0 and h values as in question 3 we can generate the following plot of the infected proportion of the population at different β values:



As you can see from above; with a β value of 0.1, the infected proportion of the population quickly decreases for the first few days. The rate of this decrease slows over time and eventually tends to zero. This makes sense as our β value represents the rate of infection and our γ value represents how long it takes for a infected individual to move from infected to recovered. Logically, if γ is greater than β, our rate of infection $\left(\frac{\beta}{\gamma}\right)$ is less than one and subsequently the infected proportion falls.

Conversely, with a β value of 0.6, the infected proportion of the population rapidly increases with a peak value of ≈ 0.33553 at 7 days. After this period of time the proportion of infected individuals decreases rapidly, eventually slowing down and tending to zero. This makes sense with the assumptions of our model. As the isn't any way for people in the recovered category to move back into the susceptible population, after a short time with a high $\frac{\beta}{\gamma}$ value, the entire population moves out of the susceptible category. This leads to the natural decline of the virus as there are no new susceptible people to infect.

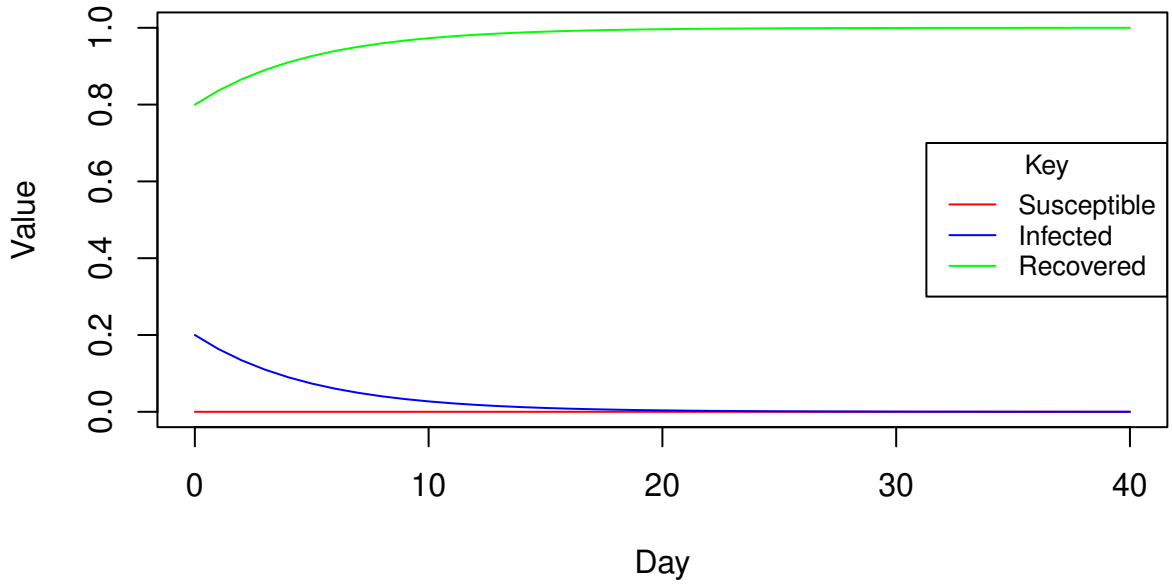
b:

In this question we are using $T = 40$, $\beta = 5$, $\gamma = 0.2$, x0= 0.9, y0= 0.1 and $h = 7$.

```
##   Week   Susceptible     Infectious Recovered
## 1    0  9.000000e-01   1.000000e-01     0.000
## 2    1  1.123114e+09  -1.123111e+09 -3242.821
## 3    2 3.778172e+163 -3.778172e+163     0.000
## 4    3           NaN            NaN       NaN
## 5    4           NaN            NaN       NaN
## 6    5           NaN            NaN       NaN
```

Here we run into a crucial problem. When we consider $h > 1$ the RK4 method isn't suitable for solving our system of ODE's. This is because when $h > 1$ the susceptible population increases (unless $x(0) = 0$). In the example above our number of susceptible individuals tends to ∞ and the number of infected people tends to $-\infty$.

c:

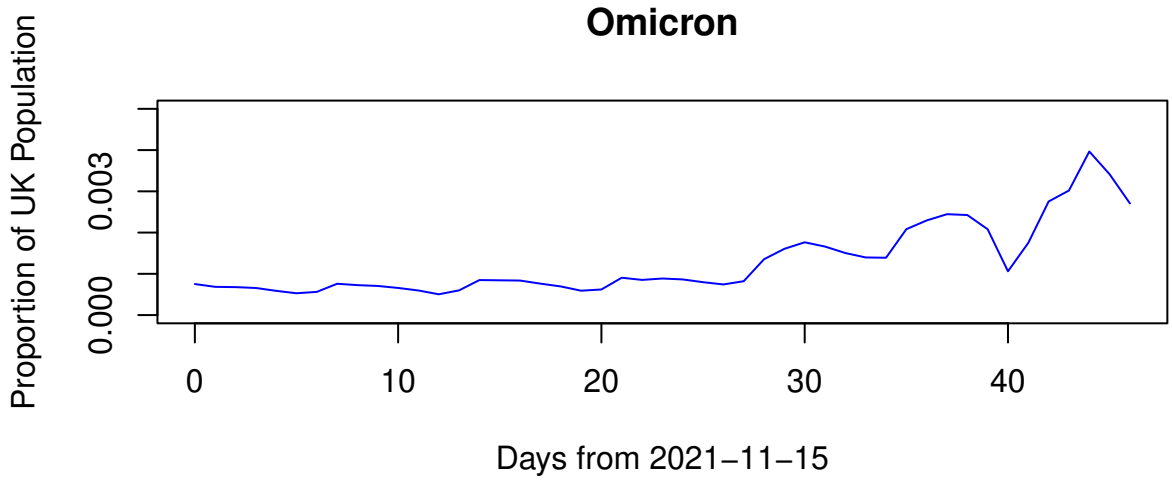


This aligns with our calculated results on paper. At every $0 \leq t \leq T$, we have $x(t) = 0$ as $x.k_0, x.k_1, x.k_2$ and $x.k_3$ are always zero. The value of y is also quite accurate to our calculated results.

Part 2:

Question 5:

a:



b:

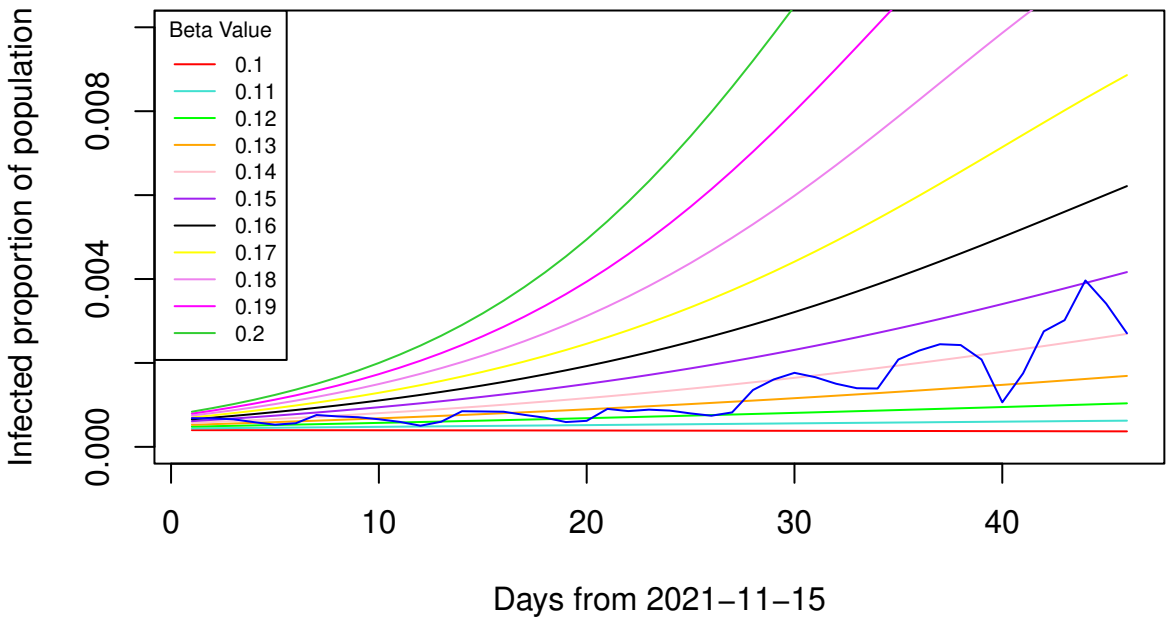
To accurately assess which β value is the closest to that of Omicron we first need to alter our code slightly. We are now interested in the new number of daily infections. We know that the number of new infected people at day $0 \leq t \leq T$ is simply $x(t-1) - x(t)$. This is due to our assumption that the only way to move from the susceptible category is to get infected.

Modifying our code from Question 3 to output a vector of the new infected proportion of the whole population allows us to test multiple values of β on the same graph.

Replacing our return value with this new sequence:

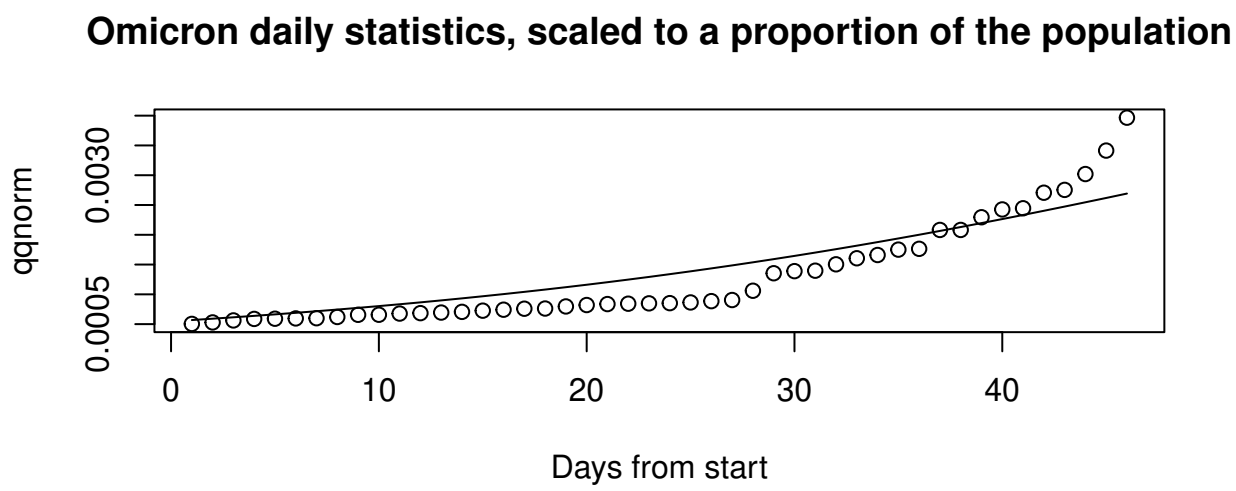
```
newY <- c(rep(0,T+1))
newY[1] <- y0
for(i in 1:T){
  newY[i+1] <- x[i]-x[i+1]
}
return(newY)
```

We can plot multiple against multiple values of β on the same graph. I decided to omit the first day as we had y0:= 0.004 for all the values which is far away from our true value at "Day 0".



c:

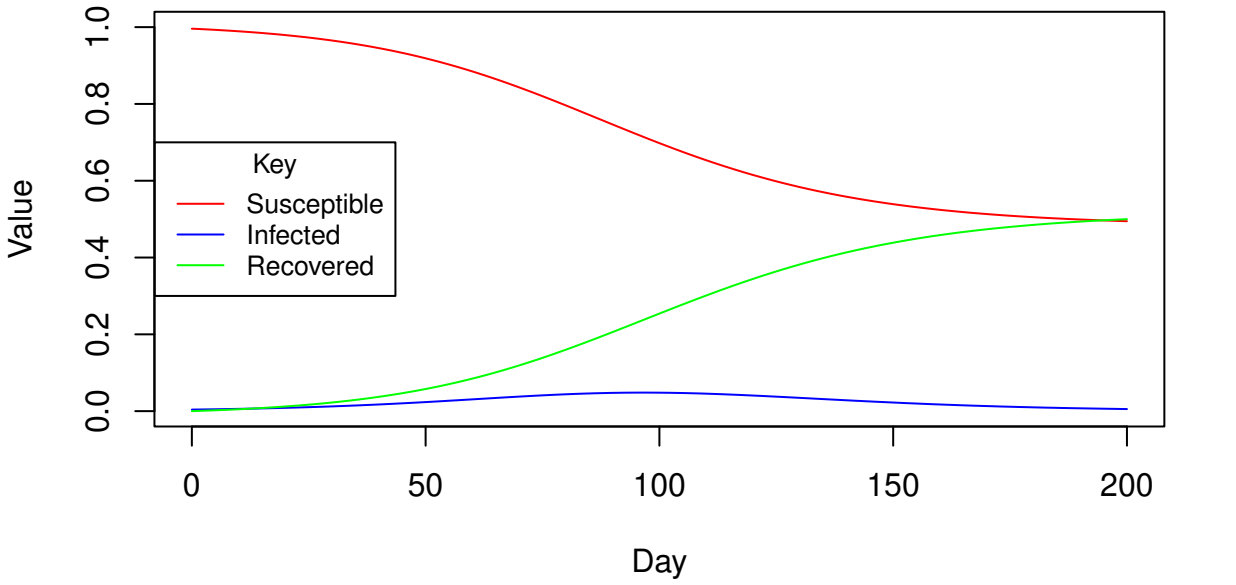
From this we can see that $\beta = 0.14$ gives the most accurate result. It isn't perfect but we can see that, for an approximation, it's not too bad.



For the graphs above I used a qqplot for the qqnorm of our Omicron data (scaled to proportion of the population). I added a line with the qqnorm of our SIR model evaluated at $\gamma = 0.1$, $\beta = 0.14$ with x0= 0.996 and y0= 0.004.

The reason I plotted the values in this was was to check the fit against the qqnorm of the true omicron values. As is evident from the plot, the lines don't match perfectly and there is deviance throughout. Regardless, the line is a relatively close fit and far closer than any other value of β from the given selection. From the experiments I carried out, I believe the true value of $\beta \in [0.13, 0.15]$, however, it fluctuates based on the time frame used to evaluate the data.

d:



e:

The graph in part **d** shows the spread of infection over 200. It is clear from the plot and the data that the infection spreads slowly, with $x(56) < 0.9$ indicating that it took 56 days for over 10% of the population to become infected. After this point the susceptible proportion rapidly falls with $x(90) < 0.75$ indicating it took roughly 3 months for 25% of the population to become infected. This is right around the peak of the infected proportion .The highest y value recorded is $y(97) = 0.04824$. After this time the rate of decrease in the susceptible population slows and $x(189) < 0.5$.

The recovered proportion at time $0 \leq t \leq 200$ is calculated by $1 - x(t) - y(t)$ so its behavior is similar to the inverse of $x(t)$. $z(65) > 0.1$ indicates it took 65 days for over 10% of the population to contract and then get cured from the disease. $z(100) > 0.25$ and $z(200) \approx 0.5$.

Part 3

Throughout this section of the code we will use the following system of ODE's:

$$\begin{aligned}
\frac{dx}{dt} &= -\beta xy - \mu x \\
\frac{dy}{dt} &= \beta xy - \gamma y \\
\frac{dz}{dt} &= \gamma y \\
\frac{dv}{dt} &= \mu x
\end{aligned}$$

These are represented the following code:

```
dxdt1 <- function(t, x, y) {
  return(-beta * x * y - mu * x)
}
dydt <- function(t, x, y) {
  return(beta * x * y - gamma * y)
}
dzdt <- function(t, y) {
  return(gamma * y)
}
dvdt <- function(t,x){
  return(mu * x)
}
```

Question 6:

a:

Keeping the majority of the code the same as in question 3, we can extend our model to a system of four ODE's very easily. Simply applying the same RK4 method as in question 3, but with revised differential equations and applying Simpsons rule as before, generates a vector of results.

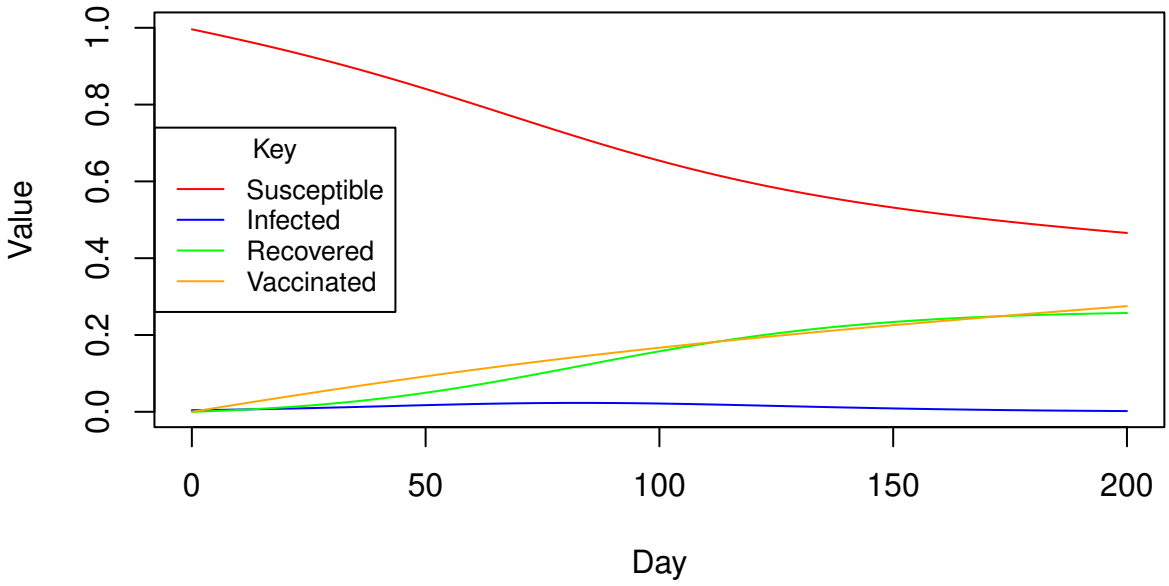
Since the population is still closed we can still calculate our recovered population by simple subtraction. However, I have left the code for the values of z in the answer to allow for more complex calculations in the future. This code should theoretically work for any set of ordered differential equations.

```
RK4SIRV <- function(T, mu, beta, gamma, x0, y0, v0, h) {
  options(scipen=999) # Removes scientific notation from final answer

    N <- x0 + y0 + v0 # Fixed population

    W <- floor(T/h) # Number of steps required

    x <- c(x0, rep(0, W)) # Generating empty vectors of length W with x[1]=x0,
    y <- c(y0, rep(0, W)) # y[1]=y0 and v[1]=v0
    v <- c(v0, rep(0,W))

    z <- c(rep(0, W+1))  # Can slightly edit code to allow for z0 not equal to 0

    for (i in 1:W) {
      # Setting a variable value, initially xi = x0, yi = y0, vi = z0
      xi <- x[i]
      yi <- y[i]
      vi <- v[i]
      #zi <- z[i]

      #Calculating K_0 terms
      x.k0 <- h * dxdt1(i, xi, yi)
      y.k0 <- h * dydt(i, xi, yi)
      v.k0 <- h * dvdt(i, xi)
      #z.k0 <- h * dzdt(i, yi)

      #Calculating K_1 terms
      x.k1 <- h * dxdt1(i + h / 2, xi + 1 / 2 * x.k0, yi + 1 / 2 * y.k0)
      y.k1 <- h * dydt(i + h / 2, xi + 1 / 2 * x.k0, yi + 1 / 2 * y.k0)
      v.k1 <- h * dvdt(i + h / 2, xi + 1 / 2 * x.k0)
      #z.k1 <- h * dzdt(i + h / 2, yi + 1 / 2 * y.k0)

      #Calculating K_2 terms
      x.k2 <- h * dxdt1(i + h / 2, xi + 1 / 2 * x.k1, yi + 1 / 2 * y.k1)
      y.k2 <- h * dydt(i + h / 2, xi + 1 / 2 * x.k1, yi + 1 / 2 * y.k1)
      v.k2 <- h * dvdt(i + h / 2, xi + 1 / 2 * x.k1)
      #z.k2 <- h * dzdt(i + h / 2, yi + 1 / 2 * y.k1)

      #Calculating K_3 terms
      x.k3 <- h * dxdt1(i + h, xi + x.k2, yi + y.k2)
      y.k3 <- h * dydt(i + h, xi + x.k2, yi + y.k2)
      v.k3 <- h * dvdt(i + h, xi + x.k2)
      #z.k3 <- h * dzdt(i + h, yi + y.k2)

      #Applying Simpsons rule to generate the next term in the vector
      x[i + 1] <- xi + 1 / 6 * (x.k0 + 2 * x.k1 + 2 * x.k2 + x.k3)
      y[i + 1] <- yi + 1 / 6 * (y.k0 + 2 * y.k1 + 2 * y.k2 + y.k3)
      v[i + 1] <- vi + 1 / 6 * (v.k0 + 2 * v.k1 + 2 * v.k2 + v.k3)
      #z[i + 1] <- zi + 1 / 6 * (z.k0 + 2 * z.k1 + 2 * z.k2 + z.k3)

    # This for loop runs W times and generates vectors of x, y and v.

    # Activating the code for z above you can calculate z independently
```

```
    }

    #Calculating values of z from our fixed population
    z <- N - x - y - v

    # Selecting the values of our vectors that correspond to a day
  xh <- x[c(seq(1,W+1,1/h))]
  yh <- y[c(seq(1,W+1,1/h))]
  zh <- z[c(seq(1,W+1,1/h))]
  vh <- v[c(seq(1,W+1,1/h))]


     #Plotting our values of x,y,z,v along the y-axis and the days on the x-axis
    plot(seq(0, T, 1), xh, type = "l", col = "red", ylim = c( 0, 1),
         ylab = "Value", xlab = "Day")
    lines(seq(0, T, 1), yh, col = "blue")
    lines(seq(0, T, 1), zh, col = "green")
    lines(seq(0, T, 1), vh, col = "orange")

     #Adding a key to the plot
    legend("left", legend = c( "Susceptible", "Infected", "Recovered",
                               "Vaccinated"),
           col = c( "red", "blue", "green", "orange"), lty = rep(1, 4) ,
           cex = 0.8, title = "Key")

      #Putting the results into a data frame, rounding to 5.d.p
     results <- data.frame(Day = 0:T, "Susceptible" = round( xh, digits = 5),
                           "Infectious" = round( yh, digits = 5),
                           "Recovered" = round(zh, digits = 5),
                           "Vaccinated" = round( vh, digits = 6))

    # Can be activated if we wanted to see data
    # return(results[c(seq(1, T+1, 1)),])
}
```

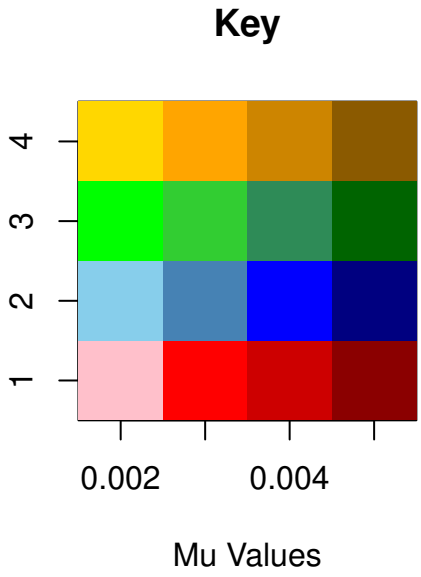
b:

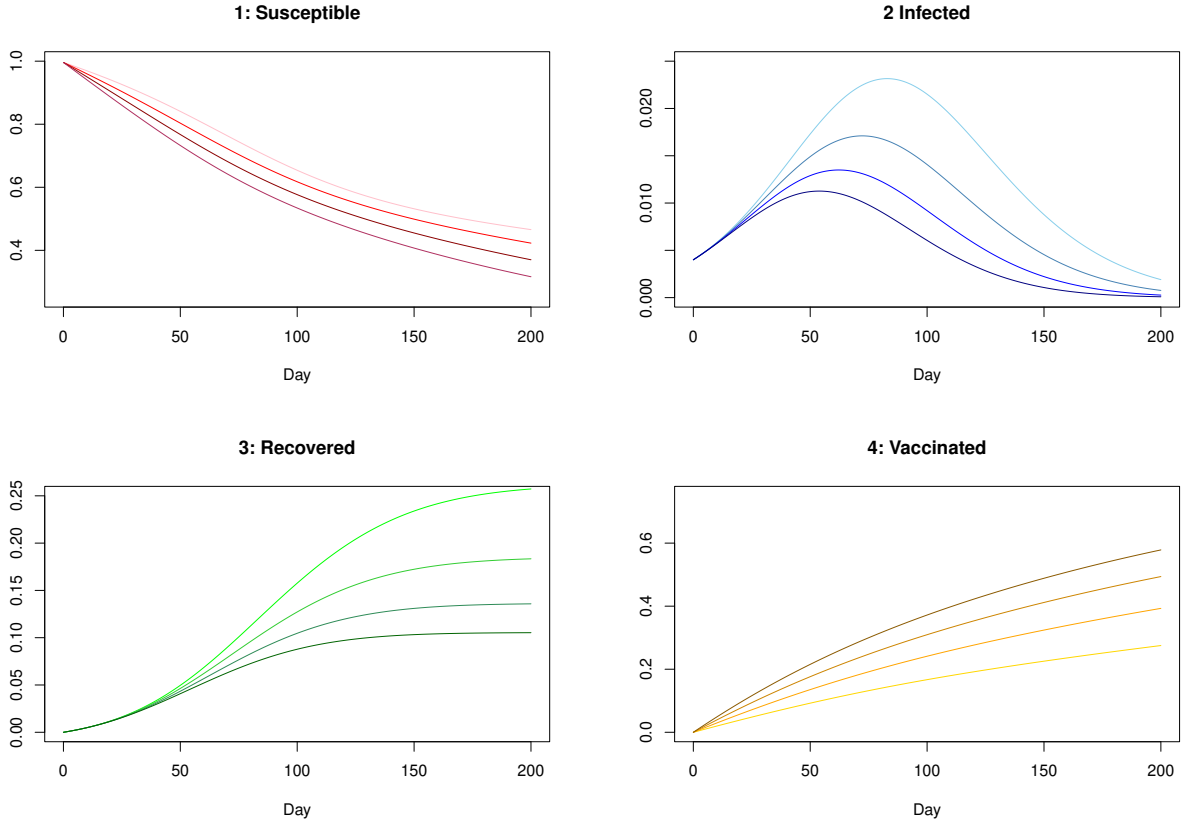
Setting $T = 200$, $\mu = 0.002$, $\beta = 0.14$, $\gamma = 0.1$,, x0= 0.996, y0= 0.004, v0= 0 and $h = 1$ we generate the following plot:



c:

To start this question I slightly edited my code to output a vector of either S,I,R or V at each day. I then plotted these values on a single graph to compare the different values at a range of mu. I changed limy on each plot to make some of the more intricate change easier to see:





As we can see from the plots above, increasing the vaccination rate has an effect on every part of our model.

As you would expect, increasing the vaccination rate increases the rate at which the susceptible proportion of the population falls. This makes intuitive sense as now a second way to move from the susceptible category. Mathematically we can see that $\forall \mu > 0$, $-\beta xy - \mu x < -\beta xy$ so the rate of the decrease of x is faster.

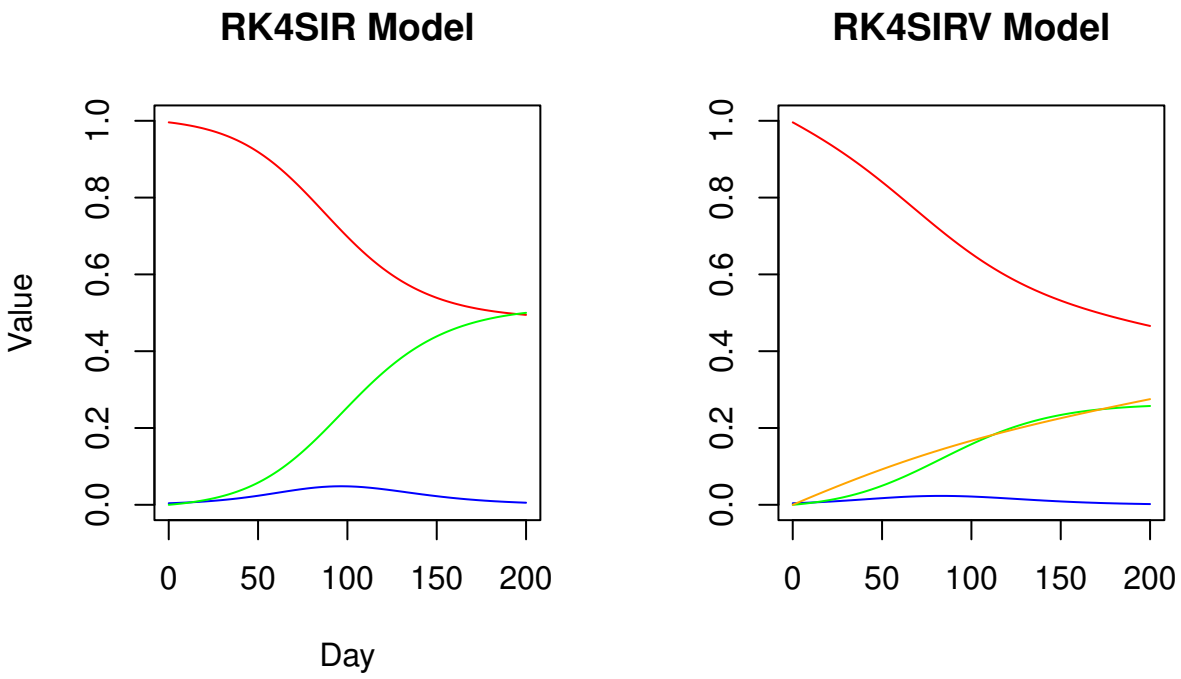
On the graph of the infected proportion we can clearly see that increasing the rate of vaccination decreases the number of infectious people at all times. Most strikingly it decreases the peak of the infection and shifts the peak to the left. This "flattening" of the infected proportion at higher mu values also fits with my knowledge of infectious diseases. When more people are vaccinated there are fewer susceptible people in the population this means the disease cannot spread as quickly.

Moving onto the recovered graph now. This graph, and that of the infected proportion, best depicts the real change that happens with an increased vaccination rate. As we can see, increasing the mu value leads to a dramatic decrease in the number of people in the recovered category. This also makes a lot of sense as the people that would've been in the recovered category are now vaccinated. As a result far fewer people die or even get the virus in the first place with an increased vaccination rate.

Finally the graph of the vaccinated. This graph makes intuitive sense. By increasing the rate at which people are getting vaccinated, you are increasing the number of people vaccinated at every interval of time.

d:

letting $\beta = 0.14$,$\gamma = 0.1$, $\mu = 0.002$, x0= 0.996, y0= 0.004, v0= 0, $h = 1$ we are comparing our RK4SIR model to our RK4SIRV model.



From the plots above the effect of vaccination is clear. I chose to use the default value of μ as the plots from part **c** made it clear that increasing the rate of vaccination only stands to increase the effect.

The main differences between the two graphs are in the infected and recovered proportions. As is clear from the graphs increasing the rate of vaccination decreases the infected proportion of the population at every time and as a result far fewer people are in the recovered category. This aligns with real world understanding as well. When a certain percentage of the population has immunity to an infection it has a harder time spreading through a population. This just goes to reinforce the effectiveness of vaccination.

References

[1] - Tony Tsai(2014) - 'https://blog.tonytsai.name/blog/2014-11-24-rk4-method-for- solving-sir-model/'

[2] - https://stackoverflow.com/questions/72495730/fourth-order-runge-kutta-for-sir-model

Feedback comments

Mark Summary:

Presentation: 9
Good programming practice: 7
Initiative: 7.
Question 1: 3
Question 2: 2
Question 3: 11
Question 4: 5
Question 5: 20
Question 6: 19

Total mark out of 100: 83

Feedback:

* Presentation [10]: Generally good. When printing tables it is a good idea to use kable as they then look much better. You should ensure your axis labels are always informative e.g. the y-axis label in Q3 reads "value" but it would have been more informative it had read something along the lines of "proportion".

* Good programming practice [10]: Generally good. More comments in your code would make it more readable, specifically you should state what the input variables are when you define a function. You should have included more of your code especially in Q4 and Q5.

* Initiative [10]: Nice to see an short introduction and a references section. Great work on Part 3.

* Question 1 [3]: Good.

* Question 2 [2]: Generally good however this is a proportions model so x+y+z=c where c=1, note that c is not the total population size.

* Question 3 [15]: Generally good - your RK algorithm would have been shorter if you had vectorised it, at present you treat each component separately. However your code is clearly working correctly. Unfortunately you loose marks as you didn't describe your results as requested by the question.

* Question 4 [10]: Part (a) Good. Part (b) Good but to be clear if the rate of of x,y,z is too large relative to the step size then the RK algorithm breaks down. Part (c) you haven't provided any workings or solutions to the equations and furthermore you haven't provided your code. However you are right the solutions should agree but you haven't shown this.

* Question 5 [20]: Well done.

* Question 6 [20]: Minor errors in code. Excellent work in part c).