Assignment 9

First Last

Stochastic, within-host dynamics

Here we will gain an understanding of how demographic stochasticity affects the within-host dynamics of viral infection. Remember that demographic stochasticity refers to the probabilistic events that befall individuals. In this case, we're talking about individual *virus particles* and individual *immune system cells*. We will therefore demonstrate how demographic stochasticity can influence: (1) the ability of the virus to establish in the host and cause a sustained infection, and (2) the time it takes for the virus to cause mortality.

Let us consider the within-host dynamics as described by the following mathematical model:

$$\frac{dV}{dt} = \phi V(1 - \frac{V}{K}) - \beta V Z$$
$$\frac{dZ}{dt} = -\beta V Z$$

We are assuming that the virus undergoes logistic growth, meaning that it is resource-limited. We also assume something very specific about the immune system. In this case, the immune system's growth is always negative in the presence of the virus. What we mean here is that the host individual starts with some amount of immune system cells, and those cells are depleted as the host fights off the infection. The host does not have a mechanism to increase the number of immune cells fast enough to meaningfully increase the number of immune cells in the timespan of infection. This type of model is used, for example, with insect hosts, because they have a limited, innate immune response to pathogens. In the code below, we will also assume that once the virus reaches some threshold ("death dose' ') the host will die of infection.

Task 1 (0 points)

Below I have developed a tau-leaping algorithm to simulate the model with demographic stochasticity. Study the code and make sure you understand what is going on.

First, we set up a function to advance the system one tau leap:

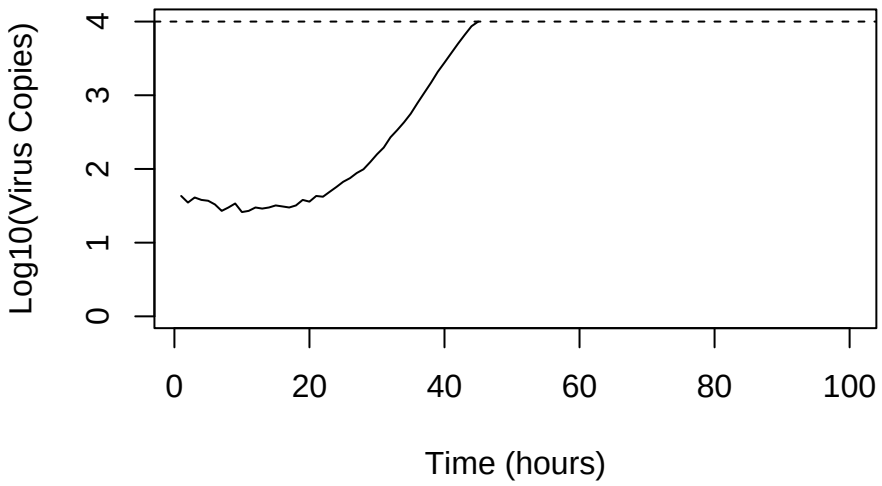
```
#######################################################
# FUNCTION #
#######################################################

tau_leap_1step = function(counter, tau, v_vec, z_vec, params){
  # Events:
  # 1. Replication of virus
  # 2. Depletion of virus = Depletion of immune system
  n_events = 2

  # Event probabilities:
  event_prob = vector(mode = "numeric", length = n_events)

  ## Replication:
  event_prob[1] = params$phi * v_vec[counter] * (1 - (v_vec[counter]/params$K))
  ## Depletion:
  event_prob[2] = params$beta * v_vec[counter] * z_vec[counter]

  # How many events of each type will occur over time period tau
  n_occur = vector(mode = "numeric", length = n_events)

  for(i in 1:n_events){
    # Poisson random variate
    n_occur[i] = rpois(1, event_prob[i] * tau)
  }
  return(n_occur)

}
```

Now, we can run a single realization of the stochastic process:

```
#######################################################
# SET-UP #
#######################################################

# Time
tau = 1 # leap size (in hours)
t_max = 100 # (in hours)
n_times = floor(t_max / tau)
t_vec = seq(1, t_max, length.out = n_times)

# Keep track of pop sizes for Virus and Immune (Z)
v_vec = vector(mode = "numeric", length = n_times)
z_vec = vector(mode = "numeric", length = n_times)

# INITIAL CONDITIONS
## How many viruses start the infection?
mean_dose = 50
```

```
## How many immune cells does the host start with?
mean_immune = 500

## At what dose is the host killed by the pathogen?
death_dose = 10^4

# Idicator variable: Did the host die?
death = 0
# If so, when did it die?
time_to_death = 0
# Indicator variable: Was the virus cleared?
cleared = 0

# Parameters:
phi = 0.39
K = 10^4.5
beta = 0.0011

params = list(
  phi = phi,
  K = K,
  beta = beta
)


#####################################################
# Run one realization of the system
#####################################################

# Start the clock:
t = 1

while(death == 0 & cleared ==0 & t <= (n_times-1)){

  if(t == 1){ # Initialize the system
    v_vec[1] = rpois(1, mean_dose)
    z_vec[1] = rpois(1, mean_immune)
  }

  # Move forward one time:
  n_occur = tau_leap_1step(t, tau, v_vec, z_vec, params)

  #-------------------------------------------------------
  #-------------------------------------------------------
  # Update your populations
  # Events:
  # 1. Replication of virus
  # 2. Depletion of virus = Depletion of immune system

  v_vec[(t+1)] = v_vec[t] + n_occur[1] - n_occur[2]
  z_vec[(t+1)] = z_vec[t] - n_occur[2]

  #-------------------------------------------------------
```

```
  #-----------------------------------------------------

  # Check if virus was cleared:
  if(v_vec[(t+1)] <= 0){
    cleared = 1
    # To make the log10 plot look nicer :)
    v_vec[(t+1)] = 1
  }

  # Correct if immune < 0
  if(z_vec[(t+1)] < 0) z_vec[(t+1)] = 0

  # Check to see if host died:
  if(v_vec[(t+1)] >= death_dose){
    death = 1
    time_to_death = (t+1)
    # Again, to make the plot look better :)
    v_vec[(t+1)] = death_dose
  }

  # Advance the time:
  t = t + 1

}


# Plot the dynamics of the virus for one realization:
plot(NA,NA,
     xlim = c(1, n_times),
     ylim = c(0, log10(death_dose+100)),
     xlab = "Time (hours)",
     ylab = "Log10(Virus Copies)")
lines(log10(v_vec) ~ t_vec)
abline(h = log10(death_dose), lty = 2)
```


Log10(Virus Copies)
0
1
2
3
4
0
20
40
60
80
100
Time (hours)

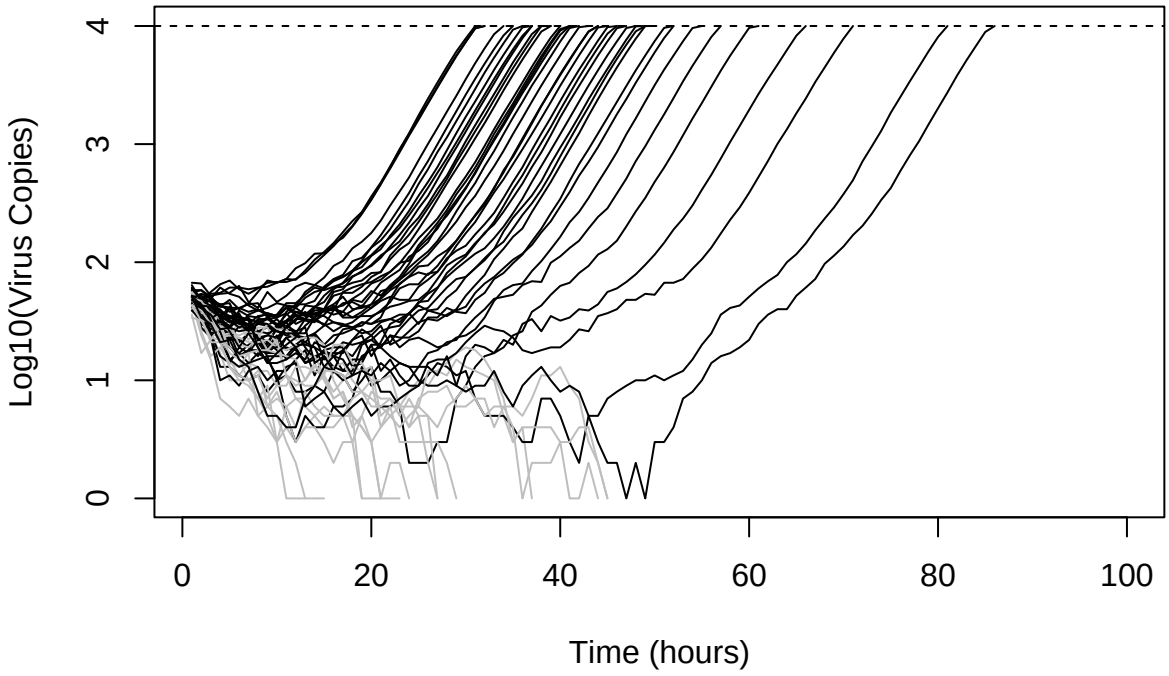
Task 2 (10 points)

Create a figure showing at least 50 realizations of the stochastic system. For each iteration, if the virus gets cleared, color the line gray, and if the host dies, color the line black.

```
n_runs = 50

plot(NA,NA,
     xlim = c(1, 100),
     ylim = c(0, log10(10^4+100)),
     xlab = "Time (hours)",
     ylab = "Log10(Virus Copies)")

abline(h = log10(10^4), lty = 2)


for (i in 1:50)
{
  ######################################################
  # SET-UP #
  ######################################################

  # Time
  tau = 1 # leap size (in hours)
  t_max = 100 # (in hours)
  n_times = floor(t_max / tau)
  t_vec = seq(1, t_max, length.out = n_times)

  # Keep track of pop sizes for Virus and Immune (Z)
  v_vec = vector(mode = "numeric", length = n_times)
  z_vec = vector(mode = "numeric", length = n_times)

  # INITIAL CONDITIONS
  ## How many viruses start the infection?
  mean_dose = 50

  ## How many immune cells does the host start with?
  mean_immune = 500

  ## At what dose is the host killed by the pathogen?
  death_dose = 10^4

  # Idicator variable: Did the host die?
  death = 0
  # If so, when did it die?
  time_to_death = 0
  # Indicator variable: Was the virus cleared?
  cleared = 0

  # Parameters:
  phi = 0.39
  K = 10^4.5
  beta = 0.0011
```

```
params = list(
  phi = phi,
  K = K,
  beta = beta
)


#####################################################
# Run one realization of the system
#####################################################

# Start the clock:
t = 1

while(death == 0 & cleared ==0 & t <= (n_times-1)){

  if(t == 1){ # Initialize the system
    v_vec[1] = rpois(1, mean_dose)
    z_vec[1] = rpois(1, mean_immune)
  }

  # Move forward one time:
  n_occur = tau_leap_1step(t, tau, v_vec, z_vec, params)

  #-------------------------------------------------------
  #-------------------------------------------------------
  # Update your populations
  # Events:
  # 1. Replication of virus
  # 2. Depletion of virus = Depletion of immune system

  v_vec[(t+1)] = v_vec[t] + n_occur[1] - n_occur[2]
  z_vec[(t+1)] = z_vec[t] - n_occur[2]

  #-------------------------------------------------------
  #-------------------------------------------------------

  # Check if virus was cleared:
  if(v_vec[(t+1)] <= 0){
    cleared = 1
    # To make the log10 plot look nicer :)
    v_vec[(t+1)] = 1
  }

  # Correct if immune < 0
  if(z_vec[(t+1)] < 0) z_vec[(t+1)] = 0

  # Check to see if host died:
  if(v_vec[(t+1)] >= death_dose){
    death = 1
    time_to_death = (t+1)
    # Again, to make the plot look better :)
    v_vec[(t+1)] = death_dose
```

```
    }

    # Advance the time:
    t = t + 1

  }

  if (death == 0)
  {
    lines(log10(v_vec) ~ t_vec, col='grey')
  } else {
    lines(log10(v_vec) ~ t_vec, col='black')
  }

}
```



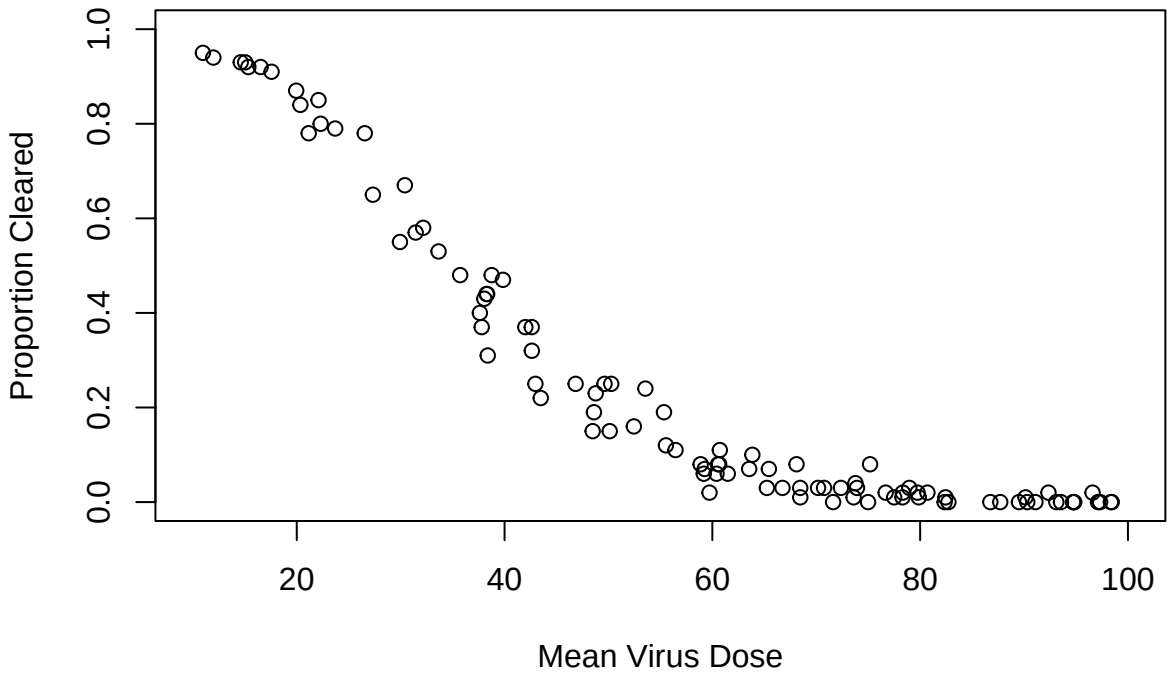
Task 3 (25 points)

Create a figure that demonstrates the effect of mean virus dose on the *proportion of iterations that result in virus clearance*. The virus dose is the amount of virus that inoculates the host at the initiation of the within-host process. The figure should have the "Proportion Cleared" on the y-axis and the "Mean Virus Dose" on the x-axis. Comment on the dynamics you see in the figure and why this pattern makes sense biologically. *HINT*: Each time you change the mean virus dose, you will need to run the model 50 times to calculate the proportion (i.e., $n/50$) of those iterations that result in a virus clearance. Then, move to the next value of mean virus dose, and repeat.

```
n_runs = 100
mean_virus_v = vector(mode='numeric')
proportion_cleared_v = vector(mode='double')

for (i in 1:n_runs)
{
  ## How many viruses start the infection?
  mean_dose = runif(1, min = 10, max = 100)
  mean_virus_v <- append(mean_virus_v, mean_dose)
  cleared_virus = 0

  for (i in 1:n_runs)
  {
    ######################################################
    # SET-UP #
    ######################################################

    # Time
    tau = 1 # leap size (in hours)
    t_max = 100 # (in hours)
    n_times = floor(t_max / tau)
    t_vec = seq(1, t_max, length.out = n_times)

    # Keep track of pop sizes for Virus and Immune (Z)
    v_vec = vector(mode = "numeric", length = n_times)
    z_vec = vector(mode = "numeric", length = n_times)

    # INITIAL CONDITIONS

    ## How many immune cells does the host start with?
    mean_immune = 500

    ## At what dose is the host killed by the pathogen?
    death_dose = 10^4

    # Idicator variable: Did the host die?
    death = 0
    # If so, when did it die?
    time_to_death = 0
    # Indicator variable: Was the virus cleared?
    cleared = 0

    # Parameters:
    phi = 0.39
    K = 10^4.5
    beta = 0.0011

    params = list(
      phi = phi,
      K = K,
      beta = beta
    )
```

```
#####################################################
# Run one realization of the system
#####################################################

# Start the clock:
t = 1

while(death == 0 & cleared ==0 & t <= (n_times-1)){

  if(t == 1){ # Initialize the system
    v_vec[1] = rpois(1, mean_dose)
    z_vec[1] = rpois(1, mean_immune)
  }

  # Move forward one time:
  n_occur = tau_leap_1step(t, tau, v_vec, z_vec, params)

  #-----------------------------------------------------
  #-----------------------------------------------------
  # Update your populations
  # Events:
  # 1. Replication of virus
  # 2. Depletion of virus = Depletion of immune system

  v_vec[(t+1)] = v_vec[t] + n_occur[1] - n_occur[2]
  z_vec[(t+1)] = z_vec[t] - n_occur[2]

  #-----------------------------------------------------
  #-----------------------------------------------------

  # Check if virus was cleared:
  if(v_vec[(t+1)] <= 0){
    cleared = 1
    # To make the log10 plot look nicer :)
    v_vec[(t+1)] = 1
  }

  # Correct if immune < 0
  if(z_vec[(t+1)] < 0) z_vec[(t+1)] = 0

  # Check to see if host died:
  if(v_vec[(t+1)] >= death_dose){
    death = 1
    time_to_death = (t+1)
    # Again, to make the plot look better :)
    v_vec[(t+1)] = death_dose
  }

  # Advance the time:
  t = t + 1

}
if (cleared == 1)
```

```
    {
      cleared_virus = cleared_virus + 1
    }
  }
  proportion_cleared = cleared_virus / n_runs
  proportion_cleared_v <- append(proportion_cleared_v, proportion_cleared)
}

plot(NA,NA,
     xlim = c(10, 100),
     ylim = c(0, 1),
     xlab = "Mean Virus Dose",
     ylab = "Proportion Cleared")

points(proportion_cleared_v ~ mean_virus_v)
```



This model indicates that the lower the initial amount of viral dose, the higher expected chance of viral clearance. As more of the virus is delivered, the chance of recovery decreases. Hence, viral load is directly proportional to chance of survival.

Task 4 (25 points)

Create a figure that demonstrates the effect of the virus growth rate, ϕ, on the *median time to death*. Comment on the dynamics you see in the figure and why this pattern makes sense biologically.

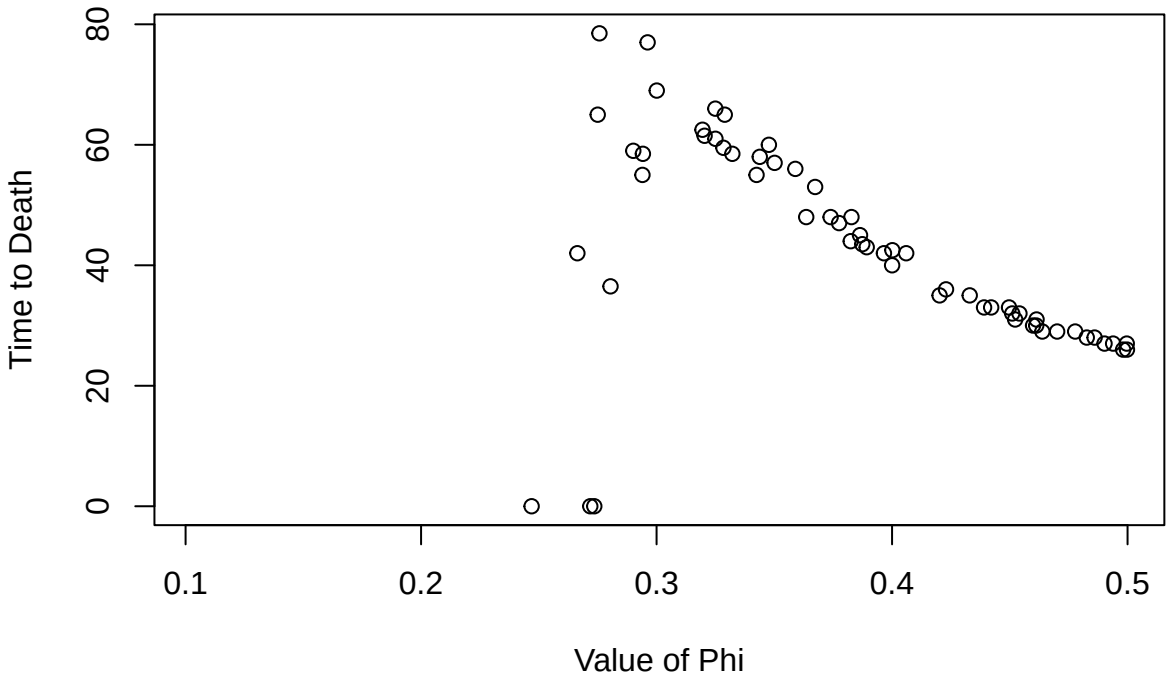
HINTS:

1. Again, for each value of ϕ you'll need to run the model 50 times to calculate a median.
2. You'll need to ignore iterations that lead to virus clearance, so that you are only using the iterations that led to virus-induced mortality. Create an array to store the time at which the host dies in each iteration; you could put an "NA'' value in this array if the host clears the infection. Then you can calculate the median of this array using the median() function in R (e.g., median(my_array, na.rm=TRUE))
3. The dynamics are quite sensitive to ϕ. This means you only need to vary ϕ over a small range, such as $0.1 - 0.5$.

```
n_runs = 100
median_virus_v = vector(mode='numeric')
phi_value_v = vector(mode='double')

for (i in 1:n_runs)
{
  phi = runif(1, 0.1, 0.5)
  temp_value_v = vector(mode='numeric')
  phi_value_v <- append(phi_value_v, phi)

  for (i in 1:n_runs)
  {
    ######################################################
    # SET-UP #
    ######################################################

    # Time
    tau = 1 # leap size (in hours)
    t_max = 100 # (in hours)
    n_times = floor(t_max / tau)
    t_vec = seq(1, t_max, length.out = n_times)

    # Keep track of pop sizes for Virus and Immune (Z)
    v_vec = vector(mode = "numeric", length = n_times)
    z_vec = vector(mode = "numeric", length = n_times)

    # INITIAL CONDITIONS
    ## How many viruses start the infection?
    mean_dose = 50

    ## How many immune cells does the host start with?
    mean_immune = 500

    ## At what dose is the host killed by the pathogen?
    death_dose = 10^4

    # Idicator variable: Did the host die?
    death = 0
    # If so, when did it die?
    time_to_death = 0
    # Indicator variable: Was the virus cleared?
    cleared = 0

    # Parameters:
    K = 10^4.5
```

```
beta = 0.0011

params = list(
  phi = phi,
  K = K,
  beta = beta
)


#####################################################
# Run one realization of the system
#####################################################

# Start the clock:
t = 1

while(death == 0 & cleared ==0 & t <= (n_times-1)){

  if(t == 1){ # Initialize the system
    v_vec[1] = rpois(1, mean_dose)
    z_vec[1] = rpois(1, mean_immune)
  }

  # Move forward one time:
  n_occur = tau_leap_1step(t, tau, v_vec, z_vec, params)

  #-----------------------------------------------------
  #-----------------------------------------------------
  # Update your populations
  # Events:
  # 1. Replication of virus
  # 2. Depletion of virus = Depletion of immune system

  v_vec[(t+1)] = v_vec[t] + n_occur[1] - n_occur[2]
  z_vec[(t+1)] = z_vec[t] - n_occur[2]

  #-----------------------------------------------------
  #-----------------------------------------------------

  # Check if virus was cleared:
  if(v_vec[(t+1)] <= 0){
    cleared = 1
    # To make the log10 plot look nicer :)
    v_vec[(t+1)] = 1
  }

  # Correct if immune < 0
  if(z_vec[(t+1)] < 0) z_vec[(t+1)] = 0

  # Check to see if host died:
  if(v_vec[(t+1)] >= death_dose){
    death = 1
    time_to_death = (t+1)
```

```
        # Again, to make the plot look better :)
        v_vec[(t+1)] = death_dose
      }

      # Advance the time:
      t = t + 1

    }
    if (cleared == 0)
    {
      temp_value_v <- append(temp_value_v, time_to_death)
    } else {
      temp_value_v <- append(temp_value_v, NA)
    }
  }
  median_time_to_death = median(temp_value_v, na.rm=TRUE)
  median_virus_v <- append(median_virus_v, median_time_to_death)
}

plot(NA,NA,
     xlim = c(min(phi_value_v), max(phi_value_v)),
     ylim = c(0, max(median_virus_v, na.rm=TRUE)),
     xlab = "Value of Phi",
     ylab = "Time to Death")

points(median_virus_v ~ phi_value_v)
```



This model shows that after a certain value of Phi, the virus will lead to death. However, viruses with a lower rate of replication will not lead to death, indicating that they will be cleared from the host. As the value of Phi increases, the time to death also decreases, which indicates that the faster a virus replicates, the faster its host will die. This is in line with the model observed in Task 3, where we saw that a higher amount of viral load will lead to a higher chance of death.