Diffusion Model

Annabelle

2024-08-15

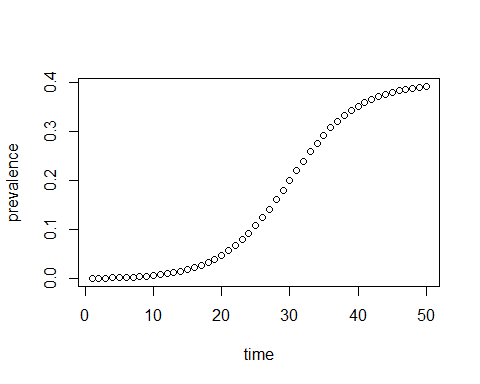
### Set up the simulation

This is a mock landscape (the size of the hex cells and landscape size aren’t important). We start the simulation by randomly starting two of the cells with .3 prevelence CWD

### Growing the disease

The first step in the simulation is to grow the disease. We use a logistic growth model with a carrying capacity (K) of .4. *I need help setting the parameters here*

newfunc <- function(r = .1, K = 200, N0 = 5, time = 125){  
 t <- seq(1,time)  
 newN <- c()  
 for(i in t){  
 C = (K\*N0)/(K-N0)  
 N = (C\*exp(1)^(r\*i))/ (1 + ((C/(K)\*exp(1)^(r\*i))))  
 newN <- c(newN, N)  
 }  
 return(newN)  
}  
prevalence <- newfunc(r = .2, K = .40, N0 = .001, time = 50)  
time = 1:50  
plot(time , prevalence)



This is set up as follows in the code: 1) for a given prevalence value, we need to know what time set that prevalence is found at 2) we also want to be able to specify a time and get the prevalence value at that time step 3) these are put together so each year we can step the prevalence one time step forward

# given prevalence, return t  
returnt <- function(N, r = .2, K = .40, N0 = .001){  
 t <- -(1/r)\*(log( (N0\*(K-N)) / (N\*(K-N0)) ))  
 return(t)  
}  
# given time, return prevalence  
returnp <- function(t1, r = .2, K = .40, N0 = .001){  
 pt <- K/(1 + ( ((K-N0)/N0)\*(exp(1)^(-r\*t1)) ) )  
 return(pt)  
}  
  
# combines returnp & returnt functions..   
# Need to make sure this only runs if prev is non 0!!  
next\_t\_prev\_func<- function(fakeprev){ #take starting prevalence  
 t0 <- returnt(N = fakeprev) # id what time step it's on  
 t\_1 <- t0 + 1 # add one to get to next time step  
 updateprev <- returnp(t\_1) # get the updated prevalence associated with that  
 return(updateprev)  
}  
  
### Was running into an issue with NAs being generated when grew too quickly so this fixes that..  
next\_t\_prev\_FIXED\_func<- function(fakeprev, r = .2, K = .40, N0 = .001){ #take starting prevalence  
 N = fakeprev  
 t0 <- -(1/r)\*(log( (N0\*(K-N)) / (N\*(K-N0)) )) # id what time step it's on  
   
 t0 <- returnt(N = fakeprev) # id what time step it's on  
 # If go past 40, returns NAs so need to counter that.. but time steps are totally fucked...   
   
 t\_1 <- t0 + 1 # add one to get to next time step  
 updateprev <- returnp(t\_1) # get the updated prevalence associated with that  
   
 t[which(is.na(t))] <- .4 # fix NAs..   
 return(updateprev)  
}

### Run the growth model

#need to filter and only run on non 0 cells  
## Will have to set time step up later !!!!!!!  
## !!! This also doesn't save previous prevalences...   
for(i in 1:nrow(hex\_df)){  
 if(hex\_df[i,2] != 0){  
 hex\_df[i,2] <- next\_t\_prev\_func(hex\_df[i,2])  
 }  
}  
# hex\_sf$prev <- hex\_df$prev # don't need to update this here..

### Model underpinnings

We can identify cells that are adjacent to one another and record the diffusion using the following code (this is also used in the disease spread functions)

## Need to get adjacency matrix...  
mymat <- st\_intersects(hex\_sf, hex\_sf)  
  
# Set up matrix to record diffusion   
mat\_prev <- matrix(NA, nrow(hex\_df), nrow(hex\_df))  
# set mat prev as diagonal  
diag(mat\_prev) <- hex\_df$prev

### Spreading the disease

We use the following equation to model how the disease spreads: (Astor & Adami 1998) Where: \ “ is the concentration of substrate in site , is a diffusion coefficient(< 0.5 to avoid substrate oscillation), and represents the th neighbor of grid element .” We also identify any negative diffusion that results from the calculation and set that to 0 *We need to set the diffusion constant for CWD* *The following also assumes a hard boundary, to adjust for this, we can create multiple versions of the same function, but have an indicator in the cells to dictate which function is used (i.e. if it’s not a try boundary use function A with 6 neighbors diffusing, but if it is a true boundary, use function B..)*

## amount of disease left in cell..  
post\_diffusion\_func <- function(cellno, mat\_prev. = mat\_prev, mymat. = mymat, D = .2){ # setting diffusion constant..  
 hold <- list()  
 adj <- (unlist(mymat.[cellno]))   
 #remove the duplicate cell (if i = 1, remove "1")  
 adj <- adj[adj != cellno]  
 for(j in adj){  
 ## Need to pull concentration out of this..   
 hold[j] <- mat\_prev.[cellno,cellno] - mat\_prev.[j,j]  
 }  
 outsum <- sum(unlist(hold))   
 # Can't have negative diffusion  
 if(outsum < 0){  
 outsum <- 0  
 }  
 return(outsum \* (D/6))  
}  
  
  
## amount of disease to allocate to neighbooring cells..  
amount\_to\_allocate\_func <- function(cellno, mat\_prev. = mat\_prev, mymat. = mymat, D = .2){  
 adj <- (unlist(mymat.[cellno]))   
 diff <- mat\_prev.[cellno,cellno] - post\_diffusion\_func(cellno, mat\_prev, mymat, D = .2)  
 return(diff/length(adj)) ## currently a hard boundary  
}  
  
### Currently being buggy..   
amount\_to\_allocate\_func2 <- function(cellno, mat\_prev. = mat\_prev, mymat. = mymat, D = .2){  
 adj <- (unlist(mymat.[cellno]))   
 #setting diffusion constant..  
 hold <- list()  
 adj <- (unlist(mymat.[cellno]))   
 #remove the duplicate cell (if i = 1, remove "1")  
 adj <- adj[adj != cellno]  
 for(j in adj){  
 ## Need to pull concentration out of this..   
 hold[j] <- mat\_prev.[cellno,cellno] - mat\_prev.[j,j]  
 }  
 outsum <- sum(unlist(hold))   
 # Can't have negative diffusion  
 if(outsum < 0){  
 outsum <- 0  
 }  
 post\_diffusion\_amount <- (outsum \* (D/6))  
 diff <- mat\_prev.[cellno,cellno] - post\_diffusion\_amount  
 return(diff/length(adj)) ## currently a hard boundary  
}

### Identifying Neighboors

## Need to get adjacency matrix...  
mymat <- st\_intersects(hex\_sf, hex\_sf)  
  
# Set up matrix to record diffusion   
mat\_prev <- matrix(NA, nrow(hex\_df), nrow(hex\_df))  
# set mat prev as diagonal  
diag(mat\_prev) <- hex\_df$prev

### Run the diffusion model

### ### Now need to iterate through matrix, allocate new prevalence values   
mat\_prev2 <- mat\_prev ## Need to fix this to make temporal...   
for(i in 1:nrow(mat\_prev)){  
 endprev <- post\_diffusion\_func(i, mat\_prev, mymat, D = .2)  
 mat\_prev2[i,i] <- endprev # overwrite prevalence   
 adj <- (unlist(mymat[i]))   
 #remove the duplicate cell (if i = 1, remove "1")  
 adj <- adj[adj != i]  
 share\_prev <- amount\_to\_allocate\_func(i, mat\_prev, mymat, D = .2)  
 for(n in adj){  
 mat\_prev2[i,n] <- share\_prev # write in added prevalence  
 }  
}  
  
### then sum columns to get new prevalence   
newprevvales <- colSums(mat\_prev2, na.rm = TRUE)  
  
hex\_sf$prev <- newprevvales

### Updated viusal

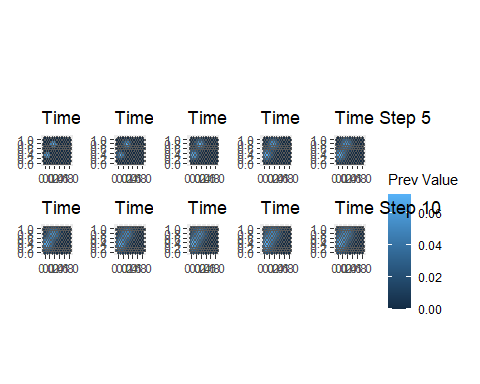
### Adding a time step

**Question - do we want to save/record previous prevalence in certain cells?**

### ### ### Set up  
noyears <- 50 # no years   
  
### Create grid   
# Convert the grid to an sf object  
hex\_sf <- st\_sf(geometry = hex)  
  
# Add a 'value' column with example values  
hex\_sf$value <- seq\_along(hex\_sf$geometry) # Ensure this column is numeric and non-NA  
  
# Check for NA values  
if (any(is.na(hex\_sf$value))) {  
 stop("NA values detected in the value column.")  
}  
  
# specify value of cells based on seq   
hex\_df <- as.data.frame(hex\_sf$value)  
hex\_df$prev <- c(0)  
hex\_df[c(22,57),2] <- .3  
  
hex\_sf$prev <- hex\_df$prev  
  
### Replicate over as many time steps as needed  
og\_landscape\_list <- replicate(noyears, hex\_sf, simplify = FALSE)  
  
## Need to get adjacency matrix...  
mymat\_adj <- st\_intersects(hex\_sf, hex\_sf)  
  
# Set up matrix to record diffusion   
mat\_prev <- matrix(NA, nrow(hex\_df), nrow(hex\_df))  
  
mat\_prev\_list <- replicate((noyears + 1), mat\_prev, simplify = FALSE)  
  
### ### ### Run over time loop   
  
for(t in 1:29){  
   
# Grow the disease   
for(i in 1:nrow(hex\_df)){  
 if(hex\_df[i,2] != 0){  
 hex\_df[i,2] <- next\_t\_prev\_FIXED\_func(hex\_df[i,2])  
 }  
}  
  
# set mat prev as diagonal  
diag(mat\_prev\_list[[t]]) <- hex\_df$prev   
  
for(i in 1:nrow(mat\_prev\_list[[t]])){  
 endprev <- post\_diffusion\_func(i, mat\_prev\_list[[t]], mymat\_adj, D = .2)  
 mat\_prev\_list[[t+1]][i,i] <- endprev # overwrite prevalence   
 adj <- (unlist(mymat\_adj[i]))   
 #remove the duplicate cell (if i = 1, remove "1")  
 adj <- adj[adj != i]  
 share\_prev <- amount\_to\_allocate\_func2(i, mat\_prev\_list[[t]], mymat\_adj, D = .2)  
 for(n in adj){  
 mat\_prev\_list[[t+1]][i,n] <- share\_prev # write in added prevalence  
 }  
}  
  
### then sum columns to get new prevalence   
newprevvales <- colSums(mat\_prev\_list[[t+1]], na.rm = TRUE)  
  
og\_landscape\_list[[t]]$prev <- hex\_df$prev <- newprevvales  
  
}

Visualize first 10 time steps

# List to store individual plots  
plot\_list <- vector("list", 10)  
  
# Create a plot for each spatial object and store in plot\_list  
for (i in 1:10) {  
 plot\_list[[i]] <- ggplot(data = og\_landscape\_list[[i]]) +  
 geom\_sf(aes(fill = prev), show.legend = i == 10) + # Only show legend on the first plot  
 labs(title = paste("Time Step", i), fill = "Prev Value")  
}  
  
# Combine all plots using patchwork  
combined\_plot <- wrap\_plots(plot\_list, ncol = 5) # Adjust ncol as needed  
  
# Display the combined plot  
print(combined\_plot)



### Repeat the above but set up a soft boundary (ie continue diffusion past edge)

### Now set a soft boundary on 3 sides and a hard boundary on 1 side

### Apply to Northeast landscape