Simulating Sudden Oak Death Dynamics

November 16, 2012

I AM WORKING ON A PROJECT WITH the Rizzo Lab examining the dynamics of Sudden Oak Death (SOD). I really have to write more about this, but today I'm just going to post the results of an initial exercise.

Here I attempt to replicate model results from Cobb et al. (2012). The model in that paper simulates the spread of disease and resulting tree mortality and stand dynamics in a mixed system of tanoak, bay laurel, and redwood. In this system, only tanoak and bay laurel carry the disease, and it mostly only kills tanoak, but all three species compete for space in the forest. More detail on the model found in the paper and the paper's supplement.

I'm developing this model as an R package - `"SODDr"` (Sudden Oak Death Dynamics in R), to replicate this work, you can install it from github. Note that this is a very rough package and is changing.

```
library(devtools) # devtools enables intsallation fr
install_github("SODDr","noamross")


library(SODDr)
```

My model modifies the original in a few ways. First, it's designed to be much more flexible, and make it easy to modify the number of species, size classes, and and disease parameters. Secondly, it is in a discrete- rather than continuous-time framework. This will make it easier to include stochasticity down the road, and ultimately fit it to actual data.

SETUP

First, I load a CSV file of parameters

```
treeparms.df <- read.csv(system.file("paper_tree_par
    stringsAsFactors = FALSE)
print(treeparms.df)
```

```
##   species ageclass S.mortality I.mortality S.recr
## 1       1        1      0.0061       0.019
## 2       1        2      0.0031       0.022     0.
## 3       1        3      0.0011       0.035     0.
## 4       1        4      0.0320       0.140     0.
## 5       2        1      0.0200       0.020
## 6       3        1      0.0050       0.020
##   S.transition I.transition S.resprout I.resprout
## 1      0.05212      0.05212          0        0.5
## 2      0.01848      0.01848          0        0.5
## 3      0.01473      0.01473          0        0.5
## 4      0.00000      0.00000          0        0.5
## 5      0.00000      0.00000          0        0.0
## 6      0.00000      0.00000          0        0.0
##            kernel.fn kernel.par1 kernel.par2  S.i
## 1 adjacent.dispersal         0.5       0.125 0.03
## 2 adjacent.dispersal         0.5       0.125 0.09
## 3 adjacent.dispersal         0.5       0.125 0.11
## 4 adjacent.dispersal         0.5       0.125 0.04
## 5 adjacent.dispersal         0.5       0.125 0.11
## 6 adjacent.dispersal         0.5       0.125 0.32
##   waifw3 waifw4 waifw5 waifw6
## 1   0.33   0.33   1.46      0
## 2   0.32   0.32   1.46      0
## 3   0.30   0.30   1.46      0
## 4   0.24   0.24   1.46      0
## 5   0.30   0.30   1.33      0
## 6   0.00   0.00   0.00      0
```

This data set includes all of the model parameters as defined by Cobb et al. (2012). For each species and age class, there are parameters for mortality, recruitment, transition between classes, the probability of resprouting, and recovering from disease. There are also parameters the the relative amount of space occupied by trees of each species/class.

For each species/class, the table includes a kernel function to describe dispersal of disease spores from diseased trees, and parameters for that function. The last set of columns represent the "“Who Acquires Infection From

Whom" (WAIFW) matrix (Anderson and May 1985), which describes the vulnerability of each class to spores originating from others.

Note that several values, such as the space occupied by different tanoak size classes, and recruitment rates, are missing. This is because, per the original paper, I calculate these so as to parameterize the model for steady-state conditions in the absence of disease. Here I do that, using Equation 8 in the papers' supplement:

```
treeparms.df <- within(treeparms.df, {
    # Calculate the relative space requirements of t
    # on initial conditions
    space[1:4] <- 0.25 * (sum(S.init[1:4])/S.init[1:

    # Set recruitment rates to steady-state levels.
    # is simply the mortality rate divided by the de
    # coefficient at simulation start.  For tanoak,
    # classes, it's a but more involved

    S.recruit[5] <- S.mortality[5]/(1 - sum(S.init *
    I.recruit[5] <- I.mortality[5]/(1 - sum(S.init *
    S.recruit[6] <- S.mortality[6]/(1 - sum(S.init *
    I.recruit[6] <- I.mortality[6]/(1 - sum(S.init *

    A2 <- S.transition[1]/(S.transition[2] + S.morta
    A3 <- (S.transition[2]/(S.transition[3] + S.mort
    A4 <- (S.transition[3]/S.mortality[4]) * A3
    S.recruit[1] <- (S.transition[1] + S.mortality[1
        (S.recruit[2] * A2 + S.recruit[3] * A3 + S.r


    I.recruit[1] <- S.recruit[1]
    A2 <- NULL
    A3 <- NULL
    A4 <- NULL
})
```

The original model allows only dispersal between adjacent cells, so the parameters table calls the dispersal kernel `adjacent.dispersal`, and gives two parameters. This function outputs the first parameter when within the cell, the second for dispersal to adjacent cells, and zero for other cells:

```
print(adjacent.dispersal)
```

```
## function (distance, local, adjacent)
## {
##     ifelse(distance < 0.5, local, ifelse(distance
##         0))
## }
## <environment: namespace:SODDr>
```

Next I set up the locations in the model. These must be in the form of a matrix with the first column being site numbers, and the second being `x` and `y` coordinates. `MakeLattice` is a convenience function that creates a regularly spaced grid:

```
locations <- MakeLattice(nx = 20, ny = 20, dist = 1)
head(locations)
```

```
##      location x y
## [1,]        1 0 0
## [2,]        2 0 1
## [3,]        3 0 2
## [4,]        4 0 3
## [5,]        5 0 4
## [6,]        6 0 5
```

Next, I create a matrix of initial population values, which should be sized by number of locations by number of species/size classes *times 2*. They should be in order of the classes as they appear in the data table, but alternating $S, I, S, I, \ldots$. In this case, initial populations are uniform across the landscape and equal to the `init` values from the parameters table.

```
initial.vec = as.vector(rbind(treeparms.df$S.init, t
init <- matrix(data = initial.vec, nrow = nrow(locat
    byrow = TRUE)
```

Finally, we set the time steps for the model

```
time.steps <- 1:100
```

DO IT!

Running `SODModel` runs the model and generates a dataframe of population by time, species, size class, and

disease status

```
pop.df <- SODModel(treeparms.df, locations, time.ste
str(pop.df)
```

```
## 'data.frame':    480000 obs. of  6 variables:
##  $ Time      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Location  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Species   : Factor w/ 3 levels "1","2","3": 1
##  $ AgeClass  : Factor w/ 4 levels "1","2","3","4"
##  $ Disease   : Factor w/ 2 levels "S","I": 1 2 1
##  $ Population: num  0.0386 0 0.0978 0 0.1138 ...
```

RESULTS

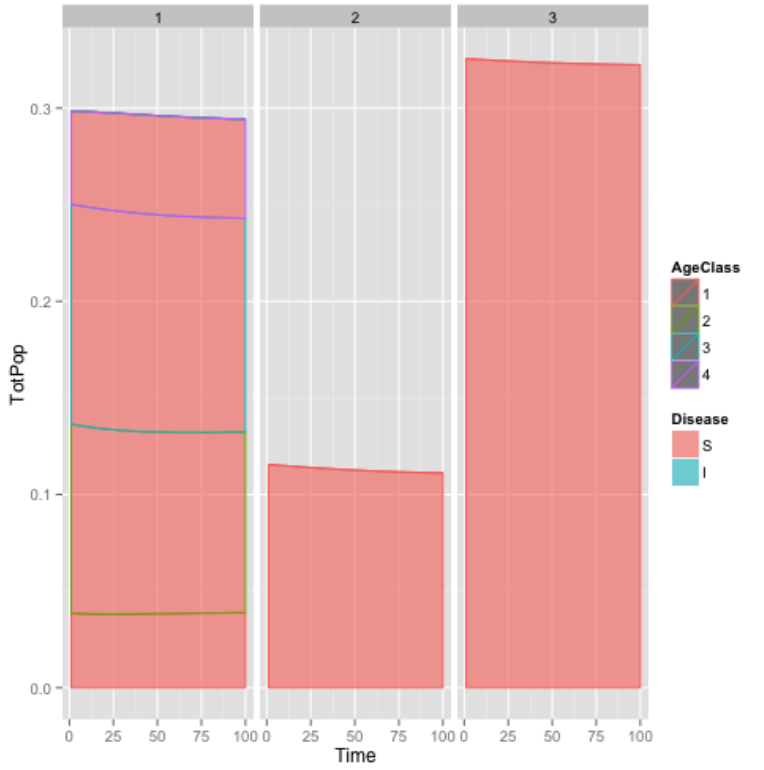
Here are the results, broken up by species and age class:

```
library(ggplot2)
library(plyr)
```

```
## Attaching package: 'plyr'
```

```
## The following object(s) are masked from 'package:
##
## id
```

```
pop.df.totals <- ddply(pop.df, c("Time", "Disease",
    summarise, TotPop = mean(Population))
dynamic.plot <- ggplot(pop.df.totals, aes(x = Time,
    color = AgeClass)) + geom_area(position = "stack
dynamic.plot
```

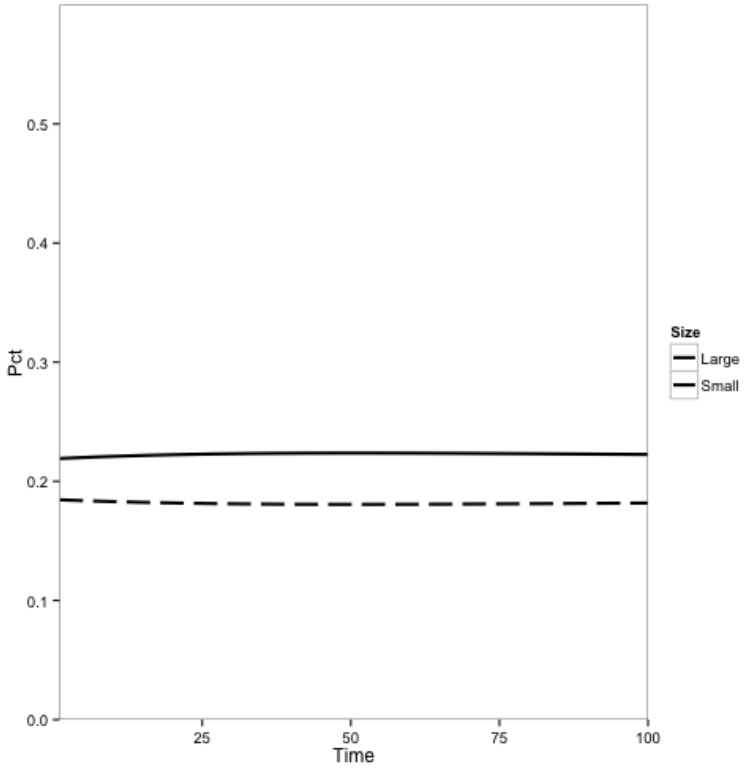


Population dynamics broken up by species (left to right, Tanoak, Bay, Redwood), and Size Class

The plot shows that this isn't quite steady-state, I suspect because of some rounding errors in copying parameters from the paper.

We can make a plot in the style of Cobb et al. (2012), showing only the populations of small and large tanoaks as a proportion of the total population:

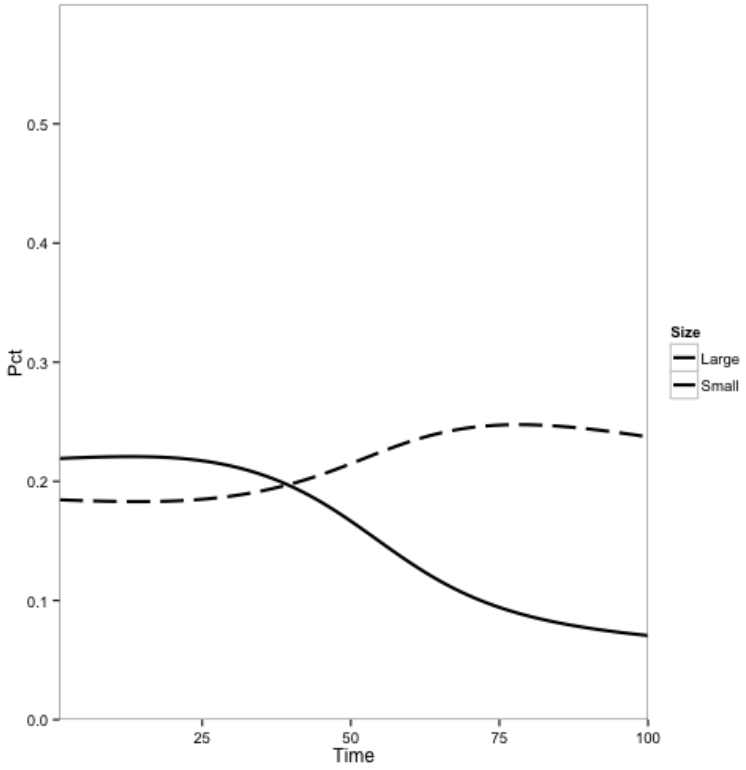
```
paper.df <- ddply(transform(pop.df.totals, Size = if
    "2", "Small", "Large")), c("Time", "Species", "S
paper.df <- ddply(paper.df, "Time", transform, Pct =
paper.plot <- ggplot(subset(paper.df, Species == 1),
    lty = Size)) + geom_line(lwd = 1) + scale_y_cont
    expand = c(0, 0)) + scale_x_continuous(expand =
    5)) + theme_bw() + theme(panel.grid.major = elem
paper.plot
```



Tanoak as a fraction of total population over time, no disease

Now, let's change the initial conditions to include disease and see what happens. I change the populations of tanoak and bay in one pixel to infectious instead of susceptible and run the model:

```
init[190, 2] <- init[190, 1]
init[190, 1] <- 0
init[190, 10] <- init[190, 9]
init[190, 9] <- 0
pop2.df <- SODModel(treeparms.df, locations, time.st
pop2.df.totals <- ddply(pop2.df, c("Time", "Disease"
    summarise, TotPop = mean(Population))
paper2.df <- ddply(transform(pop2.df.totals, Size =
    AgeClass == "2", "Small", "Large")), c("Time", "
    SizePop = sum(TotPop))
paper2.df <- ddply(paper2.df, "Time", transform, Pct
paper2.plot <- ggplot(subset(paper2.df, Species == 1
    lty = Size)) + geom_line(lwd = 1) + scale_y_cont
    expand = c(0, 0)) + scale_x_continuous(expand =
    5)) + theme_bw() + theme(panel.grid.major = elem
paper2.plot
```



Tanoak as a fraction of total population, using baseline initial conditions and introducing disease

Qualitatively, this looks a lot like the original result:

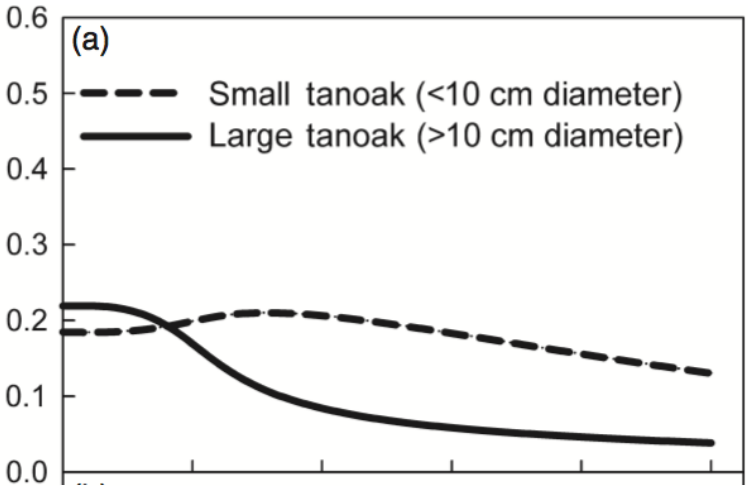


Figure 4a from Cobb et al. (2012)

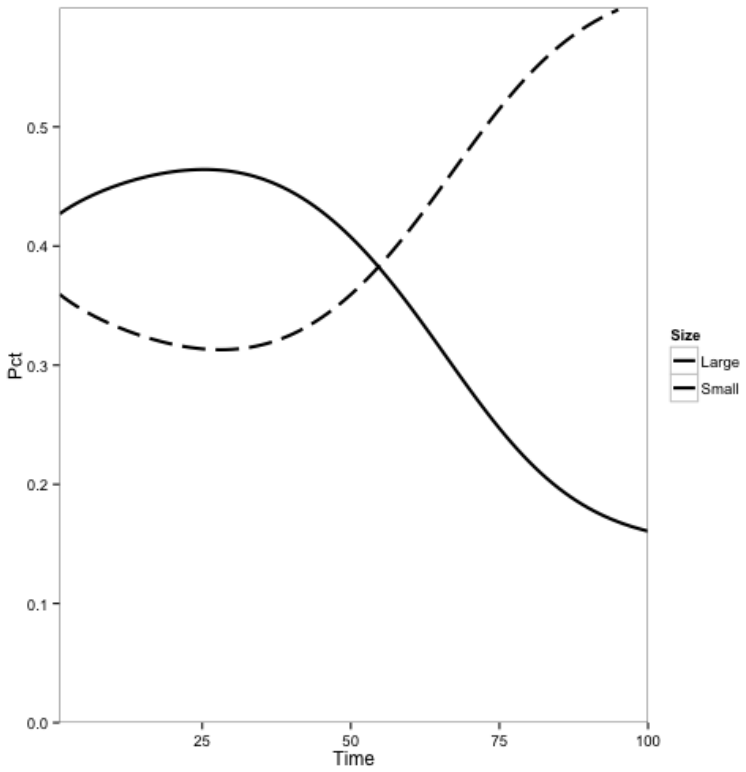
The main difference is the overall rate of change, which is expected because I haven't corrected any parameters for the change from continuous to discrete time.

Now I simulate a scenario with mostly tanoak, some redwood, and no bay laurel, with disease, using the

parameters previously calulated for the steady state without disease:

```
treeparms.df$S.init <- c(0.7 * treeparms.df$S.init[1
    0, 0.19)

initial.vec = as.vector(rbind(treeparms.df$S.init, t
init <- matrix(data = initial.vec, nrow = nrow(locat
    byrow = TRUE)

init[190, 2] <- init[190, 1]
init[190, 1] <- 0
init[190, 10] <- init[190, 9]
init[190, 9] <- 0

treeparms.df <- within(treeparms.df, {
    # Calculate the relative space requirements of t
    # on initial conditions
    space[1:4] <- 0.25 * (sum(S.init[1:4])/S.init[1:

    # Set recruitment rates to steady-state levels.
    # is simply the mortality rate divided by the de
    # coefficient at simulation start.  For tanoak,
    # classes, it's a but more involved

    S.recruit[5] <- S.mortality[5]/(1 - sum(S.init *
    I.recruit[5] <- I.mortality[5]/(1 - sum(S.init *
    S.recruit[6] <- S.mortality[6]/(1 - sum(S.init *
    I.recruit[6] <- I.mortality[6]/(1 - sum(S.init *

    A2 <- S.transition[1]/(S.transition[2] + S.morta
    A3 <- (S.transition[2]/(S.transition[3] + S.mort
    A4 <- (S.transition[3]/S.mortality[4]) * A3
    S.recruit[1] <- (S.transition[1] + S.mortality[1
        (S.recruit[2] * A2 + S.recruit[3] * A3 + S.r


    I.recruit[1] <- S.recruit[1]
    A2 <- NULL
    A3 <- NULL
    A4 <- NULL
})

pop3.df <- SODModel(treeparms.df, locations, time.st
pop3.df.totals <- ddply(pop3.df, c("Time", "Disease"
    summarise, TotPop = mean(Population))
paper3.df <- ddply(transform(pop3.df.totals, Size =
    AgeClass == "2", "Small", "Large")), c("Time", "
    SizePop = sum(TotPop))
paper3.df <- ddply(paper3.df, "Time", transform, Pct
paper3.plot <- ggplot(subset(paper3.df, Species == 1
    lty = Size)) + geom_line(lwd = 1) + scale_y_cont
    expand = c(0, 0)) + scale_x_continuous(expand =
    5)) + theme_bw() + theme(panel.grid.major = elem
paper3.plot
```

```
## Warning: Removed 5 rows containing missing values
```



Tanoak dynamics under the "Mostly Tanoak" scenario

Again, qualitatively similar to the original results:

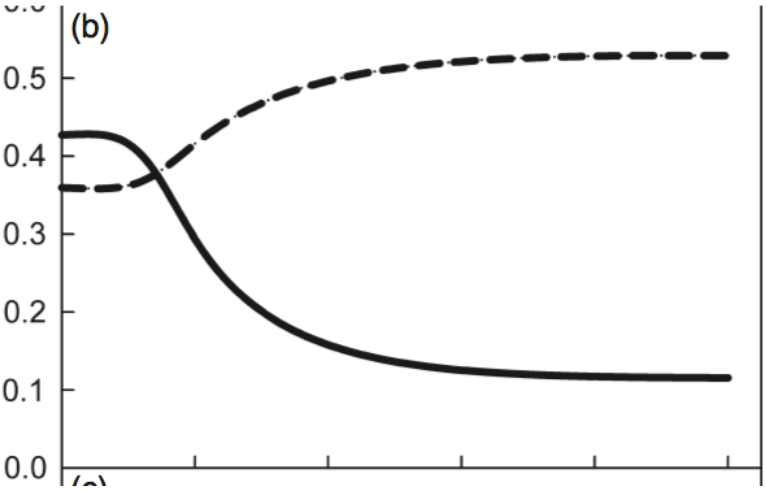
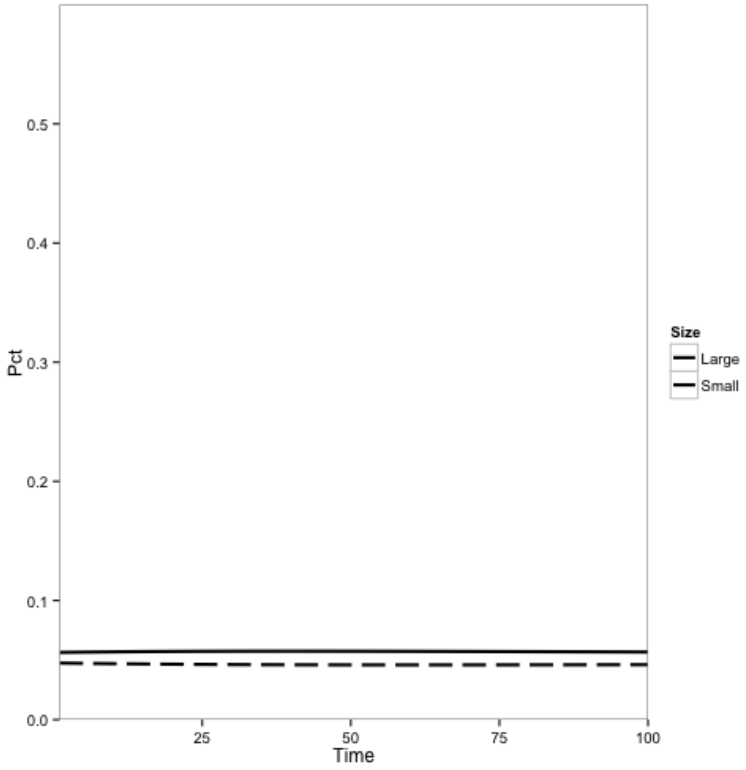


Fig 4b from Cobb et al. (2012)

Finally, I simulate the "Mostly Redwood" scenario:

```
treeparms.df$S.init <- c(0.08 * treeparms.df$S.init[
    0, 0.69)

initial.vec = as.vector(rbind(treeparms.df$S.init, t
init <- matrix(data = initial.vec, nrow = nrow(locat
    byrow = TRUE)
```

```
init[190, 2] <- init[190, 1]
init[190, 1] <- 0
init[190, 10] <- init[190, 9]
init[190, 9] <- 0

treeparms.df <- within(treeparms.df, {
    # Calculate the relative space requirements of t
    # on initial conditions
    space[1:4] <- 0.25 * (sum(S.init[1:4])/S.init[1:

    # Set recruitment rates to steady-state levels.
    # is simply the mortality rate divided by the de
    # coefficient at simulation start.  For tanoak,
    # classes, it's a but more involved

    S.recruit[5] <- S.mortality[5]/(1 - sum(S.init *
    I.recruit[5] <- I.mortality[5]/(1 - sum(S.init *
    S.recruit[6] <- S.mortality[6]/(1 - sum(S.init *
    I.recruit[6] <- I.mortality[6]/(1 - sum(S.init *

    A2 <- S.transition[1]/(S.transition[2] + S.morta
    A3 <- (S.transition[2]/(S.transition[3] + S.mort
    A4 <- (S.transition[3]/S.mortality[4]) * A3
    S.recruit[1] <- (S.transition[1] + S.mortality[1
        (S.recruit[2] * A2 + S.recruit[3] * A3 + S.r


    I.recruit[1] <- S.recruit[1]
    A2 <- NULL
    A3 <- NULL
    A4 <- NULL
})

pop4.df <- SODModel(treeparms.df, locations, time.st
pop4.df.totals <- ddply(pop4.df, c("Time", "Disease"
    summarise, TotPop = mean(Population))
paper4.df <- ddply(transform(pop4.df.totals, Size =
    AgeClass == "2", "Small", "Large")), c("Time", "
    SizePop = sum(TotPop))
paper4.df <- ddply(paper4.df, "Time", transform, Pct
paper4.plot <- ggplot(subset(paper4.df, Species == 1
    lty = Size)) + geom_line(lwd = 1) + scale_y_cont
    expand = c(0, 0)) + scale_x_continuous(expand =
    5)) + theme_bw() + theme(panel.grid.major = elem
paper4.plot
```



Tanoak dynamics under the "mostly redwood" scenario

Success! Same results as the original:

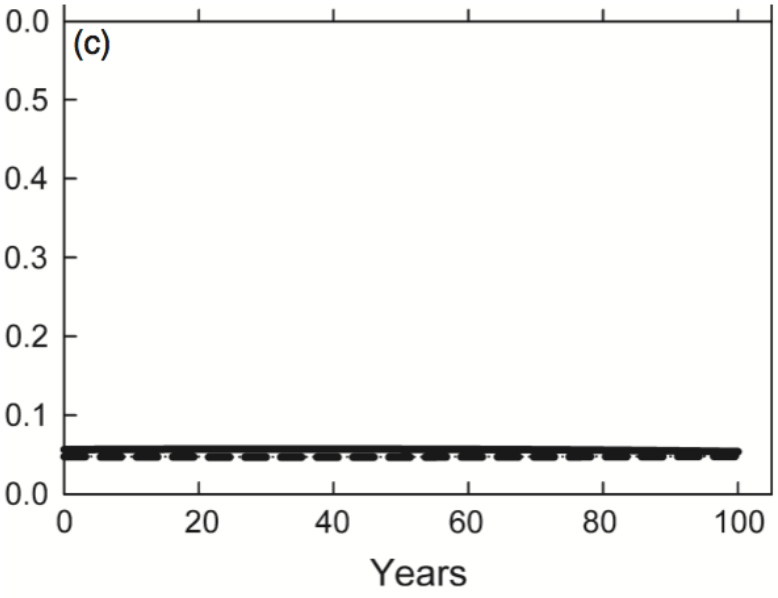


Figure 4c from Cobb et al. (2012)

Now, for fun, let's make an animation of how the disease progresses through the stand. I will use the "Mostly Tanoak" scenario", and plot the adult population as intensity, and the fraction of trees diseased as hue.

```
tanspc <- subset(pop3.df, Species == 1)
tanspc$Species <- NULL
tanspc <- ddply(tanspc, c("Time", "Location"), funct
    data.frame(TotTan = sum(x$Population[which(x$Age
        "4")]), FracDisease = sum(x$Population[which
        "3" | x$AgeClass == "4"))])/sum(x$Population
        x$AgeClass == "4")]))
})
tanspc <- merge(tanspc, as.data.frame(locations), by
dis.limits <- c(min(tanspc$FracDisease), max(tanspc$
pop.limits <- c(min(tanspc$TotTan), max(tanspc$TotTa
require(animation)
```

```
## Loading required package: animation
```

```
ani.options(nmax = 50, loop = TRUE, interval = 0.2)
for (time in time.steps) {
    print(ggplot(data = subset(tanspc, Time == time)
        alpha = TotTan)) + geom_tile() + scale_alpha
        limits = c(0, 0.4), breaks = seq(0, 0.4, 0.1
        limits = c(0, 1), breaks = seq(0, 1, 0.25),
        theme(rect = element_blank(), line = element
}
```

0:00 / 0:20

← Jordi Bascompte on Mutualistic Networks | All posts | A quick function for editing CSV files in R →