TBL 5 Exercise: Simulating Population Growth

DUE: March 15, 2017

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1. Meta-Analysis Update (+10)

Please update on the progress on your meta-analysis papers.

2. Constant Population Growth (+10)

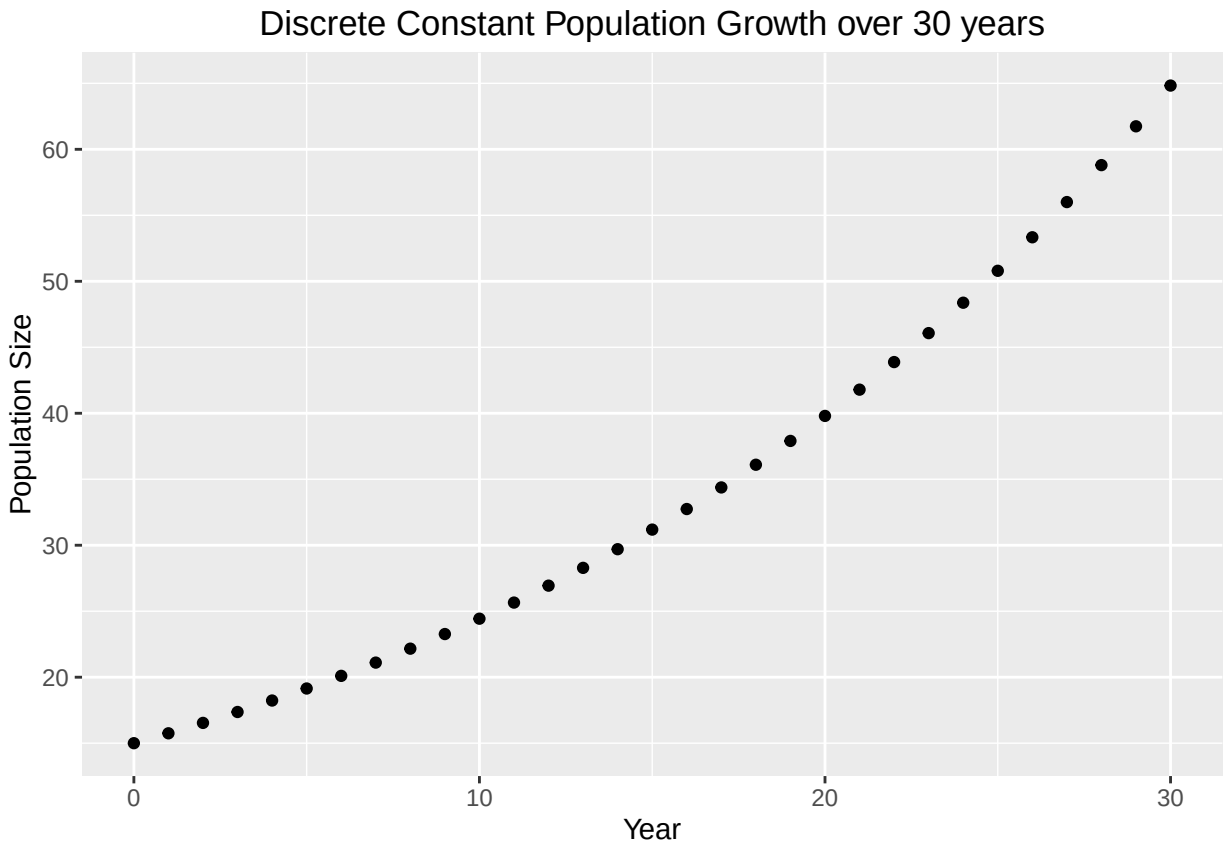
Construct a figure similar to page 10 on this week's lecture notes using the pseudocode on pages 7 and 9. However, instead of showing the results continuously, show as discrete estimates of the population size (i.e. lines not connected). Also, do a better job than your professor and show a legend on the figure (bad professor!).

```
time.zero<-0
end.time<-30
pop.initial<-15
lambda<-1.05
popsize.year<-numeric()
popsize.time<-numeric()
for(i in time.zero:end.time){
  popsize.year[i+1]<-pop.initial*lambda^i
  for(j in time.zero:end.time){
    popsize.time[j+1]<-time.zero+j
  }
}
newa<-cbind(popsize.time,popsize.year)
newa
```

```
##       popsize.time popsize.year
##  [1,]            0     15.00000
##  [2,]            1     15.75000
##  [3,]            2     16.53750
##  [4,]            3     17.36438
##  [5,]            4     18.23259
##  [6,]            5     19.14422
##  [7,]            6     20.10143
```

```
##  [8,]            7     21.10651
##  [9,]            8     22.16183
## [10,]            9     23.26992
## [11,]           10     24.43342
## [12,]           11     25.65509
## [13,]           12     26.93784
## [14,]           13     28.28474
## [15,]           14     29.69897
## [16,]           15     31.18392
## [17,]           16     32.74312
## [18,]           17     34.38027
## [19,]           18     36.09929
## [20,]           19     37.90425
## [21,]           20     39.79947
## [22,]           21     41.78944
## [23,]           22     43.87891
## [24,]           23     46.07286
## [25,]           24     48.37650
## [26,]           25     50.79532
## [27,]           26     53.33509
## [28,]           27     56.00184
## [29,]           28     58.80194
## [30,]           29     61.74203
## [31,]           30     64.82914
```

```
qplot(popsize.time,popsize.year,xlab="Year",
      ylab="Population Size",
      main="Discrete Constant Population Growth over 30 years")
```




```
#2b
lambda.2<-0.95
popsize.year.2<-numeric()
popsize.time<-numeric()
for(i in time.zero:end.time){
  popsize.year.2[i+1]<-pop.initial*lambda.2^i
  for(j in time.zero:end.time){
    popsize.time[j+1]<-time.zero+j
  }
}
newb<-cbind(popsize.time,popsize.year,popsize.year.2)
newb
```

```
##       popsize.time popsize.year popsize.year.2
##  [1,]            0     15.00000      15.000000
##  [2,]            1     15.75000      14.250000
##  [3,]            2     16.53750      13.537500
##  [4,]            3     17.36438      12.860625
##  [5,]            4     18.23259      12.217594
##  [6,]            5     19.14422      11.606714
##  [7,]            6     20.10143      11.026378
##  [8,]            7     21.10651      10.475059
##  [9,]            8     22.16183       9.951306
## [10,]            9     23.26992       9.453741
## [11,]           10     24.43342       8.981054
## [12,]           11     25.65509       8.532001
## [13,]           12     26.93784       8.105401
```

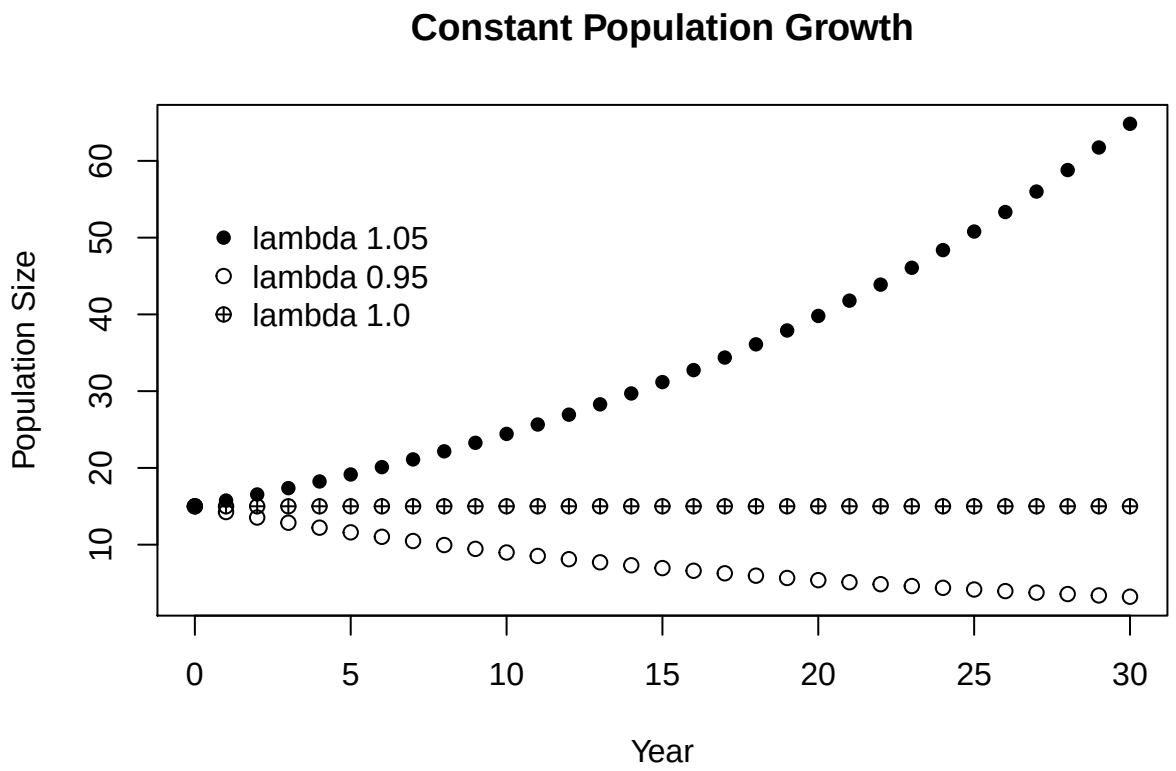
```
## [14,]           13     28.28474       7.700131
## [15,]           14     29.69897       7.315125
## [16,]           15     31.18392       6.949368
## [17,]           16     32.74312       6.601900
## [18,]           17     34.38027       6.271805
## [19,]           18     36.09929       5.958215
## [20,]           19     37.90425       5.660304
## [21,]           20     39.79947       5.377289
## [22,]           21     41.78944       5.108424
## [23,]           22     43.87891       4.853003
## [24,]           23     46.07286       4.610353
## [25,]           24     48.37650       4.379835
## [26,]           25     50.79532       4.160844
## [27,]           26     53.33509       3.952801
## [28,]           27     56.00184       3.755161
## [29,]           28     58.80194       3.567403
## [30,]           29     61.74203       3.389033
## [31,]           30     64.82914       3.219581
```

```
#2c
lambda.3<-1
popsize.year.3<-numeric()
popsize.time<-numeric()
for(i in time.zero:end.time){
  popsize.year.3[i+1]<-pop.initial*lambda.3^i
  for(j in time.zero:end.time){
    popsize.time[j+1]<-time.zero+j
  }
}
new<-cbind(popsize.time,popsize.year,popsize.year.2,popsize.year.3)
new
```

```
##       popsize.time popsize.year popsize.year.2 popsize.year.3
##  [1,]            0     15.00000      15.000000             15
##  [2,]            1     15.75000      14.250000             15
##  [3,]            2     16.53750      13.537500             15
##  [4,]            3     17.36438      12.860625             15
##  [5,]            4     18.23259      12.217594             15
##  [6,]            5     19.14422      11.606714             15
##  [7,]            6     20.10143      11.026378             15
##  [8,]            7     21.10651      10.475059             15
##  [9,]            8     22.16183       9.951306             15
## [10,]            9     23.26992       9.453741             15
## [11,]           10     24.43342       8.981054             15
## [12,]           11     25.65509       8.532001             15
## [13,]           12     26.93784       8.105401             15
## [14,]           13     28.28474       7.700131             15
## [15,]           14     29.69897       7.315125             15
## [16,]           15     31.18392       6.949368             15
## [17,]           16     32.74312       6.601900             15
## [18,]           17     34.38027       6.271805             15
## [19,]           18     36.09929       5.958215             15
## [20,]           19     37.90425       5.660304             15
## [21,]           20     39.79947       5.377289             15
```

```
## [22,]          21     41.78944       5.108424            15
## [23,]          22     43.87891       4.853003            15
## [24,]          23     46.07286       4.610353            15
## [25,]          24     48.37650       4.379835            15
## [26,]          25     50.79532       4.160844            15
## [27,]          26     53.33509       3.952801            15
## [28,]          27     56.00184       3.755161            15
## [29,]          28     58.80194       3.567403            15
## [30,]          29     61.74203       3.389033            15
## [31,]          30     64.82914       3.219581            15
```

```
#2 discrete plot with all three lambda values
new.use<-data.frame(new)
x<-new.use$popsize.time
y<-cbind(new.use$popsize.year,new.use$popsize.year.2,new.use$popsize.year.3)
matplot(x,y,type="p",pch=c(16,1,10),
        col=c(1,1,1),xlab="Year",
        ylab="Population Size",
        main="Constant Population Growth")
legend(0,55,legend = c("lambda 1.05","lambda 0.95","lambda 1.0"),pch=c(16,1,10),bty = "n")
```



Answer the following question: *Give 2 reasons why real population data may not follow these real curves, even if the growth rates on average are equal to the values in this figure.*

Catastrophes and bonanzas!

3. Stochastic Population Growth: Uniform Distribution (+15)

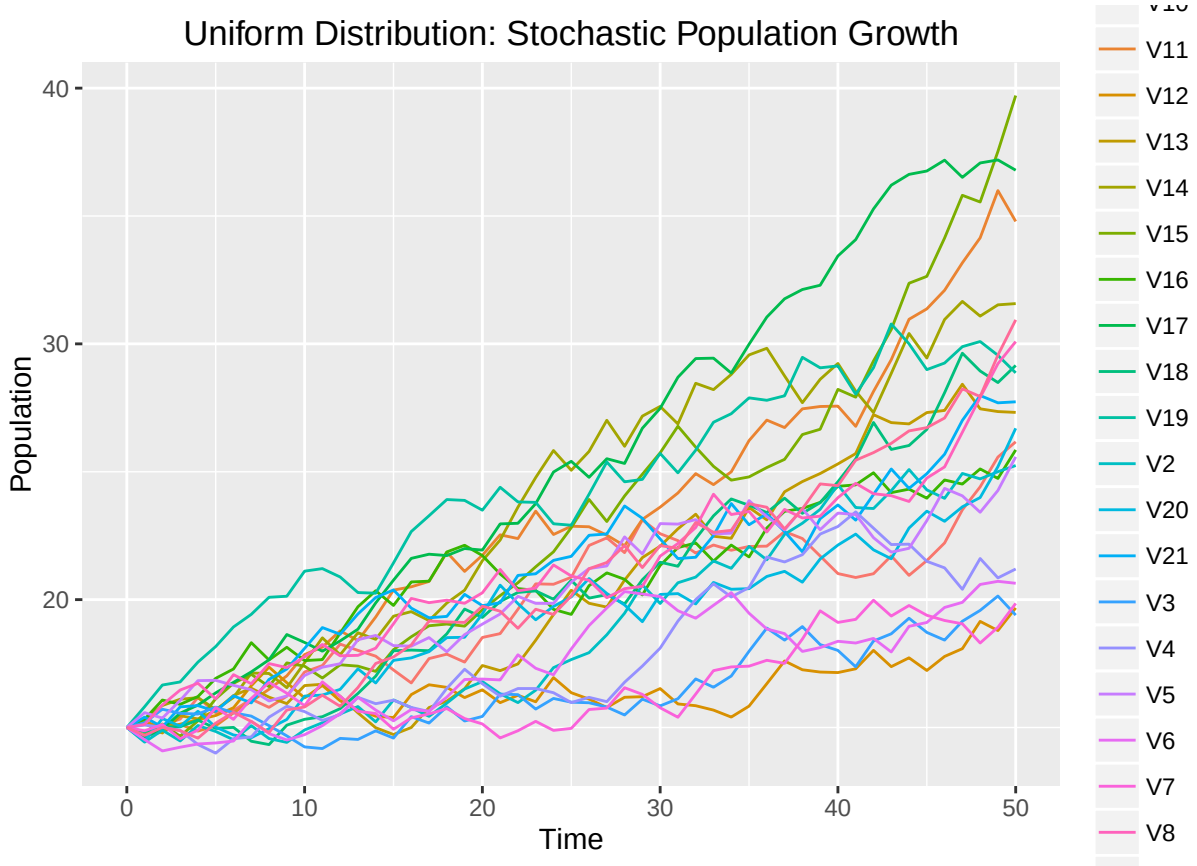
Construct a figure similar to page 16 on this week's lecture notes using the pseudocode on page 15.

```
set.seed(320)
pop.20 <- matrix(nrow = 51, ncol = 20)
pop.20[1,] <- 15
for (j in 1:dim(pop.20)[2]){
  for (i in 2:dim(pop.20)[1]) {
    current.pop <- pop.20[i-1,j]
    pop.20[i,j] = current.pop*runif(1, min = 0.96, max = 1.06)
}
}
time.20 <- seq(0, 51, by = 1)
pop.time.20 <- data.frame(cbind(time.20, pop.20))
```

```
## Warning in cbind(time.20, pop.20): number of rows of result is not a
## multiple of vector length (arg 1)
```

```
pop.20.gathered <- pop.time.20 %>%
  gather(Run, Population, 2:21)
colnames(pop.20.gathered) <- c("Time", "Run", "Population")

pop.20.plot <- ggplot(pop.20.gathered, aes(Time, Population, color = Run)) +
  geom_line()+ggtitle("Uniform Distribution: Stochastic Population Growth")
pop.20.plot
```



Answer the following question: *Adjust the code in this question to run 1000 trajectories to the fiftieth year. What is a 95% confidence interval for the population size in the 50th year? (HINT: Note I am not asking for*

a 95% confidence interval of the MEAN population size. You can answer this question in a similar way to bootstrapping exercises from last semester.)

```
set.seed(3100)
pop.1000 <- matrix(nrow=50, ncol = 1000)
pop.1000[1,] <- 15
for (j in 1:dim(pop.1000)[2]){
  for (i in 2:dim(pop.1000)[1]) {
    current.pop <- pop.1000[i-1,j]
    pop.1000[i,j] = current.pop*runif(1.01, min = 0.96, max = 1.06)
  }
}
Time2 <- seq(0, 50, by = 1)
plus.time <- cbind(Time2, pop.1000)
```

```
## Warning in cbind(Time2, pop.1000): number of rows of result is not a
## multiple of vector length (arg 1)
```

```
quantile(plus.time[50,2:1001], probs=c(0.025,0.975))
```

```
##     2.5%    97.5%
## 15.94179 35.49669
```

```
mean(plus.time[50,2:1001])
```

```
## [1] 24.43871
```

```
#24.43871 (95% CI 15.94179 to 35.49669)
```

95% CI is 15.94179 to 35.49669

4. Effect of Increasing Variation on Future Population Sizes (+15)

Adjust the code from above to run at least 5 more simulations where you assess how increasing variation in λ influences the probability of having a population with fewer than 15 individuals at year 50 (i.e. the starting population size in year zero). To do this, use 1000 trajectories and change the variation of λ by changing the $\Delta\lambda$ in the uniform distribution to increasingly larger numbers, then for each record how many out of 1000 are below 15 individuals at year 50. Finally, plot the probability of having fewer than 15 individuals vs. the variance of lambda for the six (or more simulations). *What does the figure show and why does this happen?* (HINT: As you adjust your $\Delta\lambda$ do not allow λ to be negative.)

```
#4-1
sum(plus.time[50,]<15)
```

```
## [1] 11
```

```
length(plus.time[50,])
```

```
## [1] 1001
```

```
11/1001
```

```
## [1] 0.01098901
```

```
#0.00999 -- variation is 0.10

set.seed(41)
pop.4 <- matrix(nrow=51, ncol = 1000)
pop.4[1,] <- 15
for (j in 1:dim(pop.4)[2]){
  for (i in 2:dim(pop.4)[1]) {
    current.pop.4 <- pop.4[i-1,j]
    pop.4[i,j] = current.pop.4*runif(1.01, min = 0.94, max = 1.08)
  }
}
Time2 <- seq(0, 51, by = 1)
plus.time.4 <- cbind(Time2, pop.4)
```

```
## Warning in cbind(Time2, pop.4): number of rows of result is not a multiple
## of vector length (arg 1)
```

```
sum(plus.time.4[51,]<15)
```

```
## [1] 54
```

```
length(plus.time.4[51,])
```

```
## [1] 1001
```

```
54/1001
```

```
## [1] 0.05394605
```

```
#0.0539 -- variation is 0.14

#4-2
set.seed(42)
pop.42 <- matrix(nrow=51, ncol = 1000)
pop.42[1,] <- 15
for (j in 1:dim(pop.42)[2]){
  for (i in 2:dim(pop.42)[1]) {
    current.pop.42 <- pop.42[i-1,j]
    pop.42[i,j] = current.pop.42*runif(1.01, min = 0.90, max = 1.12)
  }
}
Time2 <- seq(0, 51, by = 1)
plus.time.42 <- cbind(Time2, pop.42)
```

```
## Warning in cbind(Time2, pop.42): number of rows of result is not a multiple
## of vector length (arg 1)
```

```r
sum(plus.time.42[51,]<15)
```

```
## [1] 193
```

```r
length(plus.time.42[51,])
```

```
## [1] 1001
```

```r
193/1001
```

```
## [1] 0.1928072
```

```r
#0.1928-- variation is 0.22

#4-3
set.seed(43)
pop.43 <- matrix(nrow=51, ncol = 1000)
pop.43[1,] <- 15
for (j in 1:dim(pop.43)[2]){
  for (i in 2:dim(pop.43)[1]) {
    current.pop.43 <- pop.43[i-1,j]
    pop.43[i,j] = current.pop.43*runif(1.01, min = 0.85, max = 1.17)
  }
}
Time2 <- seq(0, 51, by = 1)
plus.time.43 <- cbind(Time2, pop.43)
```

```
## Warning in cbind(Time2, pop.43): number of rows of result is not a multiple
## of vector length (arg 1)
```

```r
sum(plus.time.43[51,]<15)
```

```
## [1] 351
```

```r
length(plus.time.43[51,])
```

```
## [1] 1001
```

```r
351/1001
```

```
## [1] 0.3506494
```

```r
#0.3506-- variation is 0.32

#4-4
set.seed(44)
pop.44 <- matrix(nrow=51, ncol = 1000)
pop.44[1,] <- 15
```

```
for (j in 1:dim(pop.44)[2]){
  for (i in 2:dim(pop.44)[1]) {
    current.pop.44 <- pop.44[i-1,j]
    pop.44[i,j] = current.pop.44*runif(1.01, min = 0.80, max = 1.22)
  }
}
Time2 <- seq(0, 51, by = 1)
plus.time.44 <- cbind(Time2, pop.44)
```

```
## Warning in cbind(Time2, pop.44): number of rows of result is not a multiple
## of vector length (arg 1)
```

```
sum(plus.time.44[51,]<15)
```

```
## [1] 448
```

```
length(plus.time.44[51,])
```

```
## [1] 1001
```

```
448/1001
```

```
## [1] 0.4475524
```

```
#0.4476-- variation is 0.42

#4-5
set.seed(45)
pop.45 <- matrix(nrow=51, ncol = 1000)
pop.45[1,] <- 15
for (j in 1:dim(pop.45)[2]){
  for (i in 2:dim(pop.45)[1]) {
    current.pop.45 <- pop.45[i-1,j]
    pop.45[i,j] = current.pop.45*runif(1.01, min = 0.75, max = 1.27)
  }
}
Time2 <- seq(0, 51, by = 1)
plus.time.45 <- cbind(Time2, pop.45)
```

```
## Warning in cbind(Time2, pop.45): number of rows of result is not a multiple
## of vector length (arg 1)
```

```
sum(plus.time.45[51,]<15)
```

```
## [1] 535
```

```
length(plus.time.45[51,])
```

```
## [1] 1001
```

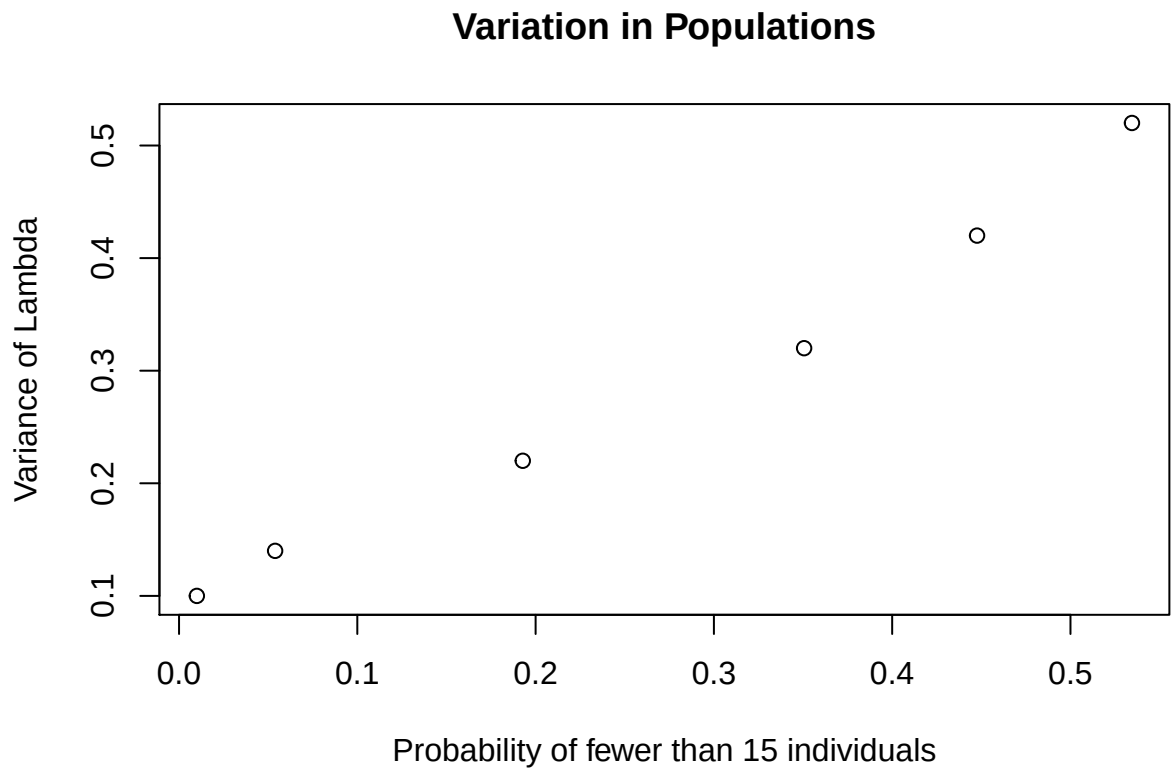
```r
535/1001
```

```
## [1] 0.5344655
```

```r
#0.5345-- variation is 0.52
#variance is max-min

#plot of variation vs delta lambdas
probability<-c(0.00999,0.0539,0.1928,0.3506,0.4476,0.5345)
variance<-c(0.10,0.14,0.22,0.32,0.42,0.52)

plot(probability,variance,xlab="Probability of fewer than 15 individuals",
     ylab="Variance of Lambda",main="Variation in Populations")
```




```r
#as variation increases the probability of getting a population size that is less than 15
#increases, this happens because the range in increasing
```

What does the figure show and why does this happen? As variation increases the probability of getting a population size that is less than 15 increases, this happens because the range is increasing.

5. Stochastic Population Growth: Lognormal Distribution (+20)

In actuality, a better model for stochastic variation in λ is the lognormal distribution rather than the uniform distribution. In other words, we assume that the natural logarithm of lambda is normally distributed. As a reminder, the lognormal distribution is defined by two parameters: the mean on the log scale and the standard deviation on the log scale.

(a) Draw 50 random values from a lognormal distribution where the mean on the log scale is 0.1 and the standard deviation on the log scale is 0.5. What are the (a) arithmetic mean, (b) geometric mean, and (c) variance of the lambda values?

```
set.seed(5)
ran.lambda<-rlnorm(50,0.1,0.5)
ran.lambda
```

```
##  [1] 0.7258385 2.2082043 0.5899332 1.1446185 2.6005434 0.8175412 0.8727700
##  [8] 0.8043783 0.9580198 1.1841842 2.0417661 0.7401594 0.6439100 1.0214598
## [15] 0.6466953 1.0309770 0.8198314 0.3708404 1.2465860 0.9707584 1.7336967
## [22] 1.7699206 2.3024665 1.5736229 1.6644662 0.9543346 2.2463227 2.3382129
## [29] 0.7956936 0.7215182 1.2942838 1.9248482 3.3458818 2.0310478 2.3154658
## [36] 1.7785295 0.6671327 0.4064736 0.4579053 1.0291116 2.3989477 0.7399212
## [43] 1.0647186 2.8514681 0.8796031 1.4639111 0.7092805 0.8779881 0.7693846
## [50] 1.0675800
```

```
lambda.arithmetric<-mean(ran.lambda)
lambda.geometric<-exp(mean(log(ran.lambda)))
lambda.variance<-var(ran.lambda)
lambda.geometric
```

```
## [1] 1.141642
```

```
lambda.arithmetric
```

```
## [1] 1.312255
```

```
lambda.variance
```

```
## [1] 0.508971
```

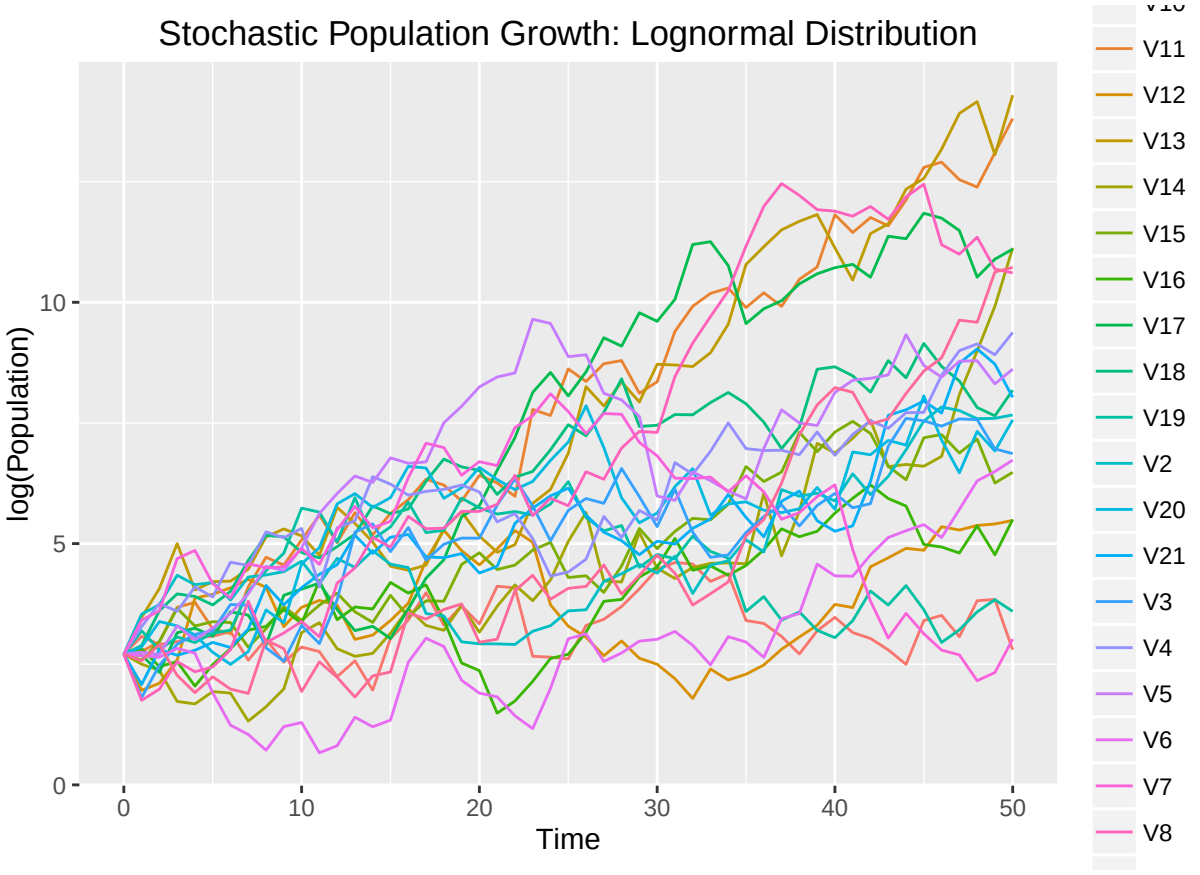
a) The arithmetic mean is 1.312255 b) The geometric mean is 1.141642 c) The variance is 0.508971

(b) Construct a figure similar to page 16 on this week's lecture notes using the pseudocode on page 15, but this time assume λ follows a lognormal distribution with parameters as described in part (a) and show the logarithm of population size on the y-axis.

```
stomat <- matrix(nrow = 51, ncol = 20)
stomat[1,] <- 15
for (j in 1:dim(stomat)[2]){
  for (i in 2:dim(stomat)[1]) {
    stobeg <- stomat[i-1,j]
    stomat[i,j] = stobeg*rlnorm(1, meanlog=0.1, sdlog = 0.5)
  }
}
time.55 <- seq(0, 51, by = 1)
time.5 <- as.data.frame(cbind(time.55, stomat))
```

```
## Warning in cbind(time.55, stomat): number of rows of result is not a
## multiple of vector length (arg 1)
```

```
sto<- time.5 %>%
  gather(Run, Population, 2:21)
colnames(sto) <- c("Time", "Run", "Population")
sto.plot <- ggplot(sto, aes(Time, log(Population), color = Run)) +geom_line()+ggtitle("Stochastic Popula
sto.plot
```



(c) Based on your figure, why do you think this type of stochastic variation is known as the "diffusion model" of population growth?

This type of stochastic variation is know as the "diffusion model" of population growth because as time moves forward the variation of population sizes grows dramatically. This is similar to the process of diffusion in physics when molecules or atoms move from a region of high concentration to a region of low concentration and spread out in the process. The word diffusion derives from the Latin word, diffundere, which means "to spread out", which perfectly describes what happens to the possible population sizes in this model.

6. Relation between Arithmetic and Geometric Mean (+15)

In this question you are going to assess whether the relationship between the geometric mean, arithmetic mean, and variance of λ on page 24 of the notes holds.

```
loglam <- rlnorm(50, meanlog = 0.1, sdlog = 0.5)
loglam.var <- var(loglam)
loglam.arithmetic <- mean(loglam)
loglam.geometric <- exp(mean(log(loglam)))
```

(a) Algebraically rearrange the equation so that all arithmetic mean and geometric mean variables are on the left hand side (i.e. y) and variance is on the right hand side (x).

```
log(loglam.geometric/loglam.arithmetic)*(-2*loglam.arithmetic^2) = loglam.var
```

$-2{\lambda_a}^2 * -ln(\frac{\lambda_g}{\lambda_a}) = Var(\lambda_t)$

(b) What should the slope and intercept be of the relationship between the left hand side (y) and the variance of λ?

The slope will be: $-ln(\frac{\lambda_g}{\lambda_a})$ **The intercept will be:** $-2{\lambda_a}^2$

(c) Complete 10 simulations from a lognormal distribution where you change the mean and standard deviation on the log scale and then calculate the arithmetic mean, geometric mean and variance of λ for each of the ten simulations. Then calculate the left hand side of the equation in (a). Record the four values for each simulation in a table where you report the mean and standard deviation on the log scale and the arithmetic mean, geometric mean, variance, and left hand side.

```
mean.seq <- seq(from = 0.06, to = 0.15, by = 0.01)
sd.seq <- seq(from = 0.46, to = 0.55, by = 0.01)
loglam.mat <- matrix(nrow = 10, ncol = 6)
for (j in 1:dim(loglam.mat)[1]){
        loglam.mat[j,1] <- mean.seq[j]
        loglam.mat[j,2] <- sd.seq[j]
        sample <- rlnorm(50, meanlog = mean.seq[j], sdlog = sd.seq[j])
        loglam.mat[j,3] <- mean(sample)
        loglam.mat[j,4] <- exp(mean(sample))
        loglam.mat[j,5] <- var(sample)
        loglam.mat[j,6] <- log(loglam.mat[j,3]/loglam.mat[j,4])*(-2*loglam.mat[j,3]^2)
}
colnames(loglam.mat) <- c("logmean", "logsd", "amean", "gmean", "var", "left")
loglam.frame <- data.frame(loglam.mat)
```

(d) Perform a regression where you regress the left hand side as y and variance of lambda as x. Are the results of the regression consistent with your answer in (b)? Support your answer statistically with appropriate confidence intervals.

```
loglam.reg <- glm(left ~ var, data = loglam.frame)
summary(loglam.reg)
```

```
##
## Call:
## glm(formula = left ~ var, data = loglam.frame)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -0.70445  -0.51818  -0.09005   0.54767   0.81746
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1623     0.5968   3.623  0.00675 **
## var           2.8529     1.1338   2.516  0.03602 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.3791215)
##
##     Null deviance: 5.4334  on 9  degrees of freedom
## Residual deviance: 3.0330  on 8  degrees of freedom
## AIC: 22.448
##
## Number of Fisher Scoring iterations: 2
```

7. Discrete Population Growth with Density Dependence (+15) (Modified from Bill Morris)

Note that our model of population growth, although it now allows for stochasticity in λ, it does not include density-dependence. The Ricker equation is commonly used in fisheries to represent the dynamics of a density-dependent population. The equation is:

$N_{t+1} = N_t e^{r(1-\frac{N_t}{K})}$

Your assignment is to write a program in R to explore the dynamics of the Ricker equation.

To do this, you will need to start by specifying the parameters (i.e., constants) r and K, as well as the starting population size N_0 and the maximum time, $t_{\max}$.

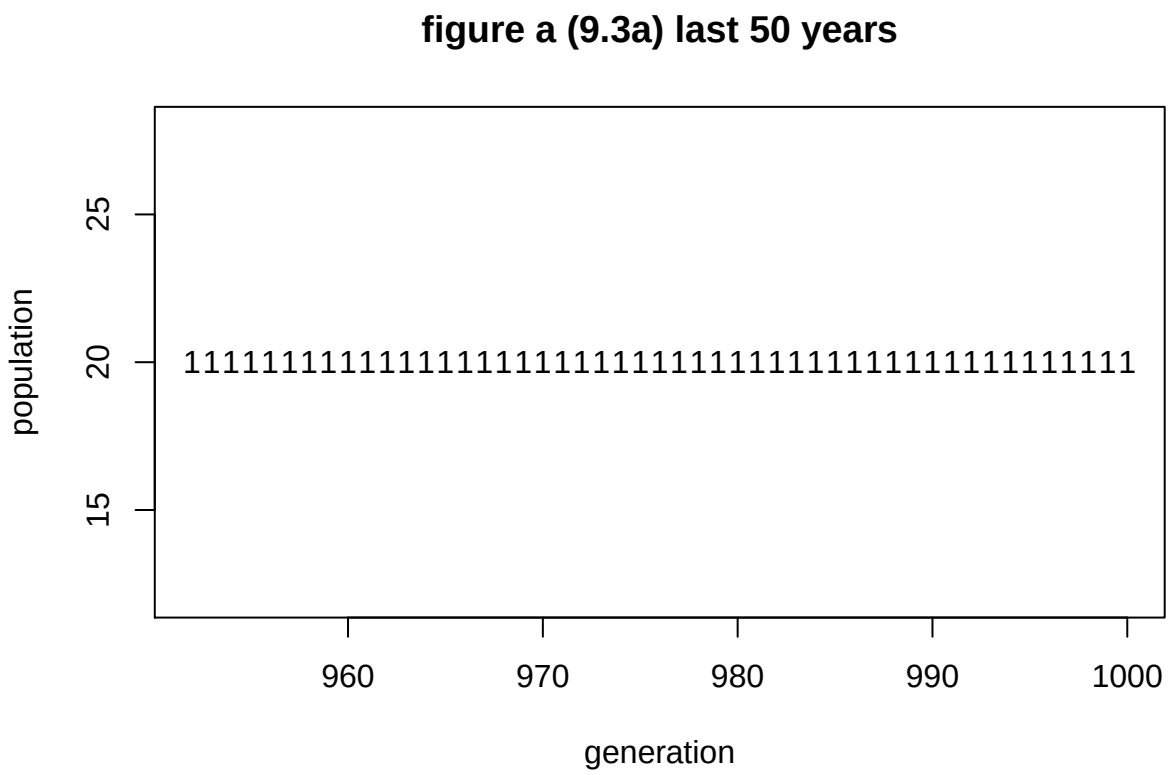
Start with r = 1 and K = 20. Set N_0 to 10. Choose $t_{\max}$ to be large (say 1000).

Then create a vector to store all the $t_{\max} + 1$ values of N (i.e. including N_0). Then calculate all the future population sizes and store them in the vector. Note this is very similar to Q2 above, just a different (and more complicated) relationship between N_{t+1} and N_t.

Now plot the last 50 years of the population trajectory, using the `matplot()` function.

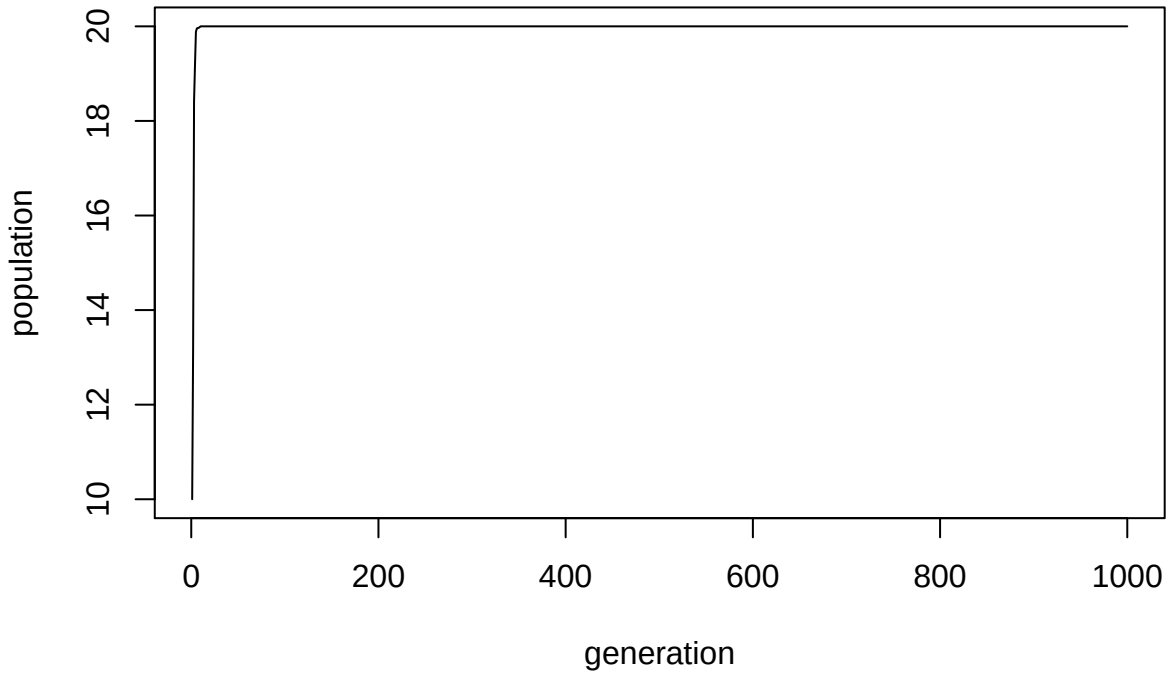
```
Na<-matrix(nrow=1001,ncol=1)
ran.lambda<-matrix(nrow = 1001,ncol=1)
ran.lambda[1,1]<- runif(1,-1,0)
gen.a<-matrix(nrow=1001,ncol=1)
gen.a[1,1]<-0
r<-1
K<-20
Na[1,1]<-10#NO
tmax=1000
for(i in 2:1001){
  gen.a[i-1,1]<-i-1
  ran.lambda[i,1]<- runif(1,0,1)
  r<-ran.lambda[i,1]
  Na[i,1]=Na[i-1,1]*exp(r*(1-Na[i-1,1]/K))
}
```

```
data.a<-cbind(gen.a,ran.lambda,Na)
matplot(data.a[952:1001,1],data.a[952:1001,3], type="p",
        xlab="generation",ylab="population",
        main="figure a (9.3a) last 50 years")
```

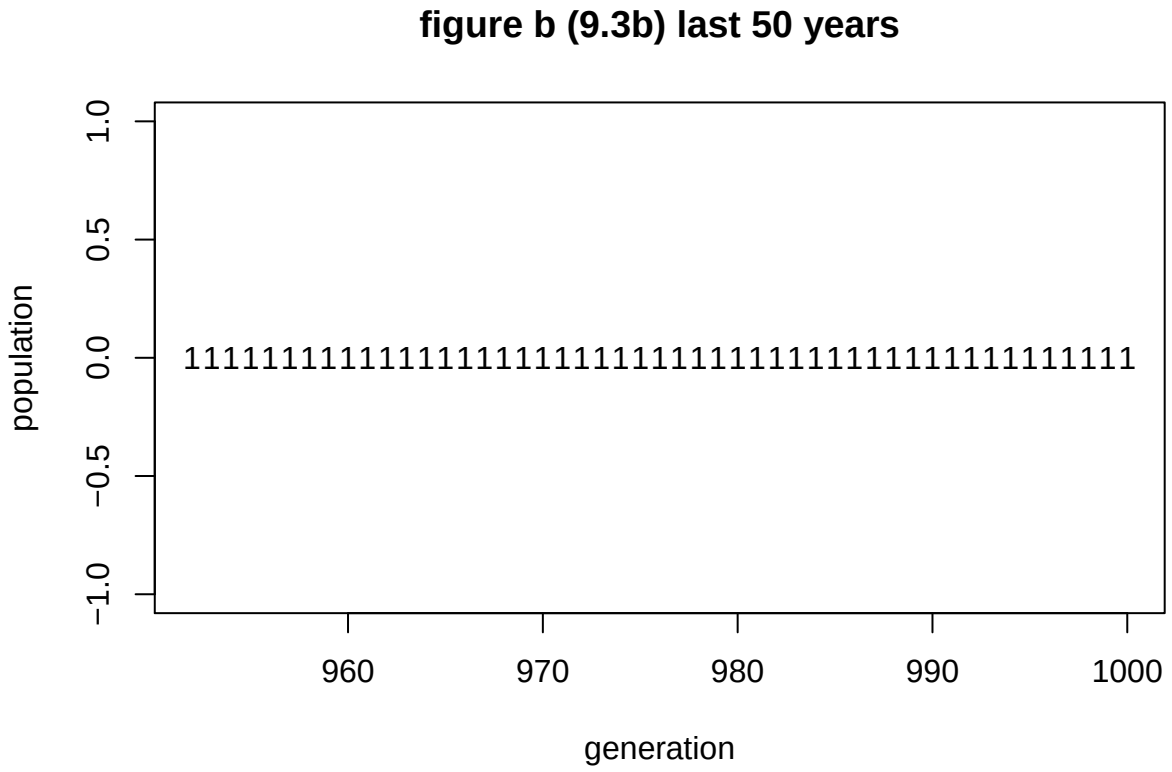



```
matplot(data.a[,1],data.a[,3], type="l",
        xlab="generation",ylab="population",
        main="figure 9.3a")
```

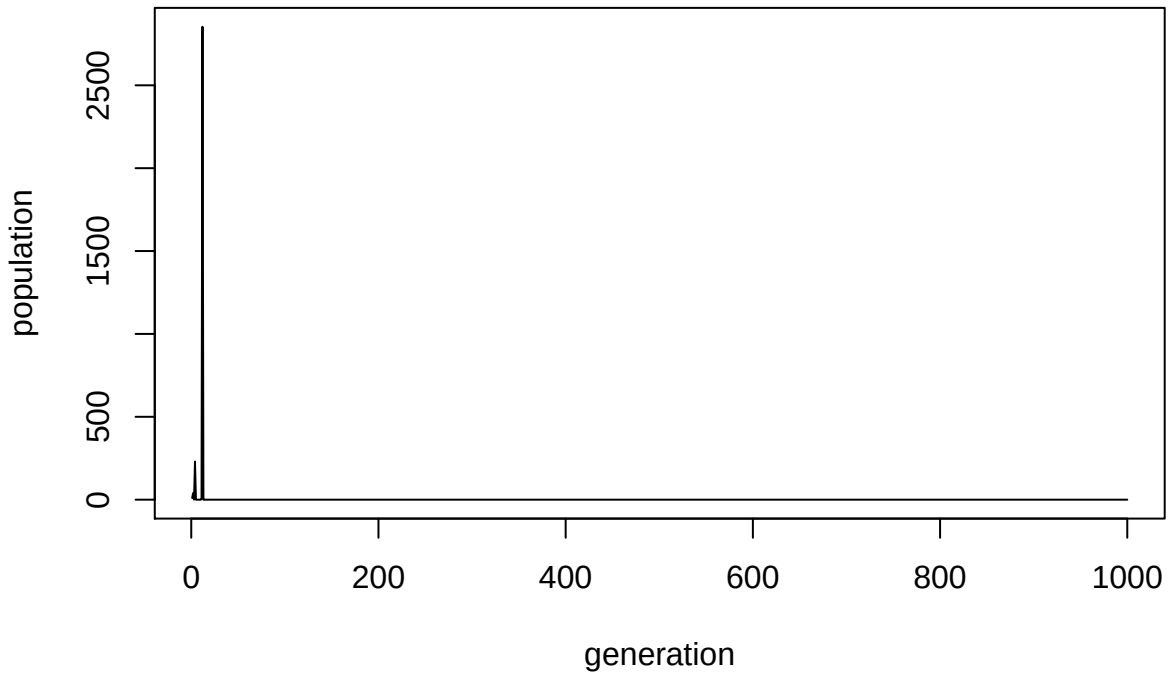
figure 9.3a



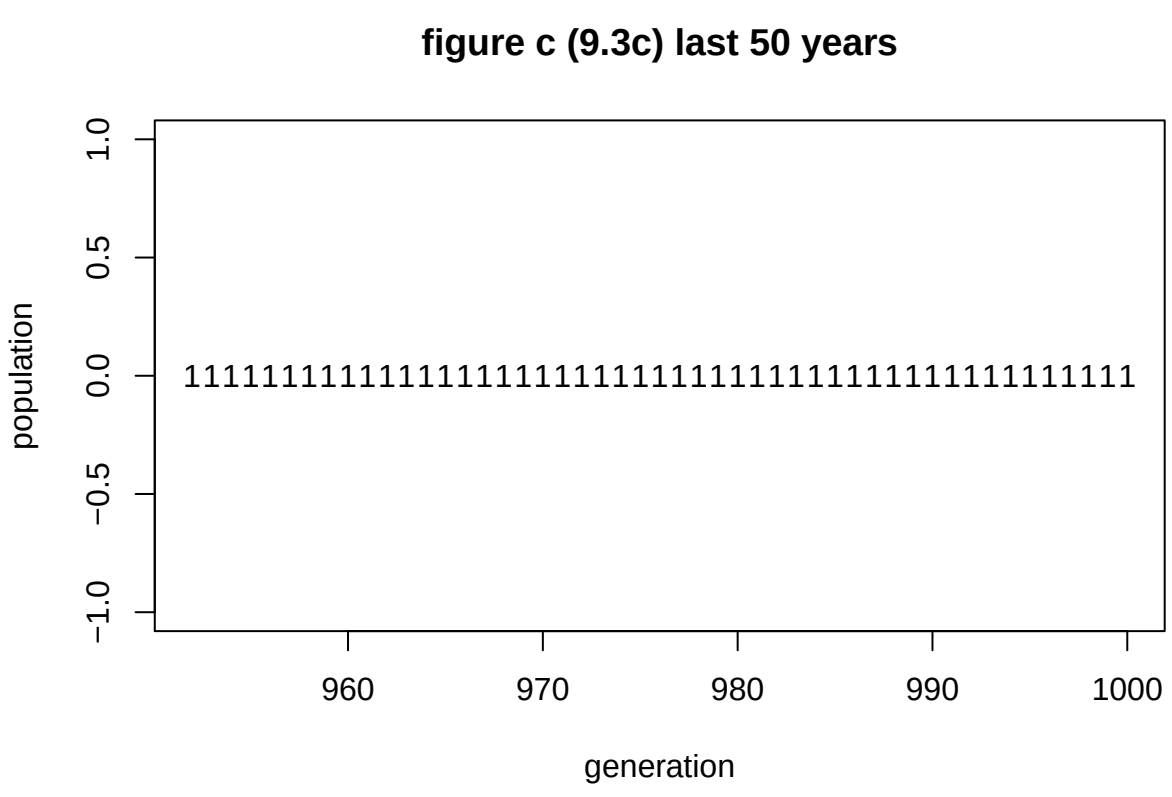

```
#figure 9.3b
Nb<-matrix(nrow=1001,ncol=1)
ran.lambda<-matrix(nrow = 1001,ncol=1)
ran.lambda[1,1]<- runif(1,-1,0)
gen.b<-matrix(nrow=1001,ncol=1)
gen.b[1,1]<-0
r<-1
K<-20
Nb[1,1]<-10#N0
tmax=1000
for(i in 2:1001){
  gen.b[i-1,1]<-i-1
  ran.lambda[i,1]<- runif(1,1,10)
  r<-ran.lambda[i,1]
  Nb[i,1]=Nb[i-1,1]*exp(r*(1-Nb[i-1,1]/K))
}
data.b<-cbind(gen.b,ran.lambda,Nb)
matplot(data.b[952:1001,1],data.b[952:1001,3], type="p",
        xlab="generation",ylab="population",
        main="figure b (9.3b) last 50 years")
```



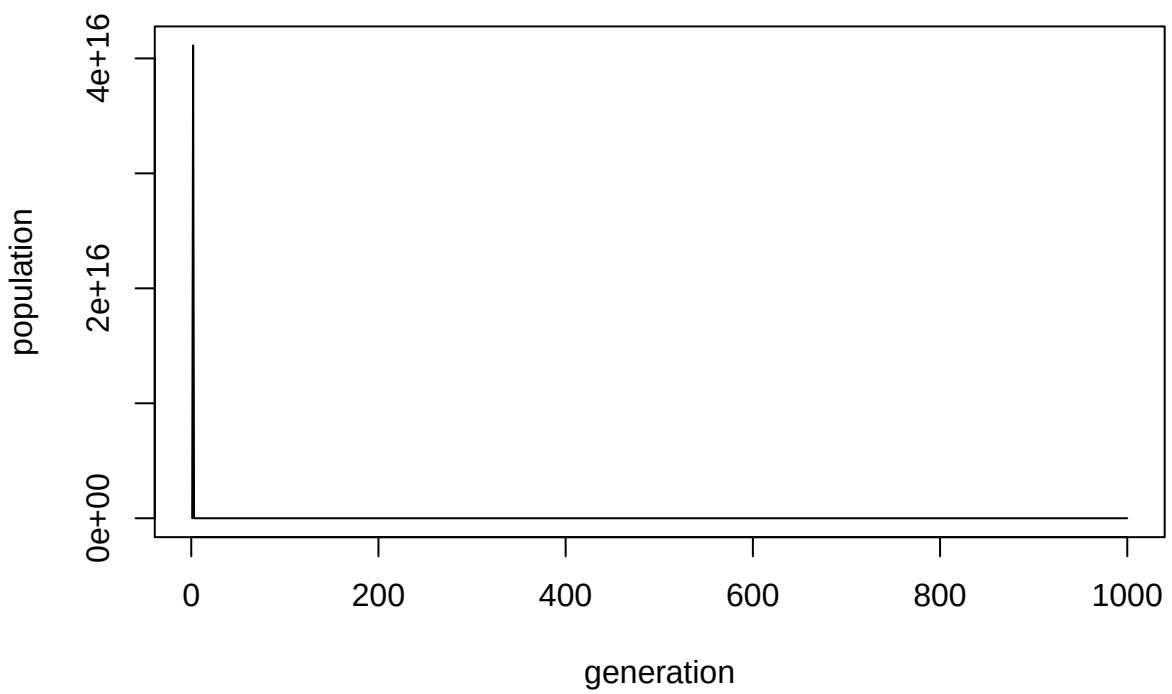

```
matplot(data.b[,1],data.b[,3], type="l",
        xlab="generation",ylab="population",
        main="figure 9.3b")
```



```r
#figure 9.3c
Nc<-matrix(nrow=1001,ncol=1)
ran.lambda<-matrix(nrow = 1001,ncol=1)
ran.lambda[1,1]<- runif(1,-1,0)
gen.c<-matrix(nrow=1001,ncol=1)
gen.c[1,1]<-0
r<-1
K<-20
Nc[1,1]<-10#N0
tmax=1000
for(i in 2:1001){
  gen.c[i-1,1]<-i-1
  ran.lambda[i,1]<- runif(1,10,100)
  r<-ran.lambda[i,1]
  Nc[i,1]=Nc[i-1,1]*exp(r*(1-Nc[i-1,1]/K))
}
data.c<-cbind(gen.c,ran.lambda,Nc)
matplot(data.c[952:1001,1],data.c[952:1001,3], type="p",
        xlab="generation",ylab="population",
        main="figure c (9.3c) last 50 years")
```




```r
matplot(data.c[,1],data.c[,3], type="l",
        xlab="generation",ylab="population",
        main="figure 9.3c")
```



Once your program is working, use it to slowly increase the value of r, noting when the dynamics in the plot changes. Describe the pattern of how the dynamics change as r increases. Ideally you should be able to get the three figures in Figure 9.3 of the Krebs reading. (+5 for each graph successfully achieved). The big idea here is that density-dependence can introduce a wide variety of population growth trajectories!!