Stochastic population dynamics

These exercises require the latest version of `popdemo`. Head to `popdemo`'s GitHub (github.com/iainmstott/popdemo) for installation instructions (don't forget also to load the package using `library(popdemo)`!)

Complete the core exercises (in normal print) first, as code in each section continues from the last. Afterward, return to the "extras" sections (in *italics*), and try writing your own code. Code to be run is in chunks:

```
# a comment
an_input()
##   an output
```

Key terms in the text are in ***bold italic***, and functions or arguments are `fixed width`.

---

1. Data

We will use a set of matrices for the polar bear *Ursus maritimus*[1]. The population is found in the southern Beaufort Sea, USA and Canada. There are 6 stages based on age and reproductive status:
Stage 1: 2-year-old
Stage 2: 3-year-old
Stage 3: 4-year-old
Stage 4: adult (5+ years old), available to breed
Stage 5: adult, with cub (0-1 years old)
Stage 6: adult, with yearling (1-2 years old).

Load in the data

```
data(Pbear); Pbear
##   $Y2001
##          2YO    3YO    4YO     Ad    AdC    AdY
##   2YO 0.0000 0.0000 0.0000 0.0000 0.0000 0.5811
##   3YO 0.9858 0.0000 0.0000 0.0000 0.0000 0.0000
##   4YO 0.0000 0.9858 0.0000 0.0000 0.0000 0.0000
##   Ad  0.0000 0.0000 0.9858 0.5061 0.3791 0.9918
##   AdC 0.0000 0.0000 0.0000 0.4857 0.0681 0.0000
##   AdY 0.0000 0.0000 0.0000 0.0000 0.5433 0.0000
##
##   $Y2002
##          2YO    3YO    4YO     Ad    AdC    AdY
##   2YO 0.0000 0.0000 0.0000 0.0000 0.0000 0.5800
##   3YO 0.9842 0.0000 0.0000 0.0000 0.0000 0.0000
##   4YO 0.0000 0.9842 0.0000 0.0000 0.0000 0.0000
##   Ad  0.0000 0.0000 0.9842 0.4563 0.3654 0.9911
##   AdC 0.0000 0.0000 0.0000 0.5348 0.0808 0.0000
##   AdY 0.0000 0.0000 0.0000 0.0000 0.5435 0.0000
##
##   $Y2003
##          2YO    3YO    4YO     Ad    AdC    AdY
##   2YO 0.0000 0.0000 0.0000 0.0000 0.0000 0.5379
##   3YO 0.9415 0.0000 0.0000 0.0000 0.0000 0.0000
##   4YO 0.0000 0.9415 0.0000 0.0000 0.0000 0.0000
##   Ad  0.0000 0.0000 0.9415 0.2840 0.3081 0.9662
##   AdC 0.0000 0.0000 0.0000 0.6822 0.1384 0.0000
##   AdY 0.0000 0.0000 0.0000 0.0000 0.5160 0.0000
##
```

[1] Hunter et al. (2010) Ecology, 91, 2883-2897.

```
##   $Y2004
##          2YO    3YO    4YO     Ad    AdC    AdY
##   2YO 0.0000 0.0000 0.0000 0.0000 0.0000 0.2773
##   3YO 0.6578 0.0000 0.0000 0.0000 0.0000 0.0000
##   4YO 0.0000 0.6578 0.0000 0.0000 0.0000 0.0000
##   Ad  0.0000 0.0000 0.6578 0.5367 0.4243 0.7587
##   AdC 0.0000 0.0000 0.0000 0.2220 0.0327 0.0000
##   AdY 0.0000 0.0000 0.0000 0.0000 0.2689 0.0000
##
##   $Y2005
##          2YO    3YO    4YO     Ad    AdC    AdY
##   2YO 0.0000 0.0000 0.0000 0.0000 0.0000 0.3165
##   3YO 0.7034 0.0000 0.0000 0.0000 0.0000 0.0000
##   4YO 0.0000 0.7034 0.0000 0.0000 0.0000 0.0000
##   Ad  0.0000 0.0000 0.7034 0.7225 0.4328 0.7943
##   AdC 0.0000 0.0000 0.0000 0.0718 0.0081 0.0000
##   AdY 0.0000 0.0000 0.0000 0.0000 0.3254 0.0000
```

The study also gives the population vector, which is the relative numers of individuals in each stage.

```
Pbearvec <- c(0.106, 0.068, 0.106, 0.461, 0.151, 0.108)
```

The polar bear matrices have a mixed age-stage definition: the stages 1-3 are defined by age, whereas stages 4-6 are defined by a mix of age and reproductive stage. Each matrix describes life cycle transitions in a *specific* year, from 2001-2005. Due to varying environmental conditions (e.g. weather, resource availability), life cycle transitions may change from year to year. For the polar bear, 2004 and 2005 were "bad years" due to low summer sea ice.

The numbers in the matrix (called ***matrix elements*** or ***transitions***) describe the probability of moving FROM stages in each column TO stages in each row, within the time interval chosen. For example, for the 2005 polar bear matrix, an adult (stage 4) has 7.18% probability of breeding and becoming an adult with a cub (stage 5). Likewise, in any year an adult without a cub has about 72.25% chance of not breeding (therefore remaining in stage 4): this is called a stasis transition. This means that 100 - (72.25 + 7.18) = 20.57% of adults without cubs die every year. Contrast these numbers with "good" years, where reproduction is high and mortality is lower. There are different types of transitions: in this matrix there are also fecundity transitions which describe recruitment of offspring (at an age of 2 years; this is the top-right transition in the matrix). Other species may have different transitions, including skipping stages through fast growth, shrinkage or fission (especially in modular organisms, e.g. most plants, corals), or asexual reproduction.

Matrix elements combine underlying ***vital rates*** such as survival, growth and reproduction. For example, the stasis transition of adults without cubs (72.25%) combines probability of not breeding successfully, and the probability of surviving the year.

Polar bear (photo by NASA Goddard Space Flight Center from Greenbelt, MD, USA)

---

2. Stochastic projections

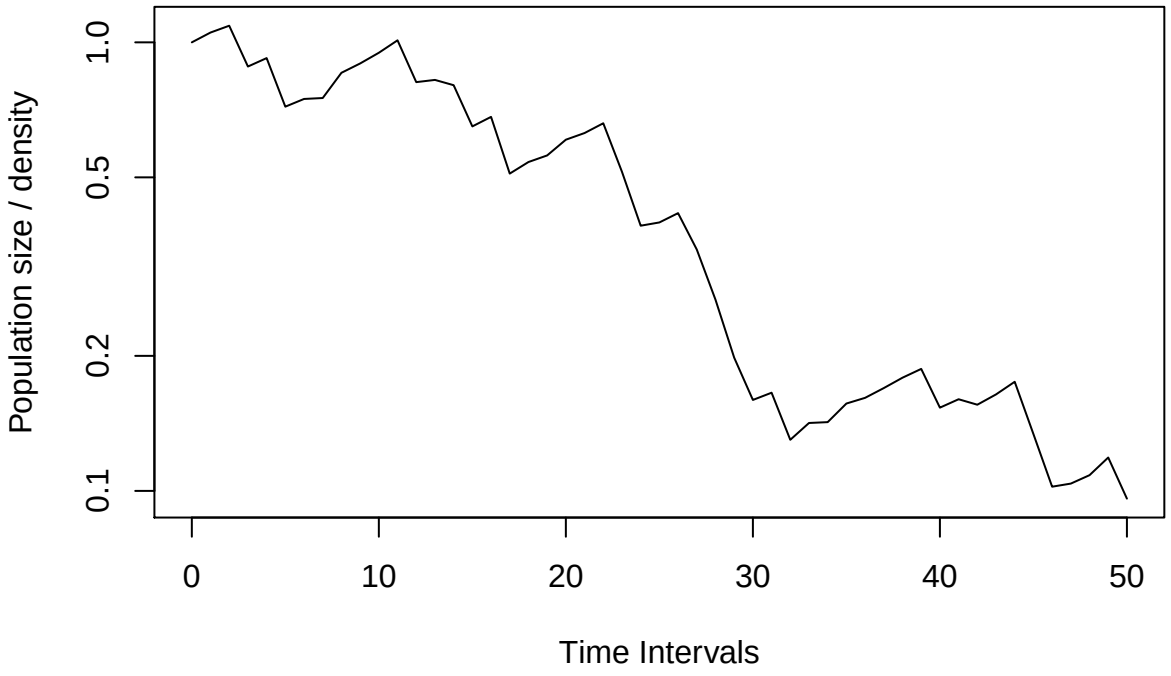
This vignette builds on the "Deterministic population dynamics" vignette, in which a single matrix is used to project population dynamics over time. Using just a single matrix to project population dynamics makes the assumption that ***vital rates*** such as survival, reproduction and growth stay fixed over time. It's a big assumption that over years or decades, the vital rates of a population won't change. One reason that they may change is because of environmental stochasticity, where environmental conditions, which change from year to year, impact vital rates. This could be due to many things, such as changes in weather or extreme weather events, disease outbreaks, varying resource availability, changes in predation or parasitism, and so on. If the vital rates change each year, then the matrix changes each year. The projection equation is:

$$\mathbf{n}_{t+1} = \mathbf{A}_t \mathbf{n}_t$$

This is called a stochastic population projection.

In the polar bear data, there are 5 matrices from 2001-2005. If you pass the `project` function a single matrix then it does a deterministic projection (see the "Deterministic Population Dynamics" vignette), but if you pass it a list of matrices as we have for the Polar bear, then it will do a stochastic projection:

```
Pbearp1.1 <- project(Pbear, Pbearvec, time = 50)
plot(Pbearp1.1, log = "y")
```



For a stochastic projection, the `project` function should usually have a list of matrices passed to the `A` argument, and a `vector`. The function will check that all the matrices have the same dimensions. `vector` can be either a single vector (as in the above case), or a matrix of several vectors, where each individual column represents one vector. The `time` argument gives the number of projection intervals, but the output will have length `time` + 1, because the population size at time 0 is included. There is one further important argument called `Aseq`, which determines the sequence of matrices chosen for the projection. By default, `Aseq = "unif"`. It's easy to see what the random sequence of matrices chosen is:

```
Aseq(Pbearp1.1)
##   Y2002 Y2001 Y2005 Y2001 Y2005 Y2003 Y2001 Y2002 Y2002 Y2001 Y2002 Y2005
##       2     1     5     1     5     3     1     2     2     1     2     5
##   Y2003 Y2003 Y2004 Y2002 Y2004 Y2002 Y2002 Y2001 Y2003 Y2002 Y2004 Y2004
##       3     3     4     2     4     2     2     1     3     2     4     4
##   Y2001 Y2002 Y2005 Y2004 Y2004 Y2005 Y2002 Y2005 Y2002 Y2001 Y2001 Y2003
##       1     2     5     4     4     5     2     5     2     1     1     3
##   Y2001 Y2003 Y2002 Y2005 Y2001 Y2003 Y2003 Y2003 Y2004 Y2004 Y2002 Y2001
##       1     3     2     5     1     3     3     3     4     4     2     1
##   Y2002 Y2005
##       2     5
```

In this vignette, we'll look at different ways of determining the sequence of matrices chosen in a projection, and what the consequences are.

2.1 Matrix selection using Markov processes

By default, the projection selects each matrix with an equal probability at each time interval. This process can be considered a ***Markov process*** with the following transition matrix:

```
p1 <- 0.4
( PbearM1 <- matrix(rep(c((1-p1)/3, (1-p1)/3, (1-p1)/3, p1/2, p1/2), 5), 5, 5) )
##        [,1] [,2] [,3] [,4] [,5]
##   [1,]  0.2  0.2  0.2  0.2  0.2
```

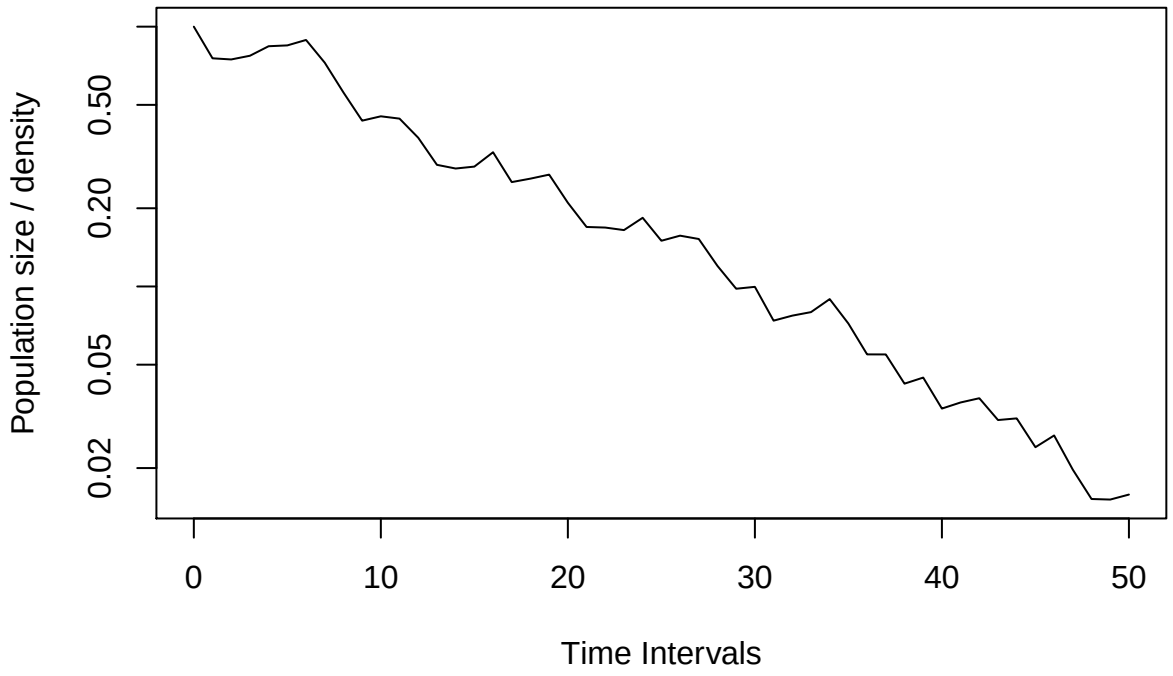
```
##    [2,]  0.2  0.2  0.2  0.2  0.2
##    [3,]  0.2  0.2  0.2  0.2  0.2
##    [4,]  0.2  0.2  0.2  0.2  0.2
##    [5,]  0.2  0.2  0.2  0.2  0.2
```

The matrix determines the probability of selecting any matrix at random based on what the previous selection was. It's defined columnwise: at each timestep, the column represents what the previous selection (at time t) was, and the rows in that column tell you the probability of selecting the next matrix (at time t+1). In this case, all the numbers in the matrix are equal to one another, so there is exactly the same probability of selecting each matrix each time, regardless of what the last selection was.

As 2004 and 2005 were "bad" years, `p1` represents the probability of selecting a "bad" year (0.4) whilst `1 - p1` represents the probability of selecting a "good" year (0.6).

We can pass this matrix to the `Aseq` argument of the function to get a new estimate of population dynamics:

```
Pbearp2.1 <- project(Pbear, Pbearvec, Aseq = PbearM1, time = 50)
plot(Pbearp2.1, log = "y")
```



This projection will look different to the previous one, because a new random sequence of matrices is generated. However, it is generated from the same set of parameters so overall it should look fairly similar (and later in the "Stochastic dynamics" section we'll see that it is the same).

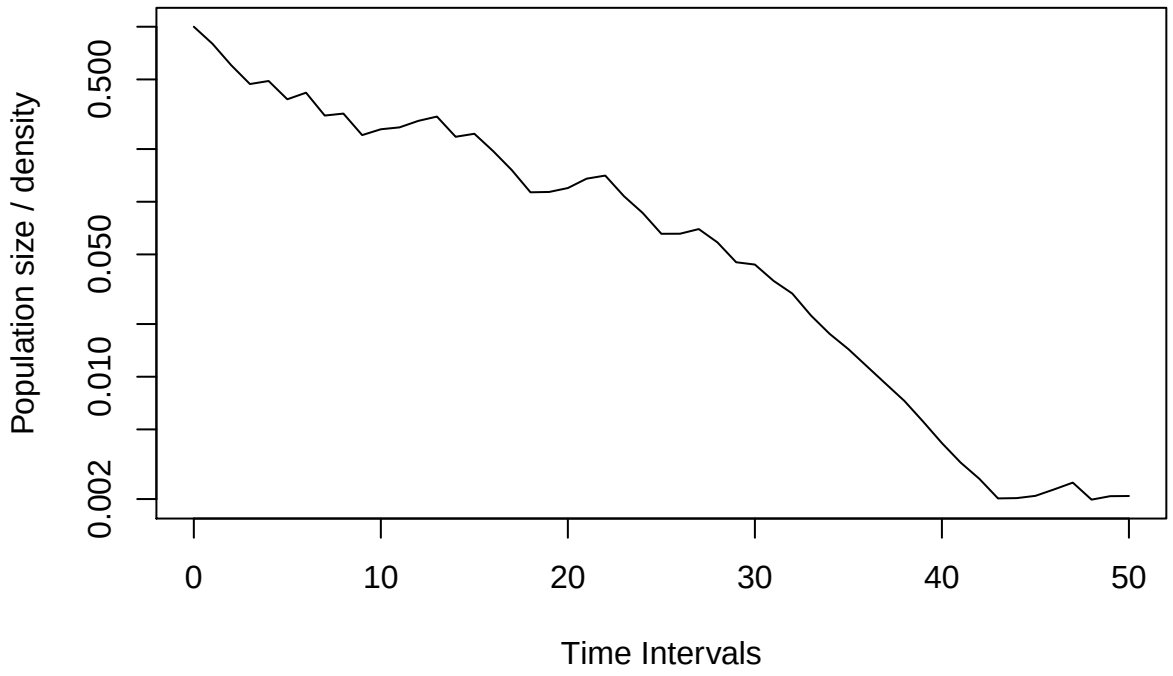
There are situations for some systems where we might want to have unequal probabilities of choosing each matrix (e.g. nonrandom patterns in environmental conditions, behaviour, or food availability). In the polar bear data, years '04 and '05 were "bad" years. If bad years occur with probability p, then we can construct a Markov transition matrix incorporating this:

```
p2 <- 0.5
( PbearM2 <- matrix(rep(c((1-p2)/3, (1-p2)/3, (1-p2)/3, p2/2, p2/2), 5), 5, 5) )
##          [,1]   [,2]   [,3]   [,4]   [,5]
##    [1,] 0.1667 0.1667 0.1667 0.1667 0.1667
##    [2,] 0.1667 0.1667 0.1667 0.1667 0.1667
##    [3,] 0.1667 0.1667 0.1667 0.1667 0.1667
```

```
##    [4,] 0.2500 0.2500 0.2500 0.2500 0.2500
##    [5,] 0.2500 0.2500 0.2500 0.2500 0.2500
```

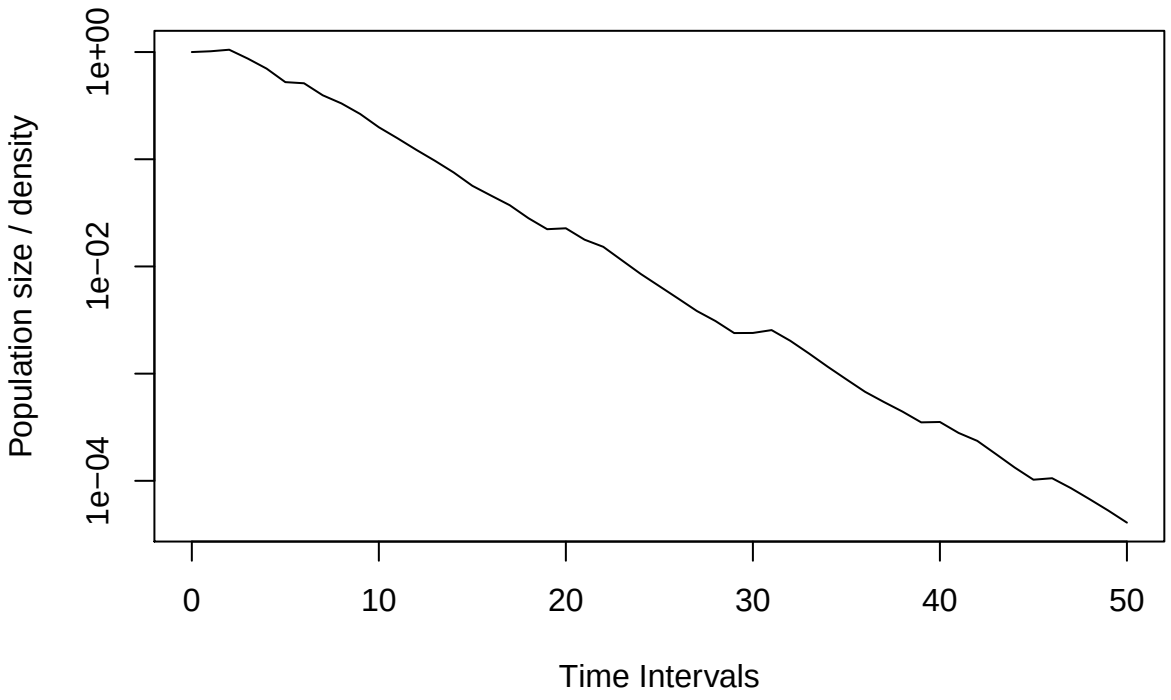
In this case, p = 0.5 means that there is now an equal probability of bad and good years. We can pass this to the `Aseq` argument of the function to get a new estimate of population dynamics:

```
Pbearp2.2 <- project(Pbear, Pbearvec, Aseq = PbearM2, time = 50)
plot(Pbearp2.2, log = "y")
```



This probably doesn't look too much different from the first projection, because in the first projection bad years have a probability of $2/5 = 0.4$ chance of being chosen, but in the second they have a probability of 0.5 of being chosen; the two are not very different. With climate change there's a good chance that bad years might become more frequent. What happens if we increase the probability of bad years so that they occur four times as often as good years?

```
p3 <- 0.8
( PbearM3 <- matrix(rep(c((1-p3)/3, (1-p3)/3, (1-p3)/3, p3/2, p3/2), 5), 5, 5) )
##           [,1]    [,2]    [,3]    [,4]    [,5]
##    [1,] 0.06667 0.06667 0.06667 0.06667 0.06667
##    [2,] 0.06667 0.06667 0.06667 0.06667 0.06667
##    [3,] 0.06667 0.06667 0.06667 0.06667 0.06667
##    [4,] 0.40000 0.40000 0.40000 0.40000 0.40000
##    [5,] 0.40000 0.40000 0.40000 0.40000 0.40000

Pbearp2.3 <- project(Pbear, Pbearvec, Aseq = PbearM3, time = 50)
plot(Pbearp2.3, log = "y")
```



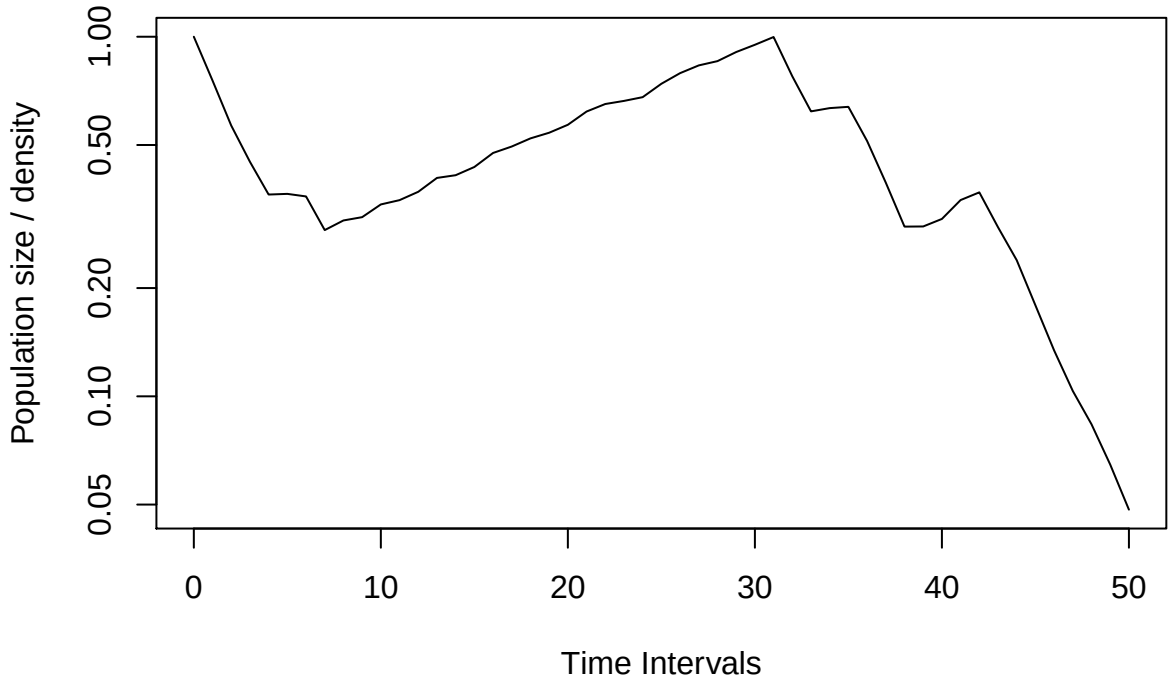
You should clearly see a steeper decline in population size as the probability of bad years increases.

At the moment, the probability a matrix is chosen doesn't depend on what the last matrix chosen was: we can tell this because all the columns are the same. The 2004 matrix always has the same probability of being chosen, regardless of whether the last one chosen was 2001, 2002, 2003, 2004 or 2005. Imagine, however, that there's positive feedback in the weather systems that determine ice cover, so that good years are more likely to follow good years and bad years are more likely to follow bad years. We can define two parameters now: p, which describes the probability next year is a bad year given that this year was a good year, and q, which is the probability next year is a bad year given that this year was a bad year:

```
p4 <- 0.2
q4 <- 0.8
( PbearM4 <- matrix(c(rep(c((1-p4)/3, (1-p4)/3, (1-p4)/3, p4/2, p4/2), 3),
                      rep(c((1-q4)/3, (1-q4)/3, (1-q4)/3, q4/2, q4/2), 2)),
                    5, 5) )
##           [,1]   [,2]   [,3]    [,4]    [,5]
##   [1,] 0.2667 0.2667 0.2667 0.06667 0.06667
##   [2,] 0.2667 0.2667 0.2667 0.06667 0.06667
##   [3,] 0.2667 0.2667 0.2667 0.06667 0.06667
##   [4,] 0.1000 0.1000 0.1000 0.40000 0.40000
##   [5,] 0.1000 0.1000 0.1000 0.40000 0.40000
```

The numbers show that if it's a good year (columns 1-3), the probability of next year being bad is low, whereas if it's a bad year (columns 4-5), the probability of next year being bad is high (see rows 4-5 of `PbearM4`). The projection should show us this is true:

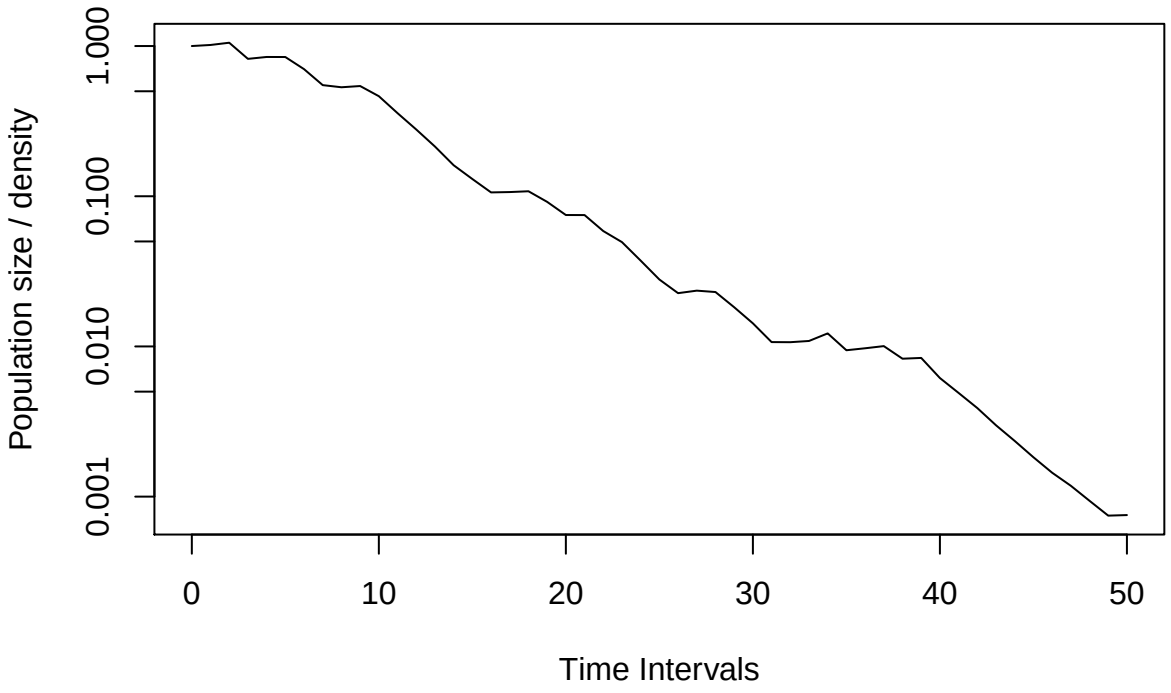
```
Pbearp2.4 <- project(Pbear, Pbearvec, Aseq = PbearM4, time = 50)
plot(Pbearp2.4, log = "y")
```



Generally, it seems that bad years occur together (where the population shrinks), and so do good years (where the population grows).

So far, the overall probability of switching between the states is symmetric: 20% chance of switching from bad to good, 20% chance of switching from good to bad. This needn't be the case: perhaps there are stronger positive feedbacks in bad weather conditions than in good. In the next projection we will increase the probability of switching from good to bad, but keep the probability of switching from bad to good the same:

```
p5 <- 0.5
q5 <- 0.8
( PbearM5 <- matrix(c(rep(c((1-p5)/3, (1-p5)/3, (1-p5)/3, p5/2, p5/2), 3),
                      rep(c((1-q5)/3, (1-q5)/3, (1-q5)/3, q5/2, q5/2), 2)),
                    5, 5) )
##          [,1]   [,2]   [,3]    [,4]    [,5]
##   [1,] 0.1667 0.1667 0.1667 0.06667 0.06667
##   [2,] 0.1667 0.1667 0.1667 0.06667 0.06667
##   [3,] 0.1667 0.1667 0.1667 0.06667 0.06667
##   [4,] 0.2500 0.2500 0.2500 0.40000 0.40000
##   [5,] 0.2500 0.2500 0.2500 0.40000 0.40000

Pbearp2.5 <- project(Pbear, Pbearvec, Aseq = PbearM5, time = 50)
plot(Pbearp2.5, log = "y")
```



In this case, whilst there are still long strings of bad years, the strings of good years are much shorter.

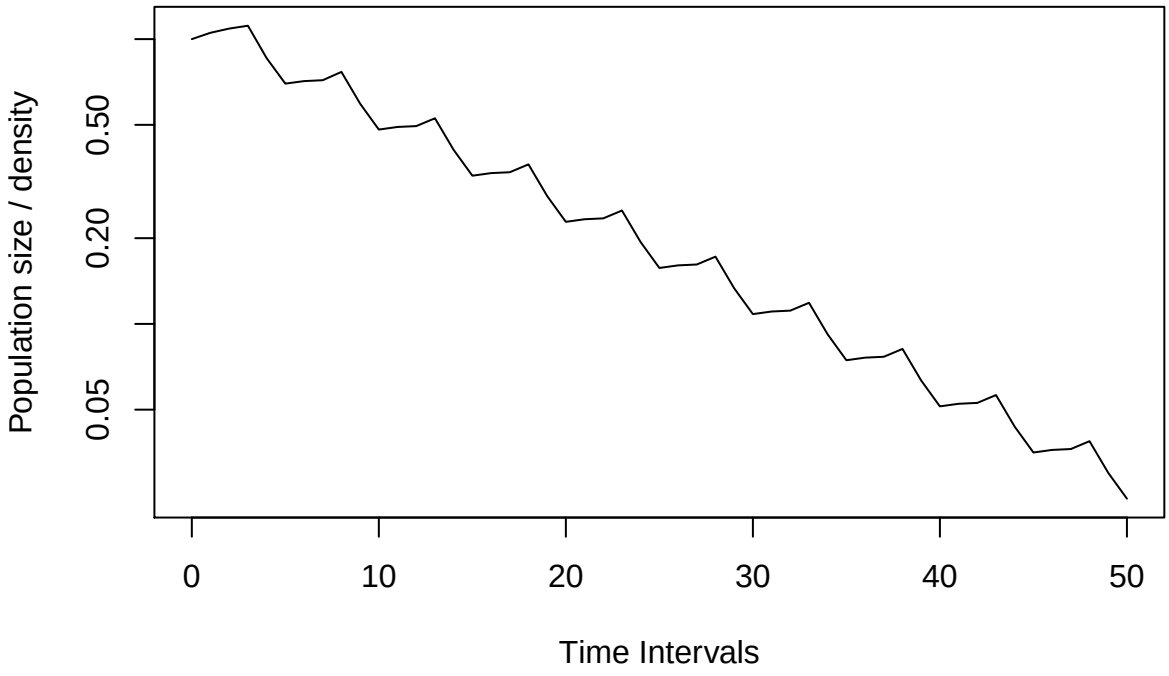
2.2 Nonrandom matrix selection

The final option for matrix selection is to give exactly the sequence of matrices to use. It's possible to force `project` to use a specific sequence of matrices, simply by passing a sequence of numbers or names to `Aseq`. These numbers or names must correspond to the elements of your list of matrices.

(so for the final projection of the polar bear model we could have instead specified c(1, 2, 3, 4, 5, 1, 2, 3,...) or c("2001", "2002", "2003", "2004", "2005", "2001", "2002", "2003",...)).

Imagine that the weather conditions are cyclic over 5-year cycles, and that the different conditions occur in the order in which they were observed. In this case, there's a 100% chance of moving from 2001 conditions to 2002 conditions, the same for 2002 to 2003, and so on, until the 5-year cycle completes and conditions move from 2005 conditions back to 2001 again, so the matrix sequence is 2001, 2002, 2003, 2004, 2005, 2001, 2002, 2003,... and so on. We can specify this using the matrix numbers (in the order they appear in the `Pbear`) list, or their names in that list:

```
( Pbearseq <- rep(1:5, 10) )
##    [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
##   [36] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

Pbearp3.1 <- project(Pbear, Pbearvec, Aseq = Pbearseq)
plot(Pbearp3.1, log = "y")
```



You would get the same result using `Aseq = rep(c("Y2001", "Y2002", "Y2003", "Y2004", "Y2005"), 10)`. The projection shows there's still a decline in overall population size, but with a clear cyclic dynamic to the trend.

ii. EXTRAS...
The type of population projection we've used is called ***matrix selection****: at each timestep a matrix is selected at random from a list. These are usually parameterised by collecting data under multiple different environmental conditions (these are often replications through time, but may also be replications through space or different experimental treatments).*

Another common form of stochastic projection is using ***element selection****. In this case, one or more of the matrix elements are selected at random from a statistical distribution. Projections using element selection are possible in* ***popdemo*** *by generating a list of matrices using element selection procedures and passing all of these to* ***project****. In this example, we'll use a matrix for the thistle Carlina vulgaris*[2]*. There are 3 stages based on size and reproductive status:*
small rosette
large rosette
large flowering rosette

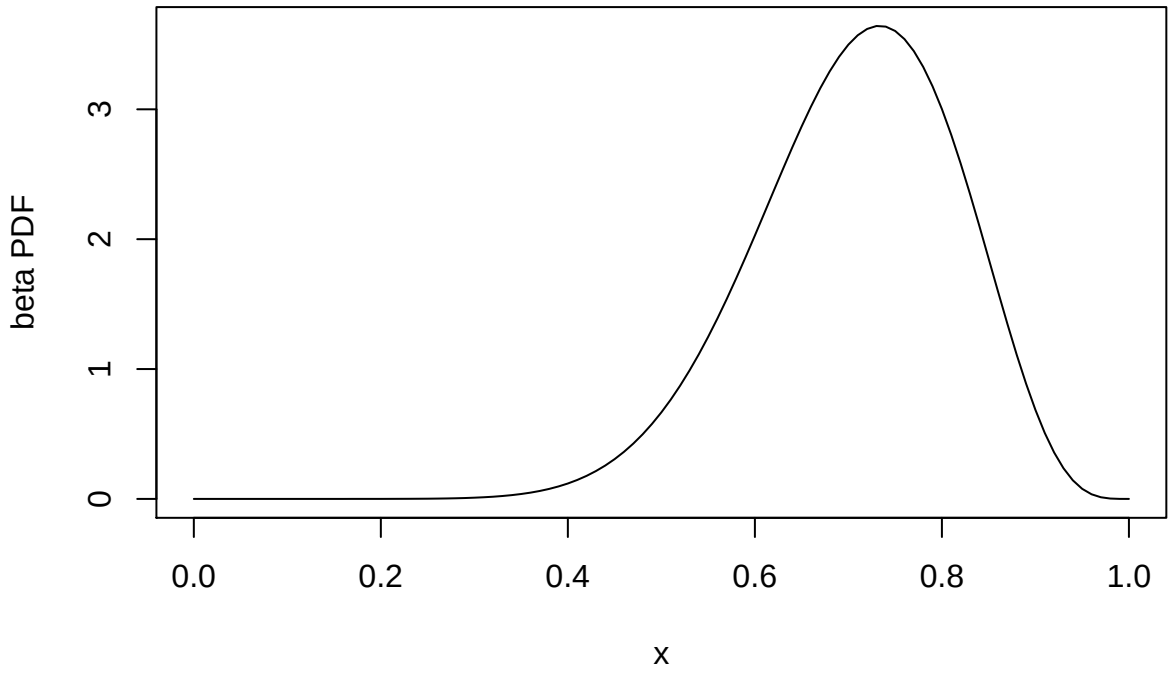
```
( Thistle <- Matlab2R("[0.5 0 2.8; 0.25 0.222 0; 0 0.667 0]") )
##        [,1]  [,2] [,3]
##   [1,] 0.50 0.000  2.8
##   [2,] 0.25 0.222  0.0
##   [3,] 0.00 0.667  0.0

#we'll use a random vector for the models
Thistlevec <- runif(3); Thistlevec <- Thistlevec / sum(Thistlevec)
```

In any given year, 88.9% of stage 2 individuals survive, with 75% of these survivors flowering and 25% not flowering. At the moment these parameters are fixed, but they don't have to be. A beta distribution with $\alpha =$

[2] Lï¿½fgren et al. (2000) Ann. Bot. Fennici, 37, 183-192.

12 and $\beta = 5$ would puts some variability around this 75% flowering probability:

```
x<- seq(0, 1, 0.01)
plot(x, dbeta(x, shape1 = 12, shape2 = 5), type = "l", ylab = "beta PDF")
```



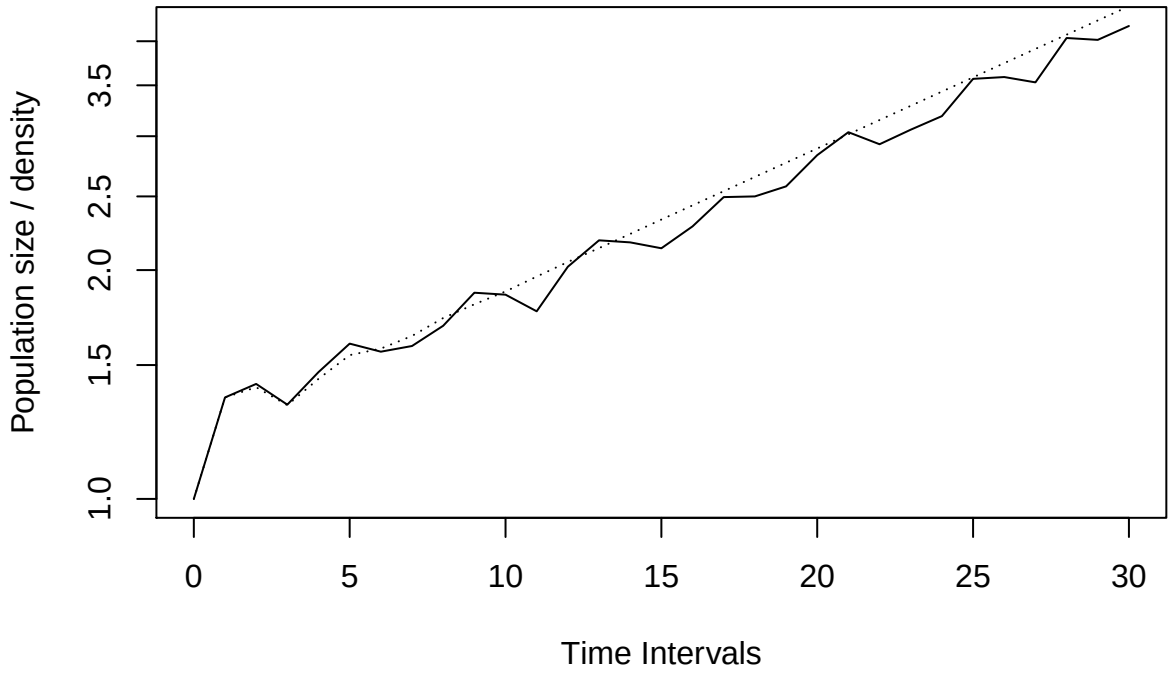
In reality it would be necessary to fit statistical models to data to quantify these sorts of probabilities, but this example illustrates the method.

If we want to create a set of 30 matrices, using this beta distribution to allow probability of flowering to vary, we can:
- replicate the projection matrix 30 times in a list
- generate 30 random beta draws using the parameters above, which describe the random probability of flowering in each year; let's call this p(flower)
- replace the [3,2] element of each of the 30 matrices with 0.889 x p(flower)
- replace the [2,2] element of each of the 30 matrices with 0.889 x (1 - p(flower))

```
#create list of 30 matrices
ThistleS <- rep(list(Thistle), 30)
#generate beta-distributed random probability of flowering
times <- 30
pflwr <- rbeta(times, shape1 = 12, shape2 = 5)
#replace [3,2] element of every matrix with new random number
a32 <- 0.889*pflwr
a22 <- 0.889*(1-pflwr)
ThistleS <- mapply(function(A, r){A[3,2] <- r; A}, ThistleS, a32, SIMPLIFY = FALSE)
#replace [3,2] element of every matrix with new random number
ThistleS <- mapply(function(A, r){A[2,2] <- r; A}, ThistleS, a22, SIMPLIFY = FALSE)
```

As we saw earlier, it's possible to force `project` to use a specific sequence of matrices, simply by passing a sequence of numbers or names to `Aseq`. For this exercise in element selection, all we need to do is pass `1:30` to `Aseq` and it selects the random matrices in order:

```
Thistlep1.1 <- project(ThistleS, vector = Thistlevec, Aseq = 1:30)
plot(Thistlep1.1, log = "y")
lines(0:30, project(Thistle, Thistlevec, time = 30), lty = 3)
```



The dotted line shows the deterministc projection for comparison. The stochastic projection should look similar, but with variation in the timestep-by-timestep growth.

The underlying biology should drive decisions about how to sample transitions in element selection. For example:
- Something like a Gamma distribution could be used to model fecundity transitions, as these should be truncated at 0 and $+\infty$
- More than one distribution can model a certain type of transition; for example, probabilities (such as flowering, survival, etc.) can be drawn from beta distributions, but other possibilities exist, e.g. truncated normal.
- Transitions are composed of underlying vital rates and they may correlate with one another, so it may be important to model stochasticity in underlying vital rates and contruct matrices from these. The example here is a simple version of this: two elements correlate perfectly negatively with one another. However, if the underlying elements are calculated from other distributions such as survival or growth kernels, or different life cycle transitions depend on things such as resource allocation trade-offs, the correlations can get much more complex.

---

3. Stochastic dynamics

Stochastic projections are a random process. Perhaps the best way to understand stochastic population growth is to project the model and calculate growth from the results. We can calculate long-term growth of a stochastic model by averaging per-timestep growth rates over a long projection time. Every stochastic projection has a mean growth and a variance in growth. Let's calculate stochastic dynamics for the five polar bear examples from the Markov processes section above:

```
stoch(Pbear, c("lambda", "var"), vector = Pbearvec, Aseq = PbearM1,
      iterations = 3000, discard = 100)
##     lambda     var
##   1 0.9368 0.01655

stoch(Pbear, c("lambda", "var"), vector = Pbearvec, Aseq = PbearM2,
      iterations = 3000, discard = 100)
##     lambda     var
##   1  0.912 0.01682

stoch(Pbear, c("lambda", "var"), vector = Pbearvec, Aseq = PbearM3,
      iterations = 3000, discard = 100)
##     lambda      var
##   1 0.8297 0.009498

stoch(Pbear, c("lambda", "var"), vector = Pbearvec, Aseq = PbearM4,
      iterations = 3000, discard = 100)
##     lambda   var
##   1 0.9074 0.018

stoch(Pbear, c("lambda", "var"), vector = Pbearvec, Aseq = PbearM5,
      iterations = 3000, discard = 100)
##     lambda     var
##   1 0.8561 0.01367
```

Look carefully at these outcomes. As predicted, the mean growth goes down the higher the probability of bad years, decreasing from projection 1 to 2 to 3.

Projections 4 has almost the same growth rate as projection 2, and there's a good reason for this. It's not as intuitive to understand the overall probability of choosing a matrix for a Markov process like the one in model 4, but it is easy to calculate: it's the dominant right eigenvector of the matrix (scaled to sum to 1). As such, we can calculate it easily using the `eigs` function in `popdemo`:

```
eigs(PbearM2, "ss")
##   [1] 0.1667 0.1667 0.1667 0.2500 0.2500

eigs(PbearM4, "ss")
##   [1] 0.1667 0.1667 0.1667 0.2500 0.2500
```

In these vectors, each number gives the probability of choosing a matrix at random (in order from 2001-2005). For model 2, the probabilities are the same as the columns in the matrix. For model 4, it works out that there's an equal and symmetric (but low) probability of switching between "bad" and "good" states (20% change of switching from good to bad, 20% chance of switching from bad to good), and therefore equal (but high) probability of staying within the states (80% chance of each). The long-run probabilities of choosing each matrix are the same as for model 2 because on average each matrix gets picked 50% of the time. This observation stresses the need for long projections to calculate stochastic growth. In the above examples we used 3000 `iterations`, `discard`ing the first 100 to eliminate effects from the initial conditions. Conversely, too many iterations could result in numbers becoming too large or too small for `R` to continue to be able to calculate them.

It's even more difficult to understand long-run probabilities in asymmetric situations such as model 5, but we can still use the dominant eigenvector:

```
eigs(PbearM5, "ss")
##   [1] 0.09524 0.09524 0.09524 0.35714 0.35714
```

This explains why population growth in model 5 is so low. Overall, more than 71% of years will be "bad" years.