

Loading projection matrices into R

Chris Stubben

July 11, 2011

This guide describes how to load projection matrices into R using a number of examples from published demographic studies. At least for small matrices, one option is to combine matrix elements into a comma-separated list of values using `c()` and then use `matrix` to reshape the vector into a square matrix.

```
R> A <- c(0, 0.3, 0, 1, 0, 0.5, 5, 0, 0)
R> A <- matrix(A, nrow = 3)
```

The following examples expand on this simple case by using two methods, either copying and pasting a matrix using `scan` or reading a matrix into R using `read.table` and related functions.

Loading matrices using scan

If a projection matrix is part of a larger PDF or HTML document, you can often copy and paste the matrix elements directly into the R console after typing the `scan` command. In this first example, the mean matrix for *Centaurea corymbosa* was copied from the first row in Table 1 from Freville et al (2004) and pasted below.

```
R> ceco<-scan()
0 0 5.905 0.368 0.639 0.025 0.001 0.152 0.051
```

Be sure to enter a blank line to terminate the input on the screen. At the R terminal, your screen should look something like this if done successfully.

```
R> ceco<-scan()
1: 0 0 5.905 0.368 0.639 0.025 0.001 0.152 0.051
10:
Read 9 items
```

Next, create a vector of stages to assign to the row and column names (i.e., the dimension names) and then use the `matrix` function to reshape the vector by rows into a 3×3 matrix. By default, a matrix is filled by columns, but usually values are copied by rows, so the `byrow` option should be set to `TRUE`.

```
R> stages <- c("seedling", "vegetative", "flowering")
R> matrix(ceco, nrow = 3, byrow = TRUE, dimnames = list(stages,
  stages))
```

	seedling	vegetative	flowering
seedling	0.000	0.000	5.905
vegetative	0.368	0.639	0.025
flowering	0.001	0.152	0.051

One of the nice features of R is the ability to create new functions. If the `matrix` command above is too tedious, you can add your own function to simplify the creation of square matrices. The `matrix2` command is also included in the latest version of `popbio` 2.3.

```
R> matrix2 <- function(x, stages, byrow = TRUE) {
  matrix(x, nrow = sqrt(length(x)), byrow = byrow,
    dimnames = list(stages, stages))
}
R> ceco <- matrix2(ceco, stages)
```

One final step after copying and pasting a matrix into R is to save the matrix to a file for future analyses (and to avoid copying and pasting again). One option is to save a binary R data file using `save` and then use `load` to reload the matrix object in the future. Another alternative is to write the matrix to a text file using `write.table` and then use `read.table` to read the file back into R. Since this function always reads a file to a `data.frame`, use `as.matrix` to convert to a matrix (matrix multiplication and a few other functions will not work on data.frames). More details about `read.table` are found in the second section of the guide.

```
R> write.table(ceco, file = "ceco.txt")
R> ceco <- as.matrix(read.table(file = "ceco.txt"))
```

Scanning matrices with characters

In most cases, a copy of a published matrix will include row names or additional values. In the next example, the projection matrix for *Sarracenia purpurea* at Hawley Bog in Table 1 from Gotelli and Ellison (2006) is pasted below. By default, `scan` will read numeric data, so use the `what` option to specify characters in order to read the stage class names (and elasticities in parentheses).

```
R> x1<-scan(, what="")
Recruit 0.0000 (0) 0.0000 (0) 0.0000 (0) 4.0000 (0)
Juvenile 0.1000 (2) 0.9540 (61) 0.0900 (2) 0.0000 (0)
Non-flowering adult 0.0000 (0) 0.0360 (3) 0.7010 (18) 0.8375 (5)
Flowering adult 0.0000 (0) 0.0000 (0) 0.1802 (6) 0.1610 (1)
```

In R, you can use the `grep` function and a regular expression to list only elements with digits or a decimal point. The matching elements are then converted to a numeric vector and the results are reshaped into a square matrix. In addition, stages (in position 1, 10, 19, 29) are assigned to the row and column names. It is usually a good idea to check the matrix by calculating lambda, elasticities, or other values reported in the original paper. In this case, the elasticities below match the copied values in parentheses above (except for the flowering to recruit transition).

```
R> stages <- x1[c(1, 10, 19, 29)]
R> sapu <- matrix2(as.numeric(grep("[0-9.]+$", x1, value = TRUE)),
  stages)
R> sapu
```

	Recruit	Juvenile	Non-flowering	Flowering
Recruit	0.0	0.000	0.000	4.000
Juvenile	0.1	0.954	0.090	0.000
Non-flowering	0.0	0.036	0.701	0.838
Flowering	0.0	0.000	0.180	0.161

```
R> round(elasticity(sapu) * 100)
```

	Recruit	Juvenile	Non-flowering	Flowering
Recruit	0	0	0	2
Juvenile	2	61	2	0
Non-flowering	0	3	18	5
Flowering	0	0	6	1

Scanning matrices with only non-zero elements

In some cases, matrix elements from multiple sites or years will be listed in a table, and often these tables will only include transitions with non-zero elements. There are a few ways to create a projection matrix from these tables, but I'll focus on using a expression of matrix elements since this technique is useful in many other situations.

First copy the pod-specific matrix elements for killer whales from the appendix in Brault and Caswell (1993).

```
R> x2<-scan(, what="")
pod n G1 G2 G3 P2 P3 P4 F2 F3
J01 22 0.9535 0.0802 0.0414 0.8827 0.9586 0.9752 0.0067 0.1632
K01 20 1.0000 0.0694 0.0418 0.9020 0.9582 0.9855 0.0062 0.1737
L01 63 0.9562 0.0722 0.0406 0.9030 0.9530 0.9798 0.0037 0.0988
A01 15 1.0000 0.0727 0.0485 0.9015 0.9515 0.9667 0.0043 0.1148
```

```

A04 12 0.8165 0.0774 0.0485 0.8903 0.9515 0.9810 0.0042 0.1054
A05 10 1.0000 0.0730 0.0485 0.9123 0.9515 0.9545 0.0027 0.0732
B01 8 1.0000 0.0746 0.0485 0.9254 0.9515 0.9810 0.0025 0.0651
C01 8 1.0000 0.0800 0.0294 0.9200 0.9706 0.9608 0.0047 0.1159
D01 12 1.0000 0.0759 0.0438 0.9241 0.9562 1.0000 0.0068 0.1761
G01 24 1.0000 0.0833 0.0714 0.9167 0.9286 1.0000 0.0061 0.1418
G12 11 1.0000 0.0784 0.0485 0.9216 0.9515 0.9810 0.0050 0.1251
H01 7 1.0000 0.0746 0.0485 0.9254 0.9515 0.9810 0.0021 0.0542
I01 7 1.0000 0.0714 0.0485 0.9286 0.9515 0.9810 0.0027 0.0732
I02 7 1.0000 0.0714 0.0485 0.9286 0.9515 1.0000 0.0045 0.1220
I11 15 1.0000 0.0714 0.0485 0.9286 0.9515 0.9810 0.0052 0.1428
I18 13 1.0000 0.0714 0.0485 0.9286 0.9515 0.9810 0.0037 0.0998
I31 7 1.0000 0.0714 0.0485 0.9286 0.9515 0.9810 0.0047 0.1273
R01 20 1.0000 0.0595 0.0485 0.8929 0.9515 1.0000 0.0024 0.0797

```

Second, format the scanned values into a matrix, select only the pod rates (by skipping the first row and first two columns), and add row and column labels.

```

R> x2 <- matrix(x2, nrow = 19, byrow = TRUE)
R> pods <- matrix(as.numeric(x2[-1, -(1:2)]), nrow = 18)
R> dimnames(pods) <- list(x2[-1, 1], x2[1, -(1:2)])
R> head(pods)

```

	G1	G2	G3	P2	P3	P4	F2	F3
J01	0.954	0.0802	0.0414	0.883	0.959	0.975	0.0067	0.1632
K01	1.000	0.0694	0.0418	0.902	0.958	0.986	0.0062	0.1737
L01	0.956	0.0722	0.0406	0.903	0.953	0.980	0.0037	0.0988
A01	1.000	0.0727	0.0485	0.901	0.952	0.967	0.0043	0.1148
A04	0.817	0.0774	0.0485	0.890	0.952	0.981	0.0042	0.1054
A05	1.000	0.0730	0.0485	0.912	0.952	0.955	0.0027	0.0732

Third, enter the projection matrix from Fig 1 in Brault and Caswell (1993) into an expression and include the commands to convert the values into a square matrix with stage class names. The symbols in this expression must match the column names in the matrix of pod-specific rates above.

```

R> podA<-expression(
  matrix2( c(
    0, F2, F3, 0,
    G1, P2, 0, 0,
    0, G2, P3, 0,
    0, 0, G3, P4),
    c("yearling", "juvenile", "mature", "postreprod")))

```

Fourth, evaluate the the R expression using a `list` of matrix elements in the first row to get a projection matrix from a single pod.

```
R> J01 <- eval(podA, as.list(pods[1, ]))
R> J01
```

	yearling	juvenile	mature	postreprod
yearling	0.000	0.0067	0.1632	0.000
juvenile	0.954	0.8827	0.0000	0.000
mature	0.000	0.0802	0.9586	0.000
postreprod	0.000	0.0000	0.0414	0.975

Finally, you can include the previous step in a loop and create projection matrices for all 18 pods. The dotplot in Figure 1 displays the pod-specific population growth rates.

```
R> whales <- vector("list", 18)
R> names(whales) <- rownames(pods)
R> for (i in 1:18) {
  whales[[i]] <- eval(podA, as.list(pods[i, ]))
}
R> print(dotplot(sort(sapply(whales, lambda)), xlab = "Growth rate"))
```

Loading matrices using `read.table`

Perhaps the most common way to load a projection matrix is to read the matrix from a file using `read.table`. In this case, the file may be stored locally or on the web and these files may include one or more projection matrices. The formats of the files will also vary, so it is often necessary to check and modify some of the default `read.table` options.

In the next example, projection matrices for *Silene acaulis* are stored in the ESA Archives (Morris and Doak 2005, see <http://www.esapubs.org/archive/mono/M075/004/default.htm> for details). To download the comma-separated matrix from Campion Crest in 1995, use `read.table` and change the default separator to a comma.

```
R> CC95 <- read.table("http://www.esapubs.org/archive/mono/M075/004/CC95.txt",
  sep = ",")
```

It is important to note again that `read.table` loads the values into a `data.frame`, so you will need to convert this to a matrix, and you will often need to add dimension names as well. Finally, you can print or even plot the projection matrix using `image2` to check the results (Figure 2).

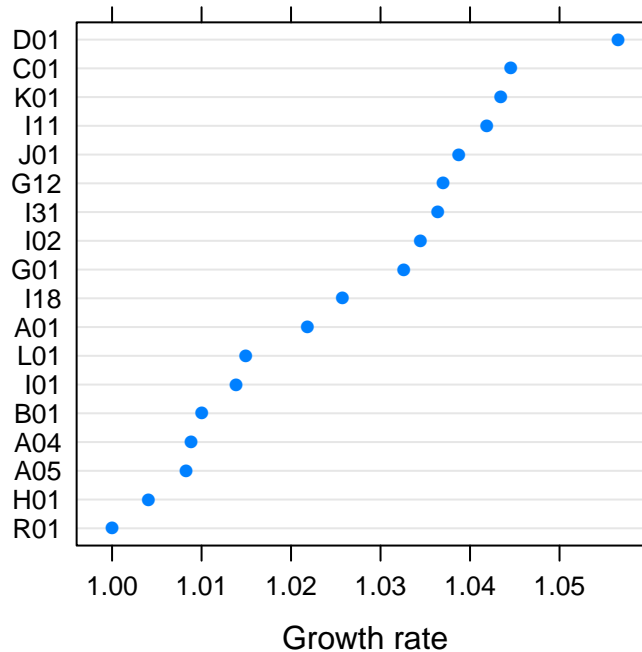


Figure 1: Growth rates for killer whale pods.

```
R> CC95 <- as.matrix(CC95)
R> stages <- c("seed", "sdling", "1R", "5R", "10R", "20R",
  "12cm", "25cm", "50cm", "100cm", "200cm", "200+cm")
R> dimnames(CC95) <- list(stages, stages)
R> image2(CC95, cex = 0.5, log = FALSE)
```

Using a loop, you can also load multiple projection matrices into a list of matrices (there are 25 total matrices from 5 sites, but this example just includes 2 sites below, CC and GU. Additional sites are PA, RG, and RI, but loading PA98.txt will cause an error since the second row has two different field separators). The last command applies the `lambda` function to all matrices in the list and returns the growth rates as a vector.

```
R> years <- 95:99
R> site <- c("CC", "GU")
R> pop <- paste(rep(site, each = 5), years, sep = "")
R> n <- length(pop)
R> silene <- vector("list", n)
R> names(silene) <- pop
R> for (i in 1:n) {
```

	seed	sdling	1R	5R	10R	20R	12cm	25cm	50cm	100cm	200cm	200+cm
seed	0.147	0	0	0	0	0	0	0	0	0.002	0.003	0.005
sdling	0.134	0	0	0	0	0	0	0	0	0.001	0.003	0.004
1R	0	0.632	0.8	0	0	0	0	0	0	0	0	0
5R	0	0	0.1	0.825	0.045	0	0	0	0	0	0	0
10R	0	0	0	0.165	0.855	0.124	0	0	0	0	0	0
20R	0	0	0	0	0.09	0.701	0	0	0	0	0	0
12cm	0	0	0	0	0	0.165	0.775	0.079	0.043	0	0	0
25cm	0	0	0	0	0	0	0.215	0.713	0	0	0	0
50cm	0	0	0	0	0	0	0	0.198	0.904	0	0	0
100cm	0	0	0	0	0	0	0	0	0.043	0.825	0.165	0
200cm	0	0	0	0	0	0	0	0	0	0.165	0.77	0.099
200+cm	0	0	0	0	0	0	0	0	0	0	0.055	0.891

Figure 2: Projection matrix for *Silene acaulis* at Campion Crest in 1995.

```

x <- paste("http://www.esapubs.org/archive/mono/M075/004/",
  pop[i], ".txt", sep = "")
y <- as.matrix(read.table(x, sep = ","))
dimnames(y) <- list(stages, stages)
silene[[i]] <- y
}
R> sapply(silene, lambda)

```

```

CC95 CC96 CC97 CC98 CC99 GU95 GU96 GU97 GU98 GU99
0.991 1.006 1.000 1.006 0.999 1.000 0.996 0.998 1.000 1.004

```

Loading matrices with HTML markup

Many population matrices that are stored in web archives include HTML formatting. In these cases, you can often use `readLines` to download the web page and then use a combination of `grep` to find matrix elements and `gsub` to remove html tags. For example, the projection matrices from *Ardisia elliptica* are also stored in the ESA archives (Koop and Horvitz 2005, see <http://www.esapubs.org/archive/ecol/E086/142/appendix-E.htm>).

	SD	SG	SJ	MJ	LJ	PR	SA	LA
SD	0	0	0	0	0	2.777	22.543	170
SG	0.099	0.364	0.038	0.027	0	0	0	0
SJ	0	0.341	0.759	0.082	0.069	0.023	0	0
MJ	0	0	0.09	0.685	0.052	0	0	0
LJ	0	0	0	0.096	0.707	0.114	0.056	0
PR	0	0	0	0.014	0.121	0.727	0	0
SA	0	0	0	0	0	0.136	0.875	0.017
LA	0	0	0	0	0	0	0.028	0.978

Figure 3: Projection matrix for *Ardisia elliptica* at Forest Edge in 1999.

Use `readLines` to download the entire page including the HTML markup. Next, use the `grep` function to find 832 lines with a matrix element between the html tags (since there are 13 matrices with 64 elements each).

```
R> arel <- readLines("http://www.esapubs.org/archive/ecol/E086/142/appendix-E.htm")
R> y <- grep(">[0-9.]+<", arel)
R> length(y)
```

```
[1] 832
```

Use `gsub` to return just the second parenthesized subexpression (the number between the tags) for those 832 lines and convert the output to a numeric matrix with 13 rows, one row for each projection matrix. To create a single projection matrix, just select a row and reshape it as an 8×8 matrix (Figure 3).

```
R> x <- gsub("(.*>)([0-9.]+)(<.*)", "\\2", arel[y])
R> x <- matrix(as.numeric(x), nrow = 13, byrow = TRUE)
R> stages = c("SD", "SG", "SJ", "MJ", "LJ", "PR", "SA",
              "LA")
R> FE99 <- matrix2(x[1, ], stages)
R> image2(FE99, cex = 0.5)
```

You can also split the 13 matrices and format them into a single list using `split`. The final steps below are use to add the population and year to the list names and to display a vector of growth rates.


```

R> arel <- split(x, 1:13)
R> arel <- lapply(arel, matrix2, stages)
R> years <- c("99", "00", "01")
R> sites <- c("FE", "TF", "HF", "AT")
R> pop <- c(paste(rep(sites, each = 3), years, sep = ""),
            "ST99")
R> names(arel) <- pop
R> sapply(arel, lambda)

FE99 FE00 FE01 TF99 TF00 TF01 HF99 HF00 HF01 AT99 AT00
1.039 0.993 1.002 1.296 1.060 1.133 1.071 1.033 1.098 1.053 1.088
AT01 ST99
0.999 1.304

```

Loading matrices expressed as vital rates

In this final example, a table of vital rates (with html markup) is used to construct matrix models of six *Orchis purpurea* populations from Jacquemyn et al. (2010). The following code downloads and formats the vital rate table.

```

R> url <- "http://www.esapubs.org/archive/ecol/E091/011/appendix-B.htm"
R> vrx <- readLines(url)
R> y <- grep(">[0-9.]+<", vrx, perl = TRUE)
R> y <- gsub("(.*>)([0-9.]+)(<.*)", "\\2", vrx[y], perl = TRUE)
R> vr <- matrix(as.numeric(y), ncol = 15, byrow = TRUE)
R> n <- substr(vrx[seq(35, 665, 18)], 55, 64)
R> rownames(vr) <- gsub(" ", "_", gsub("'", "", n))
R> colnames(vr) <- c("psi", "up", "pi", "ep", "d1", "d2",
                  "d3", "d4", "d5", "d6", "g53", "g54", "g64", "g65",
                  "g56")
R> vr[1:5, 1:5]

      psi    up    pi    ep    d1
S1_02-03 32.4 0.019 6000 0.023 0.012
S1_03-04 34.5 0.033 6000 0.023 0.027
S1_04-05 32.2 0.019 6000 0.023 0.011
S1_05-06 28.2 0.012 6000 0.023 0.000
S1_06-07 33.0 0.030 6000 0.023 0.000

```

Next, format the matrix of vital rates listed in equation 1 into an expression. Finally, evaluate the R expression using a list of vital rates from a single site to return a projection matrix. You can also create a list of all 30 matrices following the code in the earlier killer whale example.

```

R> orpu<-expression( matrix2( c(
  0, 0, 0, 0, 0, psi*up*pi*ep,
  d1, 0, 0, 0, 0, 0,
  0, d2, 0, 0, 0, 0,
  0, 0, d3*(1-g53), d4*(1-g54-g64), 0, 0,
  0, 0, d3*g53, d4*g54, d5*(1-g65), d6*g56,
  0, 0, 0, d4*g64, d5*g65, d6*(1-g56)
), c("pcorm", "tuber", "sdlng", "juv", "nonfl", "flower"))
)
R> s1<-eval(orpu, as.list(vr[1,]) )
R> s1

```

	pcorm	tuber	sdlng	juv	nonfl	flower
pcorm	0.000	0.000	0	0.00	0.000	84.887
tuber	0.012	0.000	0	0.00	0.000	0.000
sdlng	0.000	0.041	0	0.00	0.000	0.000
juv	0.000	0.000	1	0.75	0.000	0.000
nonfl	0.000	0.000	0	0.25	0.615	0.625
flower	0.000	0.000	0	0.00	0.385	0.375

References

- Brault, S., and H. Caswell. 1993. Pod-Specific Demography of Killer Whales (*Orcinus Orca*). *Ecology* 74:1444-1454.
- Freville, H., B. Colas, M. Riba, H. Caswell, A. Mignot, E. Imbert, I. Olivieri. 2004. Spatial and temporal demographic variability in the endemic plant species *Centaurea corymbosa* (Asteraceae). *Ecology* 85: 694-703.
- Gotelli, N.J. and A.M. Ellison. 2006. Forecasting extinction risk with nonstationary matrix models. *Ecological Applications* 16:51-61.
- Jacquemyn, H., R. Brys, E. Jongejans 2010. Seed limitation restricts population growth in shaded populations of a perennial woodland orchid. *Ecology*. 91:119-129.
- Koop, A.L. and C.C. Horvitz. 2005. Projection matrix analysis of the demography of an invasive, nonnative shrub (*Ardisia elliptica*). *Ecology* 86:2661-2672.
- Morris, W.F. and D.F. Doak. 2005. How general are the determinants of the stochastic population growth rate across nearby sites? *Ecological Monographs* 75:119-137.