Plant Strategies :: Chapter 4 Plant Population Demography

Daniel C. Laughlin

19 May, 2022

Hypothetical short-lived perennial plant Hierba rápido

```r
# read in data from Table 4.1
L <- c(1,0.3,0.1,0.05,0.03,0.01,0.005)
m <- c(0,0,6,7,6,3,0)
x <- c(0:6)

# make Euler-Lotka function
euler <- function(r) sum(L * m * exp(-r * x)) - 1

# solve for r given L, m, and x
res <- uniroot(f = euler, interval = c(-100, 100), tol = 1e-8, extendInt="yes")

# r
res$root
```

```
## [1] 0.05556644
```

```r
# lambda
exp(res$root)
```

```
## [1] 1.057139
```

Make matrix population model from life table of Hierba rápido

```r
# create empty matrix
A <- matrix(0,length(m)-1,length(m)-1)

# fill matrix
A[1,] <- c(m[2]*L[2]/L[1], m[3]*L[3]/L[2], m[4]*L[4]/L[3],
           m[5]*L[5]/L[4], m[6]*L[6]/L[5], m[7]*L[7]/L[6])
A[2,] <- c(L[2]/L[1],0,0,0,0,0)
A[3,] <- c(0,L[3]/L[2],0,0,0,0)
A[4,] <- c(0,0,L[4]/L[3],0,0,0)
A[5,] <- c(0,0,0,L[5]/L[4],0,0)
A[6,] <- c(0,0,0,0,L[6]/L[5],0)

# compute lambda using eigenanalysis on matrix
# this value is the same as the Euler-Lotka equation estimate
eigen(A)$values[1]
```

```
## [1] 1.057139+0i
```

```
matU <- A
matU[1,] <- c(0,0,0,0,0,0)

matF <- A
matF[2:6,] <- c(0,0,0,0,0,0)
```

Calculate life history traits from matrix model of Hierba rápido

```
# separate A into component matrices U (growth/survival) and F (fecundity)
matU <- A
matU[1,] <- c(0,0,0,0,0,0)
matF <- A
matF[2:6,] <- c(0,0,0,0,0,0)

# compute Ro net reproductive rate
Ro <- sum(L*m)  # same as Rage::net_repro_rate(matU, matF)

# compute generation time
GenT <- log(Ro)/log(exp(res$root)) # same as Rage::gen_time(matU, matF)

# compute Keyfitz's entropy
H <- sum(-log(L)*L)/sum(L) # same as Rage::entropy_k(L)

# compute degree of iteroparity
entropy_d(L,m)
```

```
## [1] 0.9377531
```

```
# compute r from net reproductive rate  and generation time
r <- log(Ro)/GenT

# compute age at maturity
mature_age(matU, matF)
```

```
## [1] 2
```

```
# compute mean life expectancy
life_expect_mean(matU)
```

```
## [1] 1.49
```

```
# compute mature life expectancy
longevity(matU)
```

```
## [1] 5
```

Project population of Hierba rápido

```
# set initial population sizes of each age class
Nx <- c(2,2,2,2,2,2)
12*exp(res$root)^10
```

```
## [1] 20.91719
```

```
project(A, vector=Nx, time=10)
```

```
## Warning in project(A, vector = Nx, time = 10): Matrix is reducible
```

```
## 1 deterministic population projection over 10 time intervals.
##
##  [1] 12.00000 24.33333 15.92667 19.64000 18.98600 21.67833 21.86240 23.66510
##  [9] 24.71154 26.32258 27.69529
```

```
projection1 <- project(A, vector=Nx, time=10, return.vec=TRUE)
```

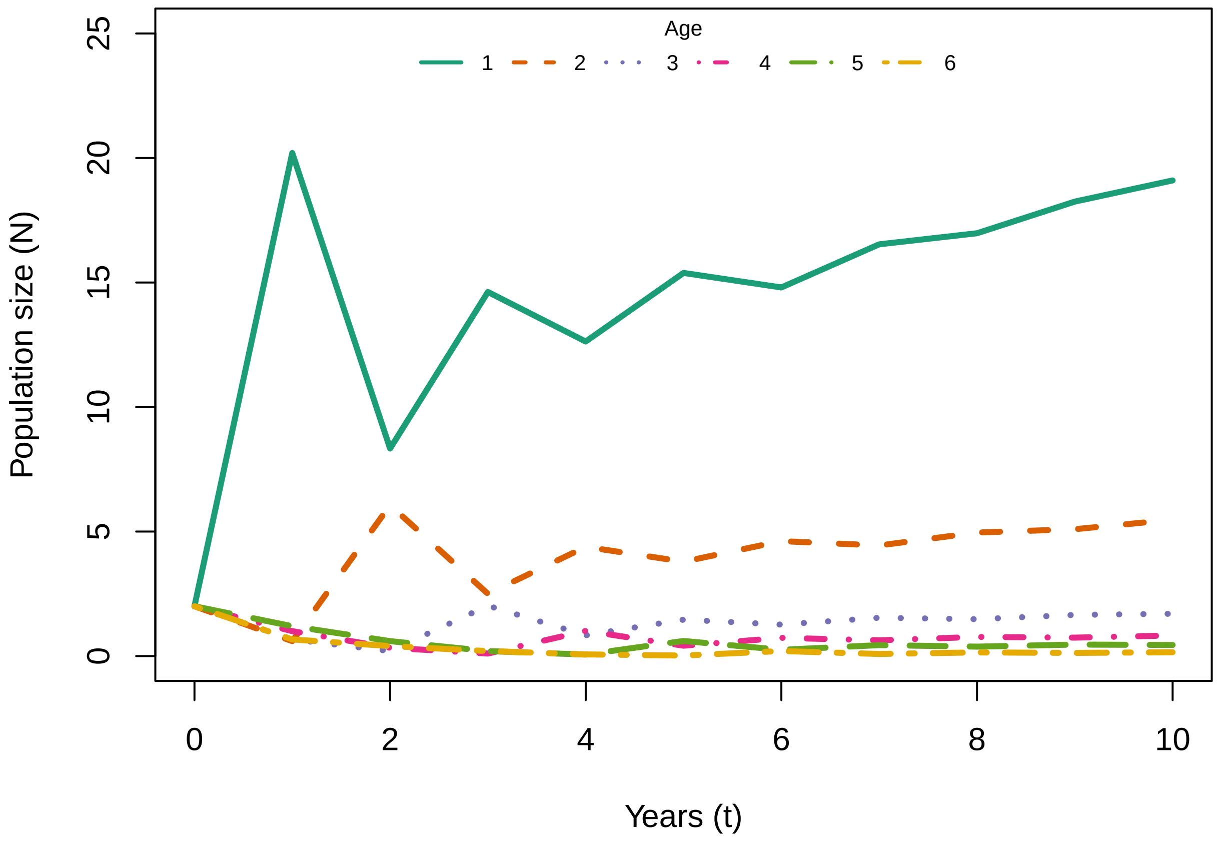
```
## Warning in project(A, vector = Nx, time = 10, return.vec = TRUE): Matrix is
## reducible
```

```
projection1@vec
```

```
##              S1       S2        S3        S4      S5         S6
##  [1,]  2.000000 2.000000 2.0000000 2.0000000 2.00000 2.00000000
##  [2,] 20.200000 0.600000 0.6666667 1.0000000 1.20000 0.66666667
##  [3,]  8.333333 6.060000 0.2000000 0.3333333 0.60000 0.40000000
##  [4,] 14.620000 2.500000 2.0200000 0.1000000 0.20000 0.20000000
##  [5,] 12.630000 4.386000 0.8333333 1.0100000 0.06000 0.06666667
##  [6,] 15.384667 3.789000 1.4620000 0.4166667 0.60600 0.02000000
##  [7,] 14.801000 4.615400 1.2630000 0.7310000 0.25000 0.20200000
##  [8,] 16.532900 4.440300 1.5384667 0.6315000 0.43860 0.08333333
##  [9,] 16.977233 4.959870 1.4801000 0.7692333 0.37890 0.14620000
## [10,] 18.248230 5.093170 1.6532900 0.7400500 0.46154 0.12630000
## [11,] 19.098575 5.474469 1.6977233 0.8266450 0.44403 0.15384667
```

```
cols <- brewer.pal(6, name="Dark2")

par(mar=c(4,4,1,1))
matplot(x=c(0:10), projection1@vec, type="l", col=cols, lwd=3, ylim=c(0,25),
        xlab="Years (t)", ylab="Population size (N)",lty=c(1:6))
legend("top",legend=c("1","2","3","4","5","6"), horiz=TRUE,bty="n",
       lty=c(1:6), lwd=2, cex=0.7, col=cols, title="Age")
```



Download compadre database of matrix population models

```
### Download COMPADRE Rfile from https://compadre-db.org/
### see for more info: https://jonesor.github.io/Rcompadre/articles/GettingStarted.html
load("COMPADRE_v.6.21.8.0.RData")
Compadre <- as_cdb(compadre)
```

Load Echinacea angustifolia matrix (#37 from Hurlburt's thesis, included in compadre)

```
cdb_check_species(Compadre, "Echinacea angustifolia")
```

```
##                  species in_db
## 1 Echinacea angustifolia  TRUE
```

```
echang <- cdb_check_species(Compadre, "Echinacea angustifolia", return_db = TRUE)

# define A, U, and F submatrices
matA <- matA(echang)[[37]]
matU <- matU(echang)[[37]]
matF <- matF(echang)[[37]]

# name stages
```

```
classInfo <- matrixClass(echang)[[37]]
matA <- name_stages(matA, c("Seedling","Small","Medium","Large","Dormant"))
```

```
## Warning in name_stages(matA, c("Seedling", "Small", "Medium", "Large", "Dormant")): Naming `prefix` 
## Warning in name_stages(matA, c("Seedling", "Small", "Medium", "Large", "Dormant")): Existing stage na
```

```
matU <- name_stages(matU, c("Seedling","Small","Medium","Large","Dormant"))
```

```
## Warning in name_stages(matU, c("Seedling", "Small", "Medium", "Large", "Dormant")): Naming `prefix` 
## Warning in name_stages(matU, c("Seedling", "Small", "Medium", "Large", "Dormant")): Existing stage na
```

```
matF <- name_stages(matF, c("Seedling","Small","Medium","Large","Dormant"))
```

```
## Warning in name_stages(matF, c("Seedling", "Small", "Medium", "Large", "Dormant")): Naming `prefix` 
## Warning in name_stages(matF, c("Seedling", "Small", "Medium", "Large", "Dormant")): Existing stage na
```

```
# make life table from matrix model
lifeT <- mpm_to_table(matU, matF)

# compute Keyfitz's entropy
entropy_k(lifeT$lx)
```

```
## [1] 1.5579
```

```
# compute degree of iteroparity
entropy_d(lifeT$lx,lifeT$mx)
```

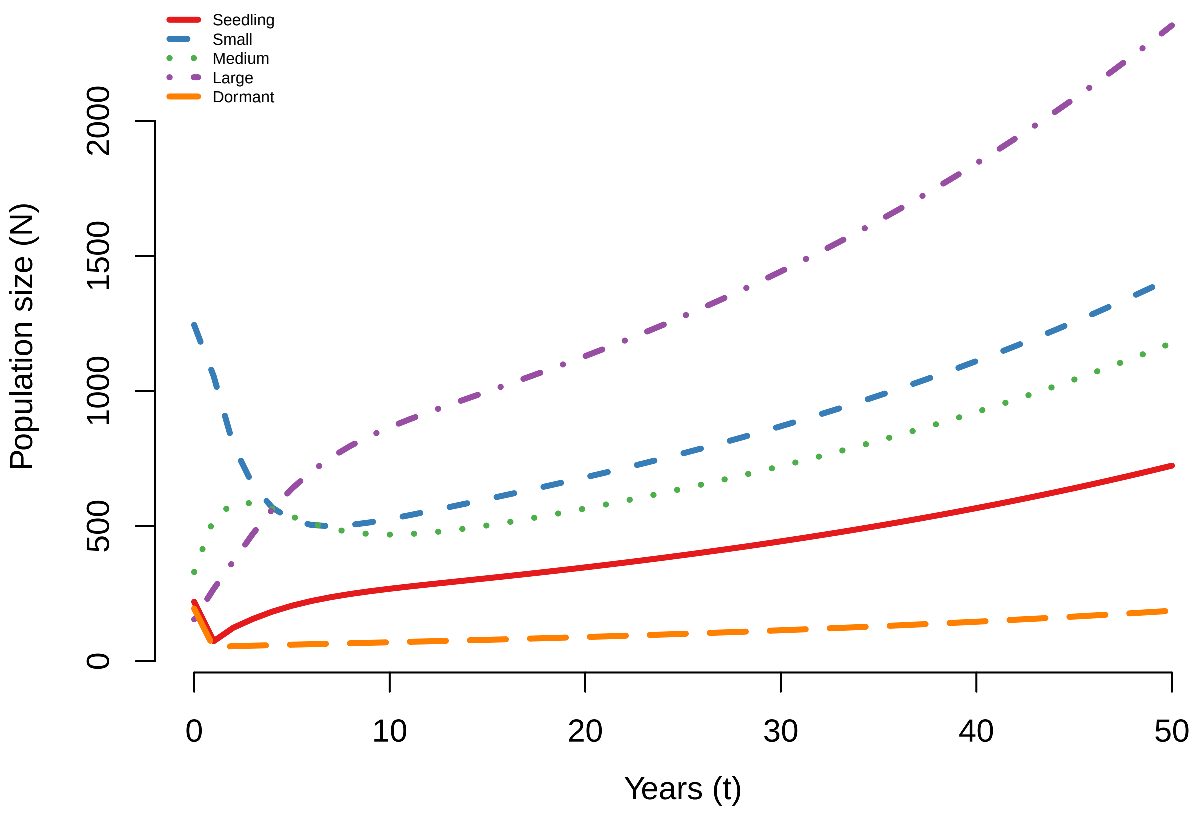
```
## [1] 3.613414
```

```
## Figure 4.4
nt <- c(220, 1245, 330, 155, 195)
projection <- project(matA, nt, time=50, return.vec=TRUE)
projection@vec
```

```
##       Seedling     Small   Medium     Large   Dormant
##  [1,] 220.0000 1245.0000 330.0000  155.0000 195.00000
##  [2,]  74.1575 1055.2018 527.7918  267.6116  50.41000
##  [3,] 123.8282  800.3079 584.2387  369.5989  55.91278
##  [4,] 156.5136  652.7476 585.1527  471.5362  57.62219
##  [5,] 183.6187  568.0894 563.2041  562.2761  59.44054
##  [6,] 205.4644  523.6334 535.4648  638.9790  61.24796
##  [7,] 223.0098  504.1697 510.1817  702.2911  63.03266
##  [8,] 237.2529  499.8433 490.6689  754.3076  64.79727
##  [9,] 249.0811  504.4207 477.6560  797.4848  66.55108
## [10,] 259.2185  514.0629 470.6498  834.1185  68.30519
## [11,] 268.2240  526.4777 468.6887  866.1366  70.07020
## [12,] 276.5122  540.3482 470.7328  895.0569  71.85525
## [13,] 284.3811  554.9558 475.8443  922.0227  73.66780
## [14,] 292.0394  569.9380 483.2533  947.8661  75.51373
## [15,] 299.6307  585.1364 492.3645  973.1771  77.39766
## [16,] 307.2515  600.5030 502.7375  998.3651  79.32320
## [17,] 314.9662  616.0461 514.0576 1023.7093  81.29318
## [18,] 322.8172  631.7985 526.1075 1049.3977  83.30994
## [19,] 330.8327  647.8005 538.7416 1075.5564  85.37543
## [20,] 339.0313  664.0921 551.8656 1102.2698  87.49136
## [21,] 347.4260  680.7092 565.4211 1129.5957  89.65930
## [22,] 356.0263  697.6825  579.3741 1157.5749  91.88074
## [23,] 364.8396  715.0380  593.7070 1186.2379  94.15711
## [24,] 373.8722  732.7975  608.4128 1215.6090  96.48983
## [25,] 383.1298  750.9801  623.4911 1245.7094  98.88034
## [26,] 392.6180  769.6021  638.9459 1276.5585 101.33008
## [27,] 402.3423  788.6786  654.7840 1308.1749 103.84053
## [28,] 412.3084  808.2236  671.0136 1340.5774 106.41318
## [29,] 422.5221  828.2503  687.6440 1373.7850 109.04959
## [30,] 432.9893  848.7717  704.6851 1407.8171 111.75132
## [31,] 443.7162  869.8008  722.1473 1442.6937 114.52000
## [32,] 454.7091  891.3505  740.0412 1478.4352 117.35728
## [33,] 465.9745  913.4340  758.3777 1515.0630 120.26486
## [34,] 477.5191  936.0646  777.1680 1552.5987 123.24448
## [35,] 489.3497  959.2558  796.4234 1591.0646 126.29792
## [36,] 501.4736  983.0217  816.1557 1630.4838 129.42701
## [37,] 513.8978 1007.3764  836.3767 1670.8797 132.63363
## [38,] 526.6299 1032.3345  857.0985 1712.2765 135.91970
## [39,] 539.6774 1057.9111  878.3337 1754.6990 139.28718
## [40,] 553.0482 1084.1213  900.0949 1798.1725 142.73809
## [41,] 566.7502 1110.9809  922.3953 1842.7232 146.27450
## [42,] 580.7918 1138.5060  945.2481 1888.3776 149.89853
## [43,] 595.1812 1166.7131  968.6671 1935.1631 153.61234
## [44,] 609.9271 1195.6190  992.6663 1983.1078 157.41816
## [45,] 625.0384 1225.2411 1017.2602 2032.2403 161.31828
## [46,] 640.5240 1255.5970 1042.4633 2082.5901 165.31502
## [47,] 656.3933 1286.7051 1068.2909 2134.1873 169.41079
## [48,] 672.6558 1318.5839 1094.7583 2187.0629 173.60803
## [49,] 689.3212 1351.2525 1121.8815 2241.2485 177.90926
## [50,] 706.3995 1384.7304 1149.6767 2296.7766 182.31705
## [51,] 723.9009 1419.0378 1178.1605 2353.6805 186.83405
```

```
cols <- brewer.pal(5, name="Set1")

par(mar=c(4,4,1,1))
matplot(x=c(0:50), projection@vec, type="l", col=cols, lwd=3,
        xlab="", ylab="",lty=c(1:5), bty="n")
title(xlab="Years (t)", ylab="", line=2.5)
title(ylab="Population size (N)")
legend("topleft",legend=c("Seedling","Small","Medium","Large","Dormant"),
       lty=c(1:5), lwd=3, col=cols, cex=0.5, bty="n")
```



Compute stable age distribution and reproductive value

```
# use right eigenvalue to compute stable stage distribution manually
stable <- eigen(matA)$vectors[,1]/sum(eigen(matA)$vectors[,1])

# use stable.stage() function to do the same
stable.stage(matA)
```

```
##   Seedling      Small     Medium      Large    Dormant
## 0.12349857 0.24208996 0.20099594 0.40154137 0.03187417
```

```
# use left eigenvalue to compute reproductive value manually
eL <- (eigen(t(matA))$vectors[,1]/sum(eigen(t(matA))$vectors[,1]))
repro <- Re(eL/eL[1])

# use reproductive.value() function to do the same
reproductive.value(matA)
```

```
## Seedling    Small   Medium    Large  Dormant
## 1.000000 1.887536 3.101482 3.530011 2.705694
```

```
### compute life history traits
# net reproductive rate
net_repro_rate(matU, matF)
```

```
## [1] 1.903635
```

```
# generation time
gen_time(matU, matF)
```

```
## [1] 26.30451
```

```
# age at maturity
mature_age(matU, matF)
```

```
## Seedling
## 5.383791
```

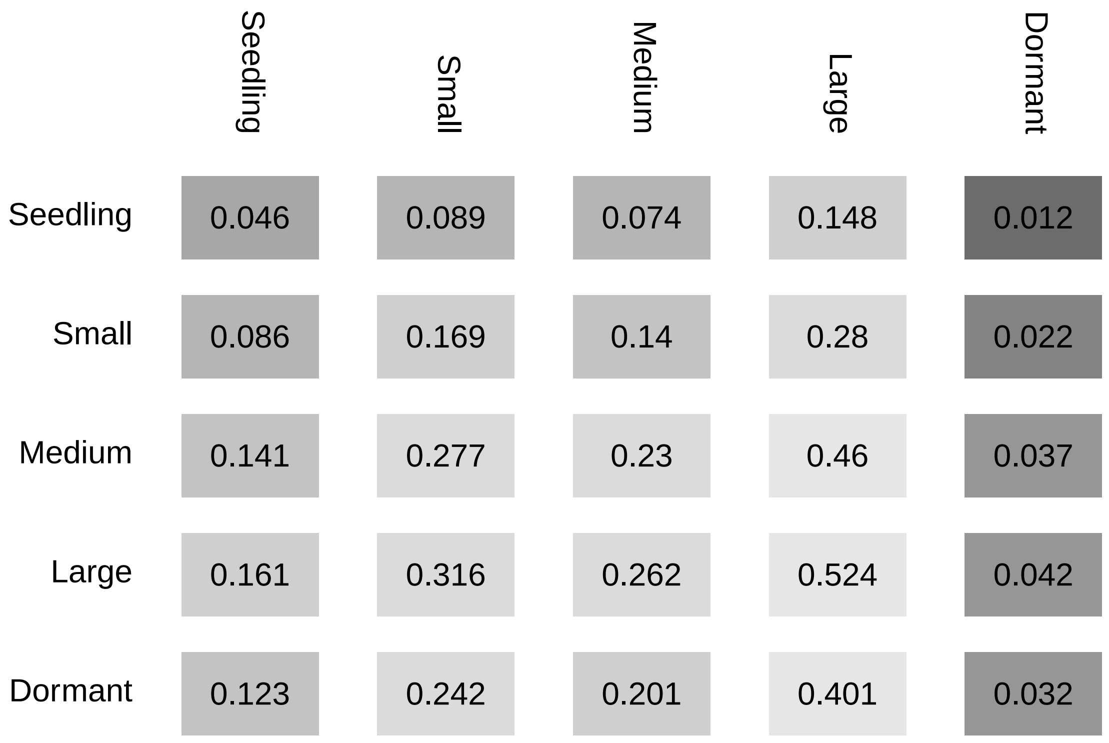
```
# longevity
longevity(matU) # mature life expectancy
```

```
## [1] 99
```

Sensitivity analysis

```
# compute sensitivities manually
num <- (reproductive.value(matA)%*%t(stable.stage(matA)))
den <- as.numeric(reproductive.value(matA)%*%stable.stage(matA))
sens <- (1/den)*num # solve by hand

# use sensitivity() function
sens <- sensitivity(matA)
image2(sens, mar=c(1,6,6,1), box.offset=.3, col=grey.colors(10))
```



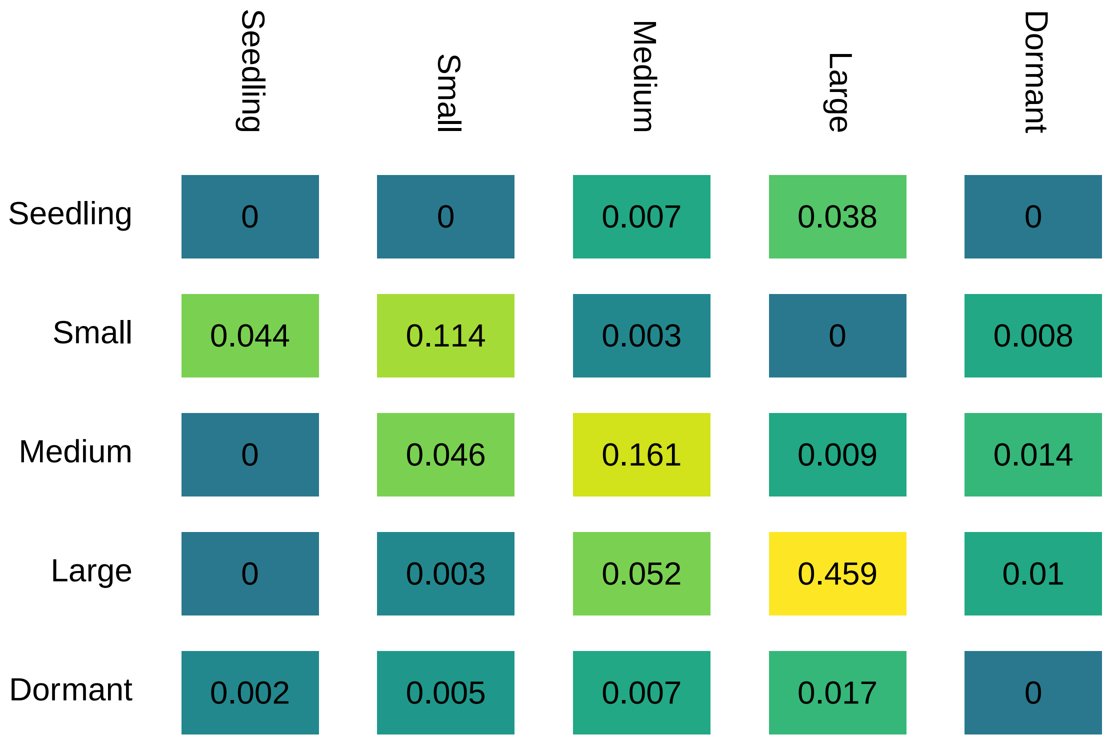
	Seedling	Small	Medium	Large	Dormant
Seedling	0.046	0.089	0.074	0.148	0.012
Small	0.086	0.169	0.14	0.28	0.022
Medium	0.141	0.277	0.23	0.46	0.037
Large	0.161	0.316	0.262	0.524	0.042
Dormant	0.123	0.242	0.201	0.401	0.032

Elasticity analysis

```
# compute elasticities manually
elas <- (matA/lambda(matA))*sens

# use elasticity() function
elas <- elasticity(matA)

### Elasticities
image2(elas, mar=c(1,6,6,1), box.offset=.3, col=viridis_pal(1, begin=0.4, end=1)(10))
```



	Seedling	Small	Medium	Large	Dormant
Seedling	0	0	0.007	0.038	0
Small	0.044	0.114	0.003	0	0.008
Medium	0	0.046	0.161	0.009	0.014
Large	0	0.003	0.052	0.459	0.01
Dormant	0.002	0.005	0.007	0.017	0

Demographic triangles

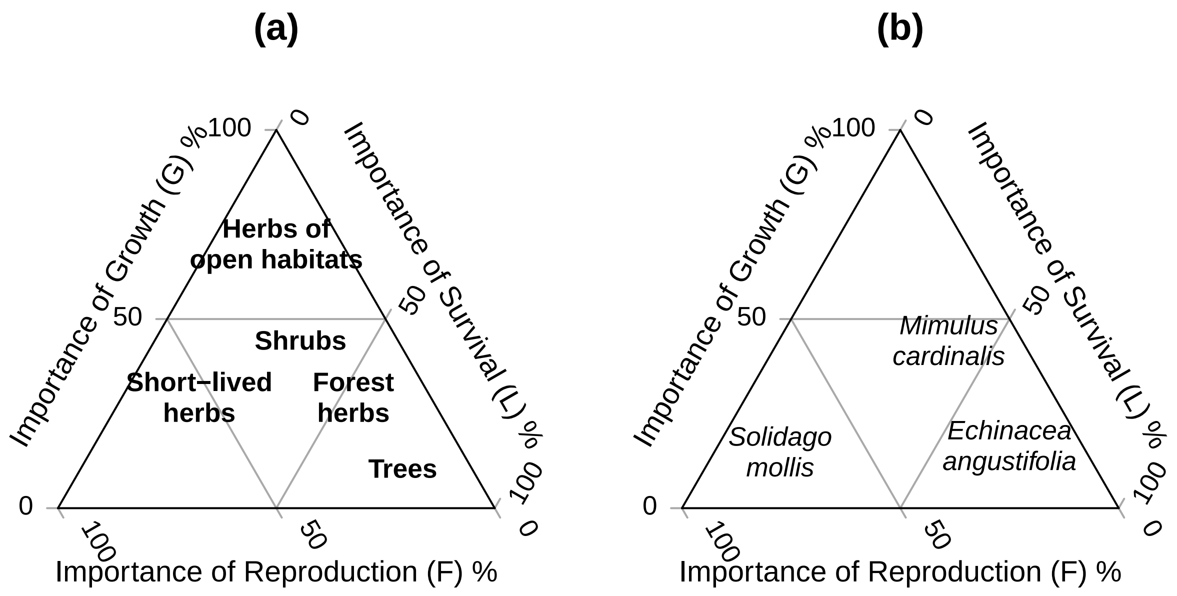
```
par(mfrow=c(1,2),mar=c(1,1,1,1))
TernaryPlot(lab.offset=0.15,alab="Importance of Growth (G) %",
            blab="Importance of Survival (L) %", clab="Importance of Reproduction (F) %",
            grid.lines = 2, lab.cex=0.9, axis.cex=0.8,
            grid.minor.lines = 0, main="(a)")
TernaryText(c(0.25, 0.15, 0.45), labels=c("Short-lived
herbs"), cex=0.8, font=2)
TernaryText(c(0.70, 0.15, 0.15), labels=c("Herbs of
open habitats"), cex=0.8, font=2)
TernaryText(c(0.25, 0.45, 0.15), labels=c("Forest
herbs"), cex=0.8, font=2)
TernaryText(c(0.40, 0.30, 0.20), labels=c("Shrubs"), cex=0.8, font=2)
TernaryText(c(0.1, 0.70, 0.15), labels=c("Trees"), cex=0.8, font=2)

### Demographic triangle calculations for Echinacea
```

```
stasis <- 0 + 0.114 + 0.003 + 0 + 0.161 + 0.009 + 0.459 + 0.002 + 0.005 + 0.007 + 0.017 + 0
fecundity <- sum(elas[1,2:5])
growth <- 0.044 + 0.008 + 0 + 0.046 + 0.014 + 0 + 0.003 + 0.052 + 0.01
sum(stasis, fecundity, growth)
```

```
## [1] 0.9996131
```

```
TernaryPlot(lab.offset=0.15,alab="Importance of Growth (G) %",
            blab="Importance of Survival (L) %", clab="Importance of Reproduction (F) %",
            grid.lines = 2, lab.cex=0.9, axis.cex=0.8,
            grid.minor.lines = 0, main="(b)")
TernaryText(c(0.15, 0.60, 0.15), labels=c("Echinacea
angustifolia"), cex=0.8, font=3)
TernaryText(c(0.40, 0.35, 0.15), labels=c("Mimulus
cardinalis"), cex=0.8, font=3)
TernaryText(c(0.15, 0.15, 0.7), labels=c("Solidago
mollis"), cex=0.8, font=3)
```



Extract mean matrix for Mimulus cardinalis in compadre

```
mimcar <- cdb_check_species(Compadre, "Mimulus cardinalis", return_db = TRUE)
matA <- mean(matA(mimcar))
matU <- mean(matU(mimcar))
matF <- mean(matF(mimcar))
classInfo <- matrixClass(mimcar)[[1]]
matA <- name_stages(matA, c("Seed","Small","Large","Reproductive"))
```

```
## Warning in name_stages(matA, c("Seed", "Small", "Large", "Reproductive")): Naming `prefix` ignored,

## Warning in name_stages(matA, c("Seed", "Small", "Large", "Reproductive")): Existing stage names have
```

```
matU <- name_stages(matU, c("Seed","Small","Large","Reproductive"))
```

```
## Warning in name_stages(matU, c("Seed", "Small", "Large", "Reproductive")): Naming `prefix` ignored,
## Warning in name_stages(matU, c("Seed", "Small", "Large", "Reproductive")): Existing stage names have
```

```
matF <- name_stages(matF, c("Seed","Small","Large","Reproductive"))
```

```
## Warning in name_stages(matF, c("Seed", "Small", "Large", "Reproductive")): Naming `prefix` ignored, u

## Warning in name_stages(matF, c("Seed", "Small", "Large", "Reproductive")): Existing stage names have
```

```
lifeT <- mpm_to_table(matU, matF)
entropy_k(lifeT$lx)
```

```
## [1] 0.3891873
```

```
entropy_d(lifeT$lx,lifeT$mx) # degree of iteroparity
```

```
## [1] 1.314065
```

```
mature_age(matU, matF) # age at maturity
```

```
##     Seed
## 2.245795
```

```
life_expect_mean(matU) #mean life expectancy
```

```
## [1] 1.245936
```

```
longevity(matU) # mature life expectancy
```

```
## [1] 3
```

```
net_repro_rate(matU, matF)
```

```
## [1] 1.340136
```

```
gen_time(matU, matF)
```

```
## [1] 4.4055
```

```
elas <- elasticity(matA)
vr_fecundity(matU, matF)
```

```
## [1] 20613.3
```

```
vr_shrinkage(matU)
```

```
## [1] 0.1622621
```

```
vr_growth(matU)
```

```
## [1] 0.2961599
```

Extract mean matrix for Solidago mollis in compadre

```
solmol <- cdb_check_species(Compadre, "Solidago mollis", return_db = TRUE)
matA <- mean(matA(solmol))
matU <- mean(matU(solmol))
matF <- mean(matF(solmol))
classInfo <- matrixClass(solmol)[[1]]
matA <- name_stages(matA, c("1 yr",">= 2 yr"))
```

```
## Warning in name_stages(matA, c("1 yr", ">= 2 yr")): Naming `prefix` ignored, using stage `names` ins
## Warning in name_stages(matA, c("1 yr", ">= 2 yr")): Existing stage names have been overwritten!
```

```
matU <- name_stages(matU, c("1 yr",">= 2 yr"))
```

```
## Warning in name_stages(matU, c("1 yr", ">= 2 yr")): Naming `prefix` ignored, using stage `names` ins
## Warning in name_stages(matU, c("1 yr", ">= 2 yr")): Existing stage names have been overwritten!
```

```
matF <- name_stages(matF, c("1 yr",">= 2 yr"))
```

```
## Warning in name_stages(matF, c("1 yr", ">= 2 yr")): Naming `prefix` ignored, using stage `names` ins
## Warning in name_stages(matF, c("1 yr", ">= 2 yr")): Existing stage names have been overwritten!
```

```
mature_age(matU, matF) # age at maturity
```

```
## 1 yr
##    1
```

```
lifeT <- mpm_to_table(matU, matF)
entropy_k(lifeT$lx)
```

```
## [1] 0.1790991
```

```
entropy_d(lifeT$lx,lifeT$mx) # degree of iteroparity
```

```
## [1] -0.3339241
```

```
life_expect_mean(matU) #mean life expectancy
```

```
## [1] 1.063112
```

```
longevity(matU) # mature life expectancy
```

```
## [1] 2
```

```
net_repro_rate(matU, matF)
```

```
## [1] 1.775752
```

```
gen_time(matU, matF)
```

```
## [1] 1.0486
```

```
net_repro_rate(matU, matF)
```

```
## [1] 1.775752
```

```
gen_time(matU, matF)
```

```
## [1] 1.0486
```

```
mature_age(matU, matF) # age at maturity
```

```
## 1 yr
##    1
```

```
life_expect_mean(matU) #mean life expectancy
```

```
## [1] 1.063112
```

```
longevity(matU) # mature life expectancy
```

```
## [1] 2
```

```
vr_fecundity(matU, matF)
```

```
## [1] 24.71012
```

```
vr_shrinkage(matU)
```

```
## [1] NA
```

```
vr_growth(matU)
```

```
## [1] NA
```

Extract mean matrix for Silene acaulis in compadre

```
silaca <- cdb_check_species(Compadre, "Silene acaulis", return_db = TRUE)
matA <- mean(matA(silaca))
matU <- mean(matU(silaca))
matF <- mean(matF(silaca))
classInfo <- matrixClass(silaca)[[1]]
matA <- name_stages(matA, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matA, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `name
## Warning in name_stages(matA, classInfo$MatrixClassAuthor): Existing stage names have been overwritten
```

```
matU <- name_stages(matU, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `name
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Existing stage names have been overwritten
```

```
matF <- name_stages(matF, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `name
## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Existing stage names have been overwritten
```

```
lifeT <- mpm_to_table(matU, matF)
entropy_k(lifeT$lx)
```

```
## [1] 2.930251
```

```
entropy_d(lifeT$lx,lifeT$mx)
```

```
## [1] 3.438417
```

```
net_repro_rate(matU, matF)
```

```
## [1] 55.13438
```

```
gen_time(matU, matF)
```

```
## [1] 89.20741
```

```
matU <- name_stages(matU, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `name
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Existing stage names have been overwritten
```

```
matF <- name_stages(matF, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `nam

## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Existing stage names have been overwritte
```

```
mature_age(matU, matF)
```

```
## Seeds in the seedbank
##              10.23312
```

```
life_expect_mean(matU)
```

```
## [1] 11.85832
```

```
longevity(matU)
```

```
## [1] 293
```

```
vr_fecundity(matU, matF)
```

```
## [1] 0.9097834
```

```
vr_shrinkage(matU)
```

```
## [1] 0.1086379
```

```
vr_growth(matU)
```

```
## [1] 0.1931123
```

Extract mean matrix for Asclepias meadii in compadre

```
ascmea <- cdb_check_species(Compadre, "Asclepias meadii", return_db = TRUE)
matA <- mean(matA(ascmea))
matU <- mean(matU(ascmea))
matF <- mean(matF(ascmea))
classInfo <- matrixClass(ascmea)[[1]]
matA <- name_stages(matA, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matA, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `nam

## Warning in name_stages(matA, classInfo$MatrixClassAuthor): Existing stage names have been overwritte
```

```
matU <- name_stages(matU, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `nam

## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Existing stage names have been overwritte
```

```
matF <- name_stages(matF, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `nam

## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Existing stage names have been overwritte
```

```
lifeT <- mpm_to_table(matU, matF)
entropy_k(lifeT$lx)
```

```
## [1] 0.9048362
```

```
entropy_d(lifeT$lx,lifeT$mx) # degree of iteroparity
```

```
## [1] 5.90408
```

```
net_repro_rate(matU, matF)
```

```
## [1] 0.04838085
```

```
gen_time(matU, matF)
```

```
## [1] 20.92615
```

```
matU <- name_stages(matU, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `nam
## Warning in name_stages(matU, classInfo$MatrixClassAuthor): Existing stage names have been overwritte
```

```
matF <- name_stages(matF, classInfo$MatrixClassAuthor)
```

```
## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Naming `prefix` ignored, using stage `nam
## Warning in name_stages(matF, classInfo$MatrixClassAuthor): Existing stage names have been overwritte
```

```
matA <- matU + matF
mature_age(matU, matF) # age at maturity
```

```
## Seedling
## 8.383499
```

```
life_expect_mean(matU) #mean life expectancy
```

```
## [1] 6.53298
```

```
longevity(matU) # mature life expectancy
```

```
## [1] 29
```

```
vr_fecundity(matU, matF)
```

```
## [1] 0.5454112
```

```
vr_shrinkage(matU)
```

```
## [1] 0.421932
```

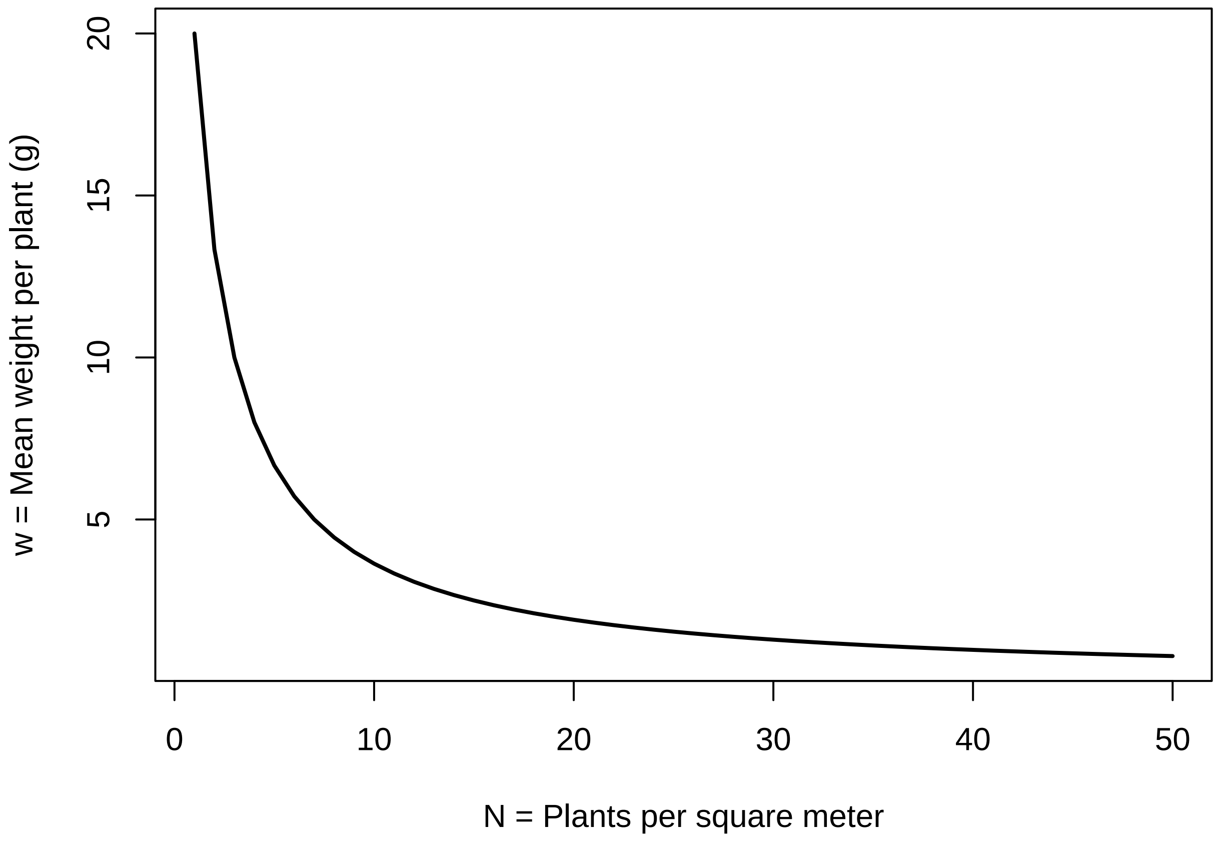
```
vr_growth(matU)
```

```
## [1] 0.4700773
```

Density dependence

```
wm <- 40
N <- c(1:50)
a <- 1
w <- wm * (1 + a*N)^-1

par(mar=c(4,4,1,1))
plot(N,w, type="l",lwd=2,
     ylab="w = Mean weight per plant (g)", xlab="N = Plants per square meter")
```



Build an Integral Projection Model based on scripts from Ellner et al. 2016, thanks to Dave Atkins for assistance with this code and data preparation

```
# load data
dat <- read.csv("twoSpp.csv", header=TRUE)

# convert area to log area
dat$area_t <- log(dat$area_t)
dat$area_tplus1 <- log(dat$area_tplus1)

# subset data to analyse one species
spDF <- dat %>% filter(species=="Festuca arizonica")

# ignore individuals with area < 0.05 cm^2 generated from converting points to polygons
spDF$area_t[spDF$area_t < (log(0.05))] <- NA
spDF$area_tplus1[spDF$area_tplus1 < (log(0.05))] <- NA

## set up vector of parameters
params=data.frame(
  surv.int=NA,
  surv.slope=NA,
  #####
  growth.int=NA,
  growth.slope=NA,
  growth.sd=NA,
  #####
  flwr.int=NA,
  flwr.slope=NA,
  #####
  seed.int=NA,
  seed.slope=NA,
  #####
  recruit.size.mean=NA,
  recruit.size.sd=NA,
  #####
  establishment.prob=NA
)
```

Fit vital rate regression models

```
# 1. survival regression using logistic regression
surv.reg = glm(survives_tplus1 ~ area_t, data=spDF, family=binomial)
params$surv.int = coefficients(surv.reg)[1]
params$surv.slope = coefficients(surv.reg)[2]
summary(surv.reg)
```

```
##
## Call:
## glm(formula = survives_tplus1 ~ area_t, family = binomial, data = spDF)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6634  -0.7608   0.5872   0.7985   1.6926
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.33431    0.04768   7.012 2.35e-12 ***
## area_t       0.51412    0.02618  19.641  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5028.4  on 4410  degrees of freedom
## Residual deviance: 4550.0  on 4409  degrees of freedom
##   (1117 observations deleted due to missingness)
## AIC: 4554
##
## Number of Fisher Scoring iterations: 4
```

```
# 2. growth regression using linear model
growth.reg=lm(area_tplus1 ~ area_t, data=spDF)
params$growth.int=coefficients(growth.reg)[1]
params$growth.slope=coefficients(growth.reg)[2]
params$growth.sd=sd(resid(growth.reg))
```

```
summary(growth.reg)
```

```
##
## Call:
## lm(formula = area_tplus1 ~ area_t, data = spDF)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5849 -0.7266  0.0500  0.7535  3.7080
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.55275    0.03180   17.38   <2e-16 ***
## area_t       0.72366    0.01227   58.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.104 on 3224 degrees of freedom
##   (2302 observations deleted due to missingness)
## Multiple R-squared:  0.5189, Adjusted R-squared:  0.5188
## F-statistic:  3478 on 1 and 3224 DF,  p-value: < 2.2e-16
```

```
## 3. flowering probability using logistic regression
flower.reg = glm(flwr.sim ~ area_t, data=spDF, family=binomial)
params$flwr.int = coefficients(flower.reg)[1]
params$flwr.slope = coefficients(flower.reg)[2]
summary(flower.reg)
```

```
##
## Call:
## glm(formula = flwr.sim ~ area_t, family = binomial, data = spDF)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9340  -0.9214  -0.7095   1.1652   2.3330
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.32647    0.05066  -26.18   <2e-16 ***
## area_t       0.45661    0.02126   21.48   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6485.9  on 4898  degrees of freedom
## Residual deviance: 5944.7  on 4897  degrees of freedom
##   (629 observations deleted due to missingness)
## AIC: 5948.7
##
## Number of Fisher Scoring iterations: 4
```

```
# 4. seeds regression using poisson regression
# note that the seeds in this example were simulated from an empirical relationship
seed.reg = glm(seed.sim ~ area_t, data = spDF, family = "poisson")
```

```
params$seed.int=coefficients(seed.reg)[1]
params$seed.slope=coefficients(seed.reg)[2]
summary(seed.reg)
```

```
##
## Call:
## glm(formula = seed.sim ~ area_t, family = "poisson", data = spDF)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -106.02   -17.52   -10.95    11.27    77.17
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 3.972351   0.001868    2126   <2e-16 ***
## area_t      0.702698   0.000462    1521   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4625651  on 4898  degrees of freedom
## Residual deviance: 2198539  on 4897  degrees of freedom
##   (629 observations deleted due to missingness)
## AIC: 2213674
##
## Number of Fisher Scoring iterations: 6
```

```
# 5. size distribution of recruits using Gaussian distribution
params$recruit.size.mean = mean(spDF$area_t[spDF$recruit==1], na.rm =TRUE)
params$recruit.size.sd = sd(spDF$area_t[spDF$recruit==1], na.rm =TRUE)

## 6. establishment probability
params$establishment.prob = sum(spDF$recruit, na.rm = TRUE) / sum(spDF$seed.sim[spDF$flwr.sim==1], na.r
```

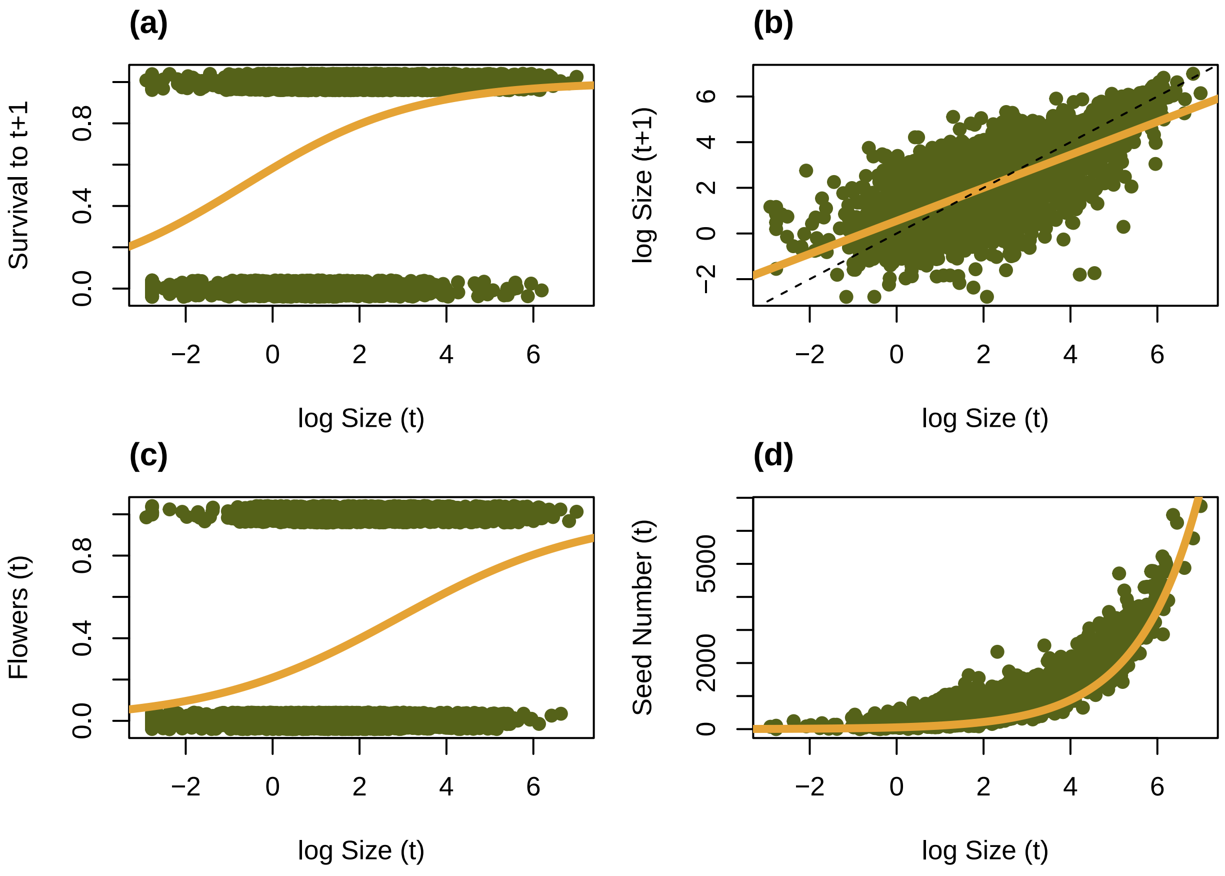
Plot vital rate regression models

```
par(mfrow=c(2,2), mar=c(4,4,2,1))
cols <- met.brewer("Degas",7)
xx=seq(-4,8,length.out=1000) # sizes at which to evaluate predictions
plot(spDF$area_t, jitter(spDF$survives_tplus1, 0.2), xlab="log Size (t)",ylab="Survival to t+1", col=co
lines(xx,predict(surv.reg, data.frame(area_t=xx), type="response"), col=cols[3],lwd=4)
title("(a)", adj = 0, line = 1)

plot(spDF$area_t, spDF$area_tplus1, xlab="log Size (t)", ylab="log Size (t+1)", col=cols[4], pch=19)
lines(xx,predict(growth.reg, data.frame(area_t=xx)), col=cols[3],lwd=4)
title("(b)", adj = 0, line = 1)
abline(0,1, lty=2)

plot(spDF$area_t, jitter(spDF$flwr.sim, 0.2), xlab="log Size (t)",ylab="Flowers (t)", col=cols[4], pch=
lines(xx,predict(flower.reg, data.frame(area_t=xx), type="response"),col=cols[3],lwd=4)
title("(c)", adj = 0, line = 1)
```

```
plot(spDF$area_t[spDF$flwr.sim==1], spDF$seed.sim[spDF$flwr.sim==1], xlab="log Size (t)", ylab="Seed Nu
     col=cols[4], pch=19)
lines(xx,predict(seed.reg,data.frame(area_t=xx),type="response"),col=cols[3],lwd=4)
title("(d)", adj = 0, line = 1)
```



Create IPM functions

```
## Growth function using linear model
g_z1z <- function(area_tplus1, area_t, params)
{
  mu <- params$growth.int + params$growth.slope * area_t     # mean size next year
  sig <- params$growth.sd                                    # sd about mean
  p.den.grow <- dnorm(area_tplus1, mean = mu, sd = sig)    # pdf that you are size area_tplus1 given you
  return(p.den.grow)
}

## Survival function using logistic regression
s_z <- function(area_t, params)
{
  linear.p <- params$surv.int + params$surv.slope * area_t # linear predictor
  p <- 1/(1+exp(-linear.p))                               # logistic transformation to probability
  return(p)
}
```

```r
## Probability of flowering function using logistic regression
p_bz <- function(area_t, params)
{
  linear.p <- params$flwr.int + params$flwr.slope * area_t      # linear predictor
  p <- 1/(1+exp(-linear.p))                                  # logistic transformation to probability
  return(p)
}

## Seed production function using poisson regression
b_z <- function(area_t, params)
{
  N <- exp(params$seed.int + params$seed.slope * area_t)    # seed production of a size z plant
  return(N)
}

## Recruit size pdf using Gaussian distribution
c_0z1 <- function(area_t, params)
{
  mu <- params$recruit.size.mean
  sig <- params$recruit.size.sd
  p.deRecr <- dnorm(area_t, mean = mu, sd = sig)        # pdf of a size z1 recruit
  return(p.deRecr)
}
```

Define kernels

```r
## Define the survival kernel
P_z1z <- function (area_tplus1, area_t, params) {
  return( s_z(area_t, params) * g_z1z(area_tplus1, area_t, params) )
}

## Define the reproduction kernel
F_z1z <- function (area_tplus1, area_t, params) {
  return( p_bz(area_t, params) * b_z(area_t, params) * params$establishment.prob * c_0z1(area_tplus1, pa
}

## Build the discretized kernel
mk_K <- function(m, params, L, U) {
  # mesh points
  h <- (U - L)/m
  meshpts <- L + ((1:m) - 1/2) * h
  P <- h * (outer(meshpts, meshpts, P_z1z, params = params))
  F <- h * (outer(meshpts, meshpts, F_z1z, params = params))
  K <- P + F
  ## compute the eigen vectors / values
  IPM.eig.sys <- eigen(K)
  ## lambda
  lambda <- Re(IPM.eig.sys$values[1])
  return(list(lambda=lambda, K = K, meshpts = meshpts, P = P, F = F))
}
```

Fit kernel and examine lambda

```
## set upper and lower integration limits to avoid evictions
lim1 <- 0.9*min(spDF$area_t, na.rm=TRUE)
lim2 <- 1.1*max(spDF$area_t, na.rm=TRUE)

out <- mk_K(m=100, params=params, L=lim1, U=lim2)
out$lambda
```

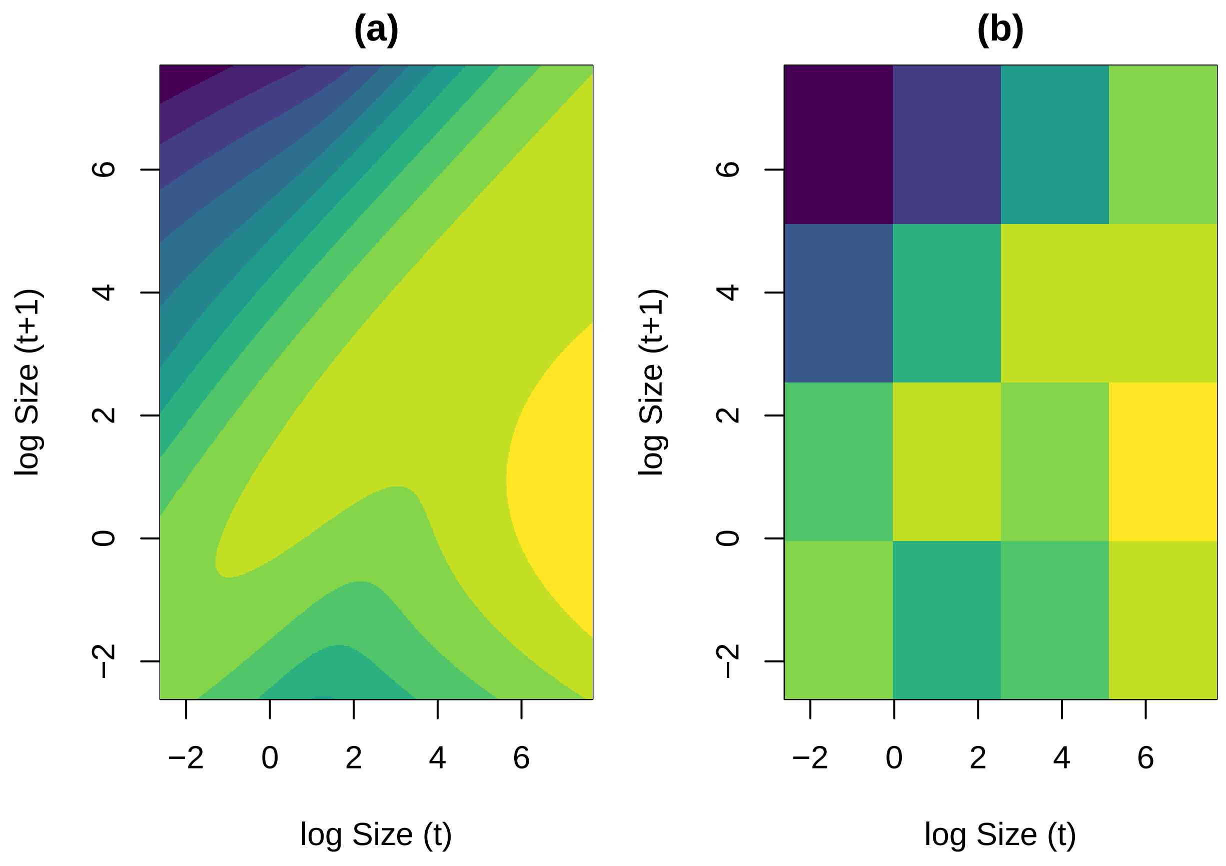
```
## [1] 0.9961385
```

Plot the kernel

```
par(mfrow=c(1,2), mar=c(4,4,2,1))

# model with 500 very small size classes
out <- mk_K(m=500, params=params, L=lim1, U=lim2)
image(out$meshpts, out$meshpts, t(out$K)^0.01, col = viridis_pal()(12), ylim=c(lim1,lim2),xlim=c(lim1,li
      xlab="log Size (t)", ylab="log Size (t+1)", main="(a)")

# compare to a model with only 4 size classes more similar to matrix model
out <- mk_K(m=4, params=params, L=lim1, U=lim2)
image(out$meshpts, out$meshpts, t(out$K)^0.05, col = viridis_pal()(12), ylim=c(lim1,lim2),xlim=c(lim1,li
      xlab="log Size (t)", ylab="log Size (t+1)", main="(b)")
```


(a)
(b)
log Size (t+1)
log Size (t)