Wood's Quadratic Programming Method

Sophie Wulfing

2/17/2022

```
##              [,1]
##  [1,]  76.5230312
##  [2,]  27.8603269
##  [3,]   2.2288262
##  [4,]   1.8573551
##  [5,]  57.5780089
##  [6,]  37.8900446
##  [7,]   1.8573551
##  [8,]   0.0000000
##  [9,]  40.4903417
## [10,]  50.8915305
## [11,]   3.3432392
## [12,]   0.0000000
## [13,]  71.6939079
## [14,]  16.7161961
## [15,]   8.1723626
## [16,]   1.1144131
## [17,] 121.0995542
## [18,]  28.9747400
## [19,]   5.5720654
## [20,]   2.2288262
## [21,] 119.9851412
## [22,]  52.0059435
## [23,]   6.6864785
## [24,]   0.7429421
## [25,]  78.7518574
## [26,]  41.6047548
## [27,]  14.4873700
## [28,]   1.1144131
## [29,] 118.8707281
## [30,]  53.4918276
## [31,]  14.4873700
## [32,]   1.1144131
## [33,] 119.9851412
## [34,]  39.0044576
## [35,]  10.7726597
## [36,]   1.1144131
## [37,]  73.5512630
## [38,]  26.3744428
## [39,]   4.4576523
## [40,]   2.2288262
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
```

```
##  [1,]   -1    0    0    0    0    0    0    0
##  [2,]    0   -1    0    0    0    0    0    0
##  [3,]    0    0   -1    0    0    0    0    0
##  [4,]    0    0    0   -1    0    0    0    0
##  [5,]    0    0    0    0   -1    0    0    0
##  [6,]    0    0    0    0    0   -1    0    0
##  [7,]    0    0    0    0    0    0   -1    0
##  [8,]    0    0    0    0    0    0    0   -1
##  [9,]    1    1    0    0    0    0    0    0
## [10,]    0    0    1    1    0    0    0    0
## [11,]    0    0    0    0    1    1    0    0
## [12,]    0    0    0    0    0    0    0    1
```

```
##           [,1]      [,2]       [,3]       [,4]
## [1,] 0.6295838 0.0000000 0.00000000 26.7004983
## [2,] 0.2752164 0.3219070 0.00000000  0.0000000
## [3,] 0.0000000 0.1300588 0.39275810  0.0000000
## [4,] 0.0000000 0.0000000 0.09317825  0.3309474
```

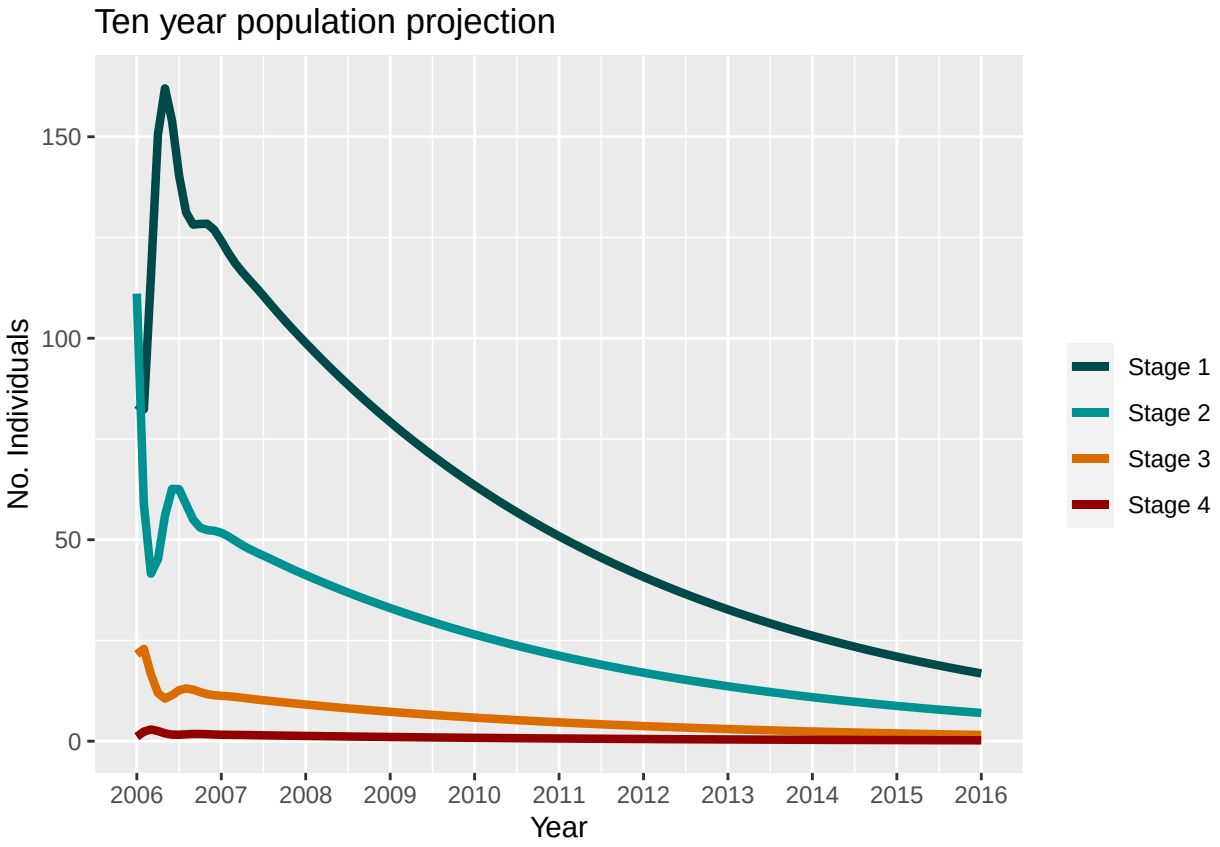
```
 [,1] [,2] [,3] [,4]
```

[1,] “P1” “0” “0” “F4” [2,] “G1” “P2” “0” “0” [3,] “0” “G2” “P3” “0” [4,] “0” “0” “G3” “P4”

$$\begin{bmatrix} P1 & 0 & 0 & F4 \\ G1 & P2 & 0 & 0 \\ 0 & G2 & P3 & 0 \\ 0 & 0 & G3 & P4 \end{bmatrix}$$

$$\begin{bmatrix} 0.629583847097241 & 0 & 0 & 26.7004982678535 \\ 0.275216425741737 & 0.321906972442815 & 0 & 0 \\ 0 & 0.130058765403675 & 0.392758096317347 & 0 \\ 0 & 0 & 0.0931782465194735 & 0.330947383185268 \end{bmatrix}$$

$$\begin{bmatrix} P1 = 0.63 & 0 & 0 & F4 = 26.7 \\ G1 = 0.275 & P2 = 0.322 & 0 & 0 \\ 0 & G2 = 0.13 & P3 = 0.393 & 0 \\ 0 & 0 & G3 = 0.093 & P4 = 0.331 \end{bmatrix}$$




```
#eigenvecors and vals
A_eigen <- eigen(A)
A_eigen
```

```
## eigen() decomposition
## $values
## [1]  0.9817200+0.0000000i  0.4166356+0.5323073i  0.4166356-0.5323073i
## [4] -0.1397949+0.0000000i
##
## $vectors
##                [,1]                     [,2]                     [,3]
## [1,] -0.91954608+0i -0.88561224+0.00000000i -0.88561224+0.00000000i
## [2,] -0.38355441+0i -0.07898305+0.44382834i -0.07898305-0.44382834i
## [3,] -0.08469922+0i  0.10735902+0.02411371i  0.10735902-0.02411371i
## [4,] -0.01212732+0i  0.00706315-0.01765577i  0.00706315+0.01765577i
##                [,4]
## [1,] -0.85207585+0i
## [2,]  0.50791492+0i
## [3,] -0.12404171+0i
## [4,]  0.02455269+0i
```

```
#Intrinsic Rate of Increast (r): lambda = e^r
r <- log(A_eigen$values[1])
r
```

```
## [1] -0.01844917+0i
```

```r
#stable stage dist
A_stable_stage <- A_eigen$vectors[,1]/sum(A_eigen$vectors[,1])
A_stable_stage
```

```
## [1] 0.65685286+0i 0.27398172+0i 0.06050260+0i 0.00866282+0i
```

```r
#reproductive value
A_repro_value <- eigen(t(A))$vectors[,1]/eigen(t(A))$vectors[1,1]
A_repro_value
```

```
## [1]  1.000000+0i  1.279488+0i  6.491088+0i 41.028923+0i
```

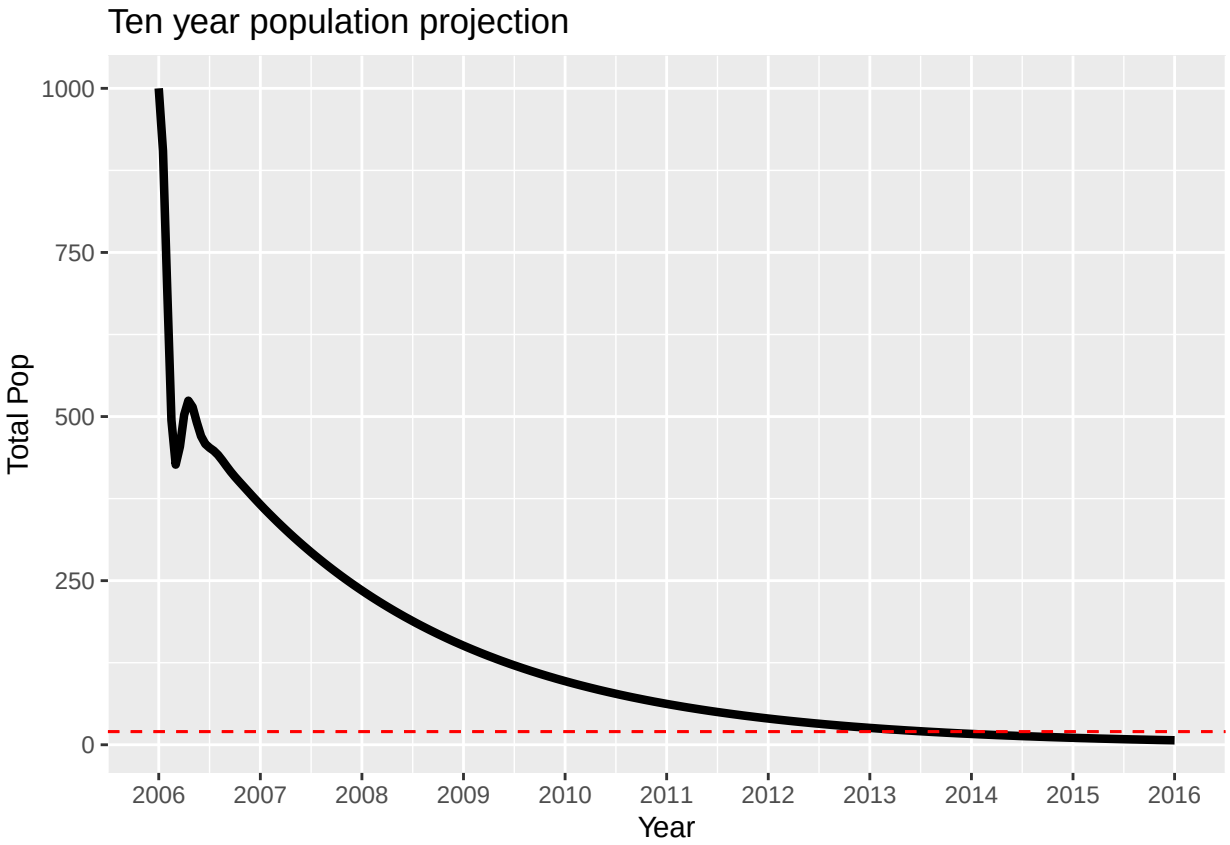
```r
#mean reproductive value- is the avg no offspring?
A_repro_value %*% A_stable_stage
```

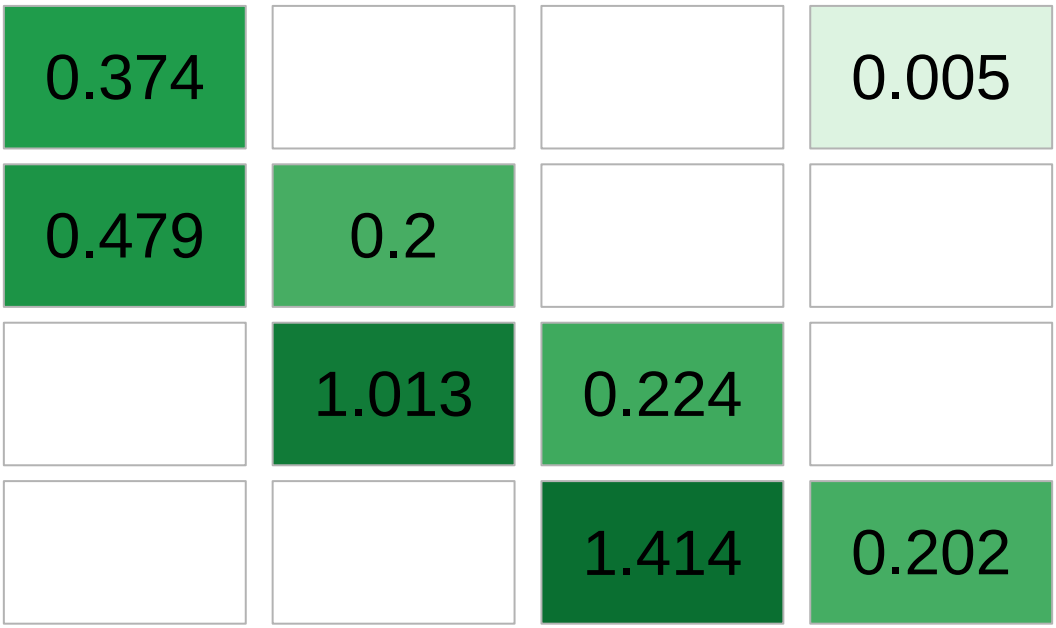
```
##             [,1]
## [1,] 1.755563+0i
```

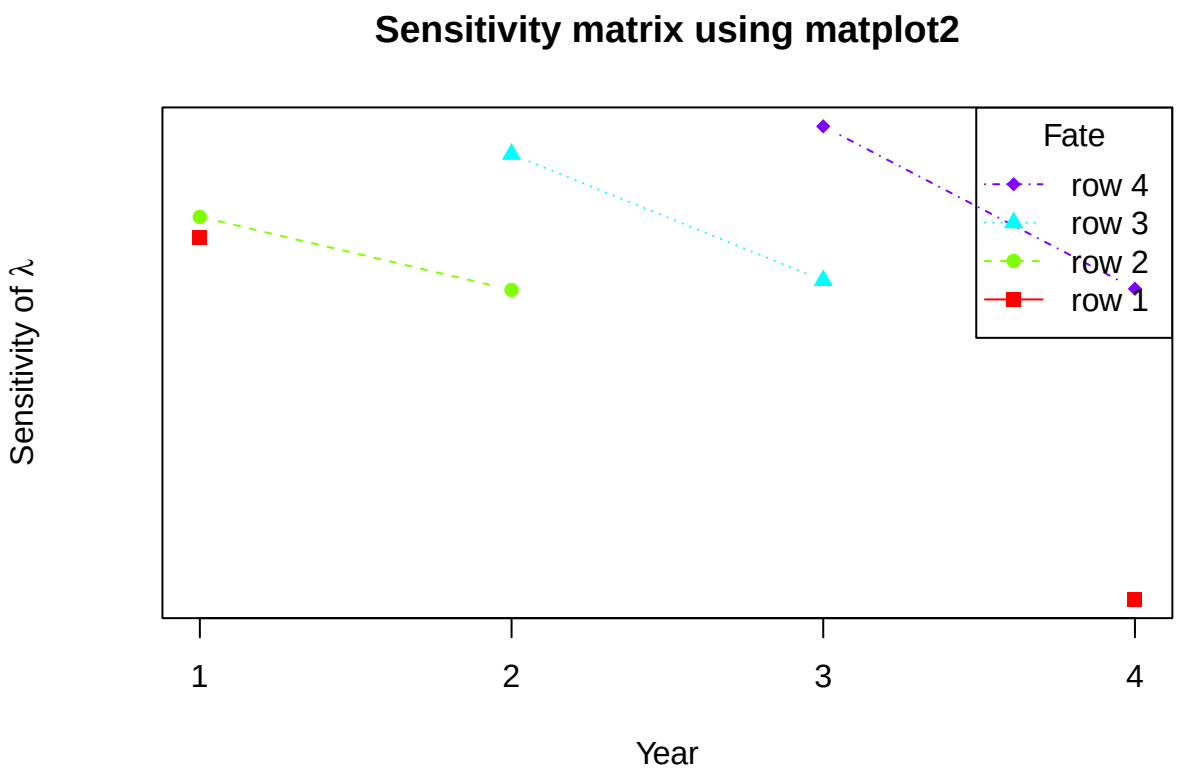
```r
#. Vandermeer (1975, 1978)

#DO KEYFIT FUNCTION:
## Keyfitz function
keyfitz<-function(x,y){ # you provide the observed x
sum(abs(x-y))/2 # and stable stage dist vectors
}
#SEE https://cws.auburn.edu/shared/files%3Fid=217&filename=ConMan_FileDownload_MatrixPopulation.pdf

#Good eigval and vector sources;
#https://setosa.io/ev/eigenvectors-and-eigenvalues/
#http://biom300.weebly.com/eigenvalues-and-eigenvectors-in-r.html
```


Ten year population projection
Total Pop
1000
750
500
250
0
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
Year


0.374
0.005
0.479
0.2
1.013
0.224
1.414
0.202

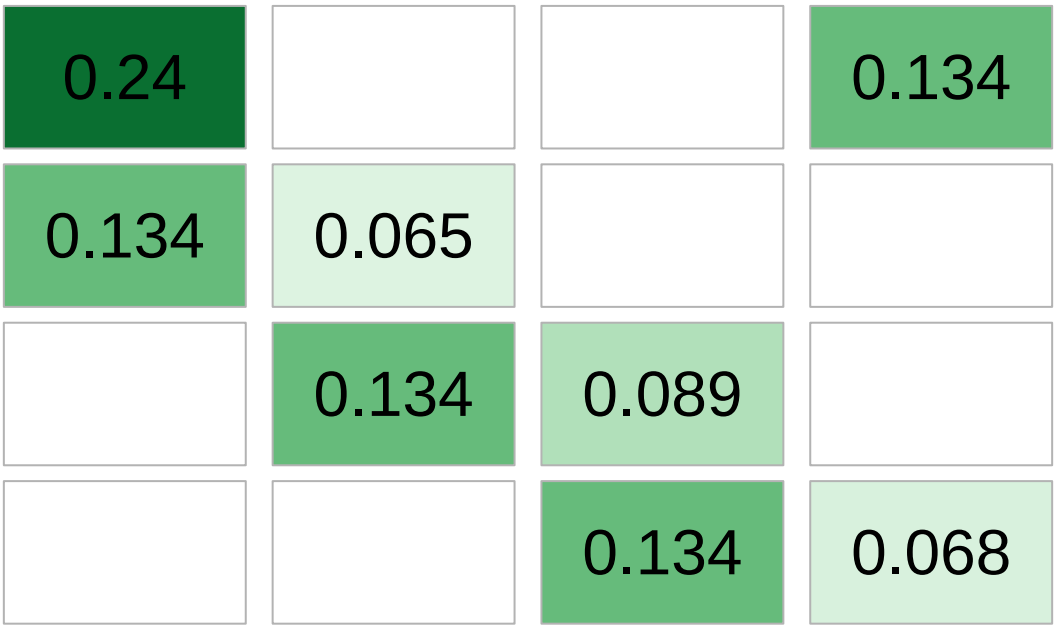



```
cols <- hcl.colors(1000, palette = "Greens 3", alpha = NULL, rev = TRUE, fixup = TRUE)#, end = .85)

elas <- elasticity(A)

for(i in 1:length(A)){
  if(A[i] == 0){
    elas[i] <-  NA
  }
}

image2(elas, mar=c(1,3.5,5,1), border="gray70", col = c("white", cols[150:850]), text.cex = 2 )
```




```
# # Summed elasticities for teasel.
# # fertility in last column, stasis P on diagonal, and growth in bottom-left triangle
# c(F=sum(elas[,4]), P=sum(diag(elas)), G=sum(elas[row(elas)>col(elas)]))
#
# elas <- elasticity(tortoise[["med.high"]])
# image2(elas, mar=c(1,3.5,5,1),  log=FALSE)
#  title("Tortoise elasticity matrix", line=2.5)
# # Summed elasticities for tortoise (see example 9.4)
# # fertility in top row, stasis on diagonal, and growth on subdiagonal
# c(F=sum(elas[1,]), P=sum(diag(elas)), G=sum(elas[row(elas)==col(elas)+1]))

#https://rdrr.io/cran/popbio/man/elasticity.html
```

Possible Helpful Links:

https://stackoverflow.com/questions/12349122/solving-quadratic-programming-using-r

https://stackoverflow.com/questions/55727368/how-to-minimize-a-function-in-r-with-two-constraints

https://stackoverflow.com/questions/31301694/least-square-optimization-of-matrices-in-r?rq=1

https://henrywang.nl/quadratic-programming-with-r/

https://cran.r-project.org/web/packages/quadprog/quadprog.pdf

Also when you start doing this in markdown, here's the website for citations: https://www.anthonyschmidt.co/post/2021-10-25-a-zotero-workflow-for-r/