Transient Perturbation Analysis

Load the popdemo package

```
library(popdemo)
```

Data

We will use a matrix for the desert tortoise *Gopherus agassizzii*, with medium fecundity, which was published in Doak et al. (1994) Ecol. Appl., 4, 446-460. There are 8 stages are based on age and size (carapace length in mm):
Yearling (age 0-1)
Juvenile 1 (<60 mm)
Juvenile 2 (90-99mm)
Immature 1 (100-139mm)
Immature 2 (140-179mm)
Subadult (180-207mm)
Adult 1 (208-239mm)
Adult 2 (>240mm).

Load in the data

```
data(Tort)
```

Exercise 1: Sensitivity of population inerita

Using popdemo, it is possible to calculate the sensitivity of population inertia in much the same way as for lambda. Transient sensitivities are a little different to asymptotic sensitivites: they may be positive or negative. Counterintuitively, increasing some vital rates can lead to a decrease in the transient dynamics!

We calculate the sensitivity of population inertia because it's an index that is very amenable to mathematical trickery. Although it is a measure of transient dynamics (transient dynamics lead to the population having an inertia), it is calculated using the dominant eigenvectors of the matrix.

Calculating the sensitivity values for inertia is super simple. Let's do it for a specific population vector first:

```
Tortvec<-Matlab2R("[1;1;2;3;5;8;13;21]")
( Sens1<-tfsm_inertia(Tort, Tortvec, tolerance=1e-5) )
```

```
##          [,1]      [,2]      [,3]      [,4]      [,5]       [,6]
## [1,] 0.000000  0.000000  0.000000  0.000000  0.000000  0.1059443
## [2,] 1.828402  2.235003  0.000000  0.000000  0.000000  0.0000000
## [3,] 0.000000 -2.469342 -2.000924  0.000000  0.000000  0.0000000
## [4,] 0.000000  0.000000 -6.599622 -3.867125  0.000000  0.0000000
## [5,] 0.000000  0.000000  0.000000 -4.906062 -3.350736  0.0000000
```

```
## [6,] 0.000000  0.000000  0.000000  0.000000 -4.714519 -3.4406241
## [7,] 0.000000  0.000000  0.000000  0.000000  0.000000 -2.4693636
## [8,] 0.000000  0.000000  0.000000  0.000000  0.000000  0.0000000
##            [,7]       [,8]
## [1,]  0.1427843  0.7018267
## [2,]  0.0000000  0.0000000
## [3,]  0.0000000  0.0000000
## [4,]  0.0000000  0.0000000
## [5,]  0.0000000  0.0000000
## [6,]  0.0000000  0.0000000
## [7,] -8.0661786  0.0000000
## [8,] -8.7688976 16.3527272
```

Many of the matrix transitions have a negative effect on population inertia, but there's a large positive effect of the survival of the largest adults.

We can also calculate sensitivities for the bounds on population dynamics.

```
( Sensupr<-tfsm_inertia(Tort, bound="upper", tolerance=1e-5) )
```

```
##         [,1]      [,2]      [,3]      [,4]      [,5]        [,6]
## [1,] 0.00000  0.000000  0.000000  0.000000  0.000000 -0.02836702
## [2,] 2.61707  3.400695  0.000000  0.000000  0.000000  0.00000000
## [3,] 0.00000 -1.379916 -1.916767  0.000000  0.000000  0.00000000
## [4,] 0.00000  0.000000 -6.697363 -4.515380  0.000000  0.00000000
## [5,] 0.00000  0.000000  0.000000 -5.665248 -4.687062  0.00000000
## [6,] 0.00000  0.000000  0.000000  0.000000 -6.791893 -8.16874179
## [7,] 0.00000  0.000000  0.000000  0.000000  0.000000 -6.62429939
## [8,] 0.00000  0.000000  0.000000  0.000000  0.000000  0.00000000
##             [,7]     [,8]
## [1,]  -0.7376066  1.75365
## [2,]   0.0000000  0.00000
## [3,]   0.0000000  0.00000
## [4,]   0.0000000  0.00000
## [5,]   0.0000000  0.00000
## [6,]   0.0000000  0.00000
## [7,] -30.4594953  0.00000
## [8,] -34.7995915 43.43719
```

```
( Senslwr<-tfsm_inertia(Tort, bound="lower", tolerance=1e-5) )
```

```
##           [,1]      [,2]      [,3]       [,4]       [,5]        [,6]
## [1,] 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000  0.01093941
## [2,] 0.2154514 0.3414165 0.0000000 0.00000000 0.00000000  0.00000000
## [3,] 0.0000000 0.5028474 0.1385316 0.00000000 0.00000000  0.00000000
## [4,] 0.0000000 0.0000000 0.2999582 0.06148234 0.00000000  0.00000000
## [5,] 0.0000000 0.0000000 0.0000000 0.13604670 0.02909221  0.00000000
## [6,] 0.0000000 0.0000000 0.0000000 0.00000000 0.10871337 -0.01614332
## [7,] 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000  0.01684339
## [8,] 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000  0.00000000
##               [,7]         [,8]
```

```
## [1,] -0.0001746125 -0.004585577
## [2,]  0.0000000000  0.000000000
## [3,]  0.0000000000  0.000000000
## [4,]  0.0000000000  0.000000000
## [5,]  0.0000000000  0.000000000
## [6,]  0.0000000000  0.000000000
## [7,] -0.5351484154  0.000000000
## [8,] -0.5944874594 -0.216656349
```

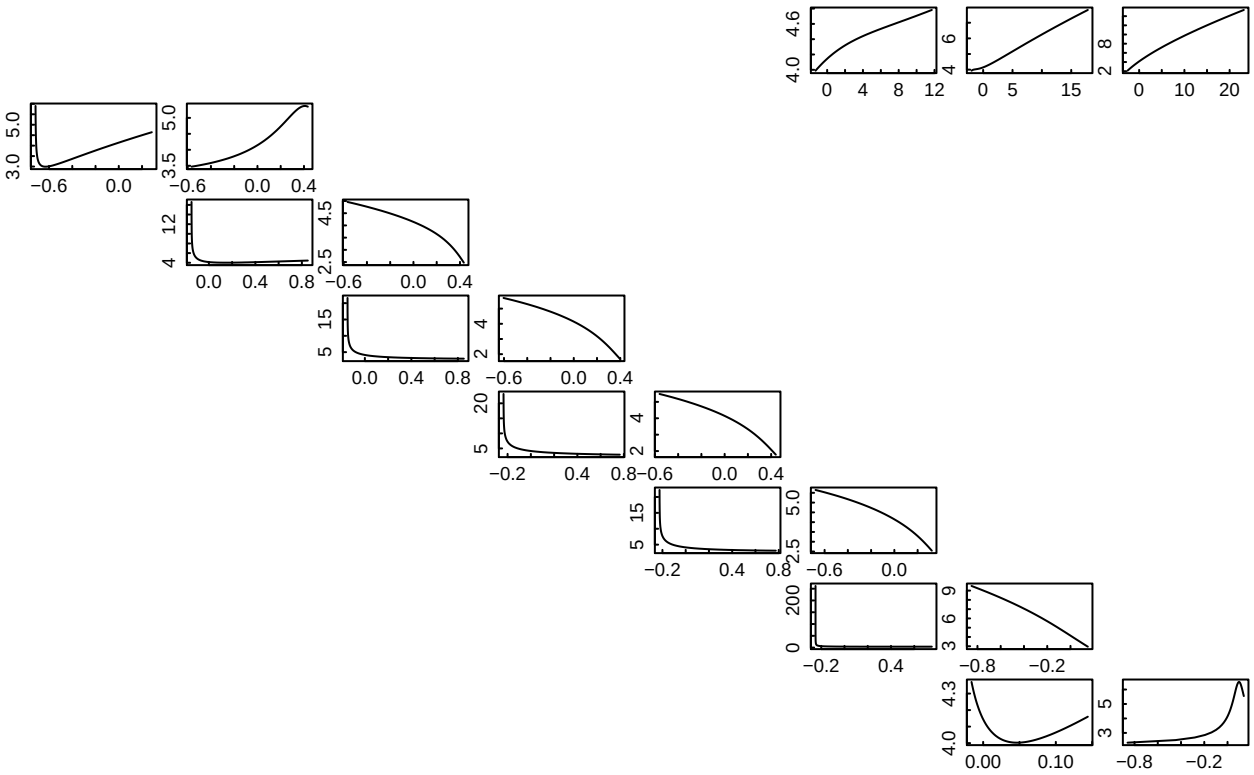
Note that the sensitivities of the different bounds are different. This will be the same for any population vector: the sensitivities depend on the population structure.

Exercise 2: Transfer function analyses

In actual fact, the sensitivites of any demographic quantity, be it asymptotic or transient growth, are linearisations of nonlinear relationships. The effect of changing a life cycle transition or a vital rate on transient population dynamics can be very nonlinear: sensitivities are tangents to these functions at infinitessimally small perturbation magnitudes of near 0 (i.e. differentiations of the curve).

Popdemo provides tools for visualising the nonlinear relationships between matrix entries and population dynamics. In the function in the previous exercise, tfsm stands for transfer function sensitivity matrix. This function calculates transfer functions of inertia and then differentiates them to get the sensitivity (we also needed to tweak the tolerance of the algorithm that calculates the differential for weird mathematical reasons). We can also do transfer function analysis across the whole matrix.

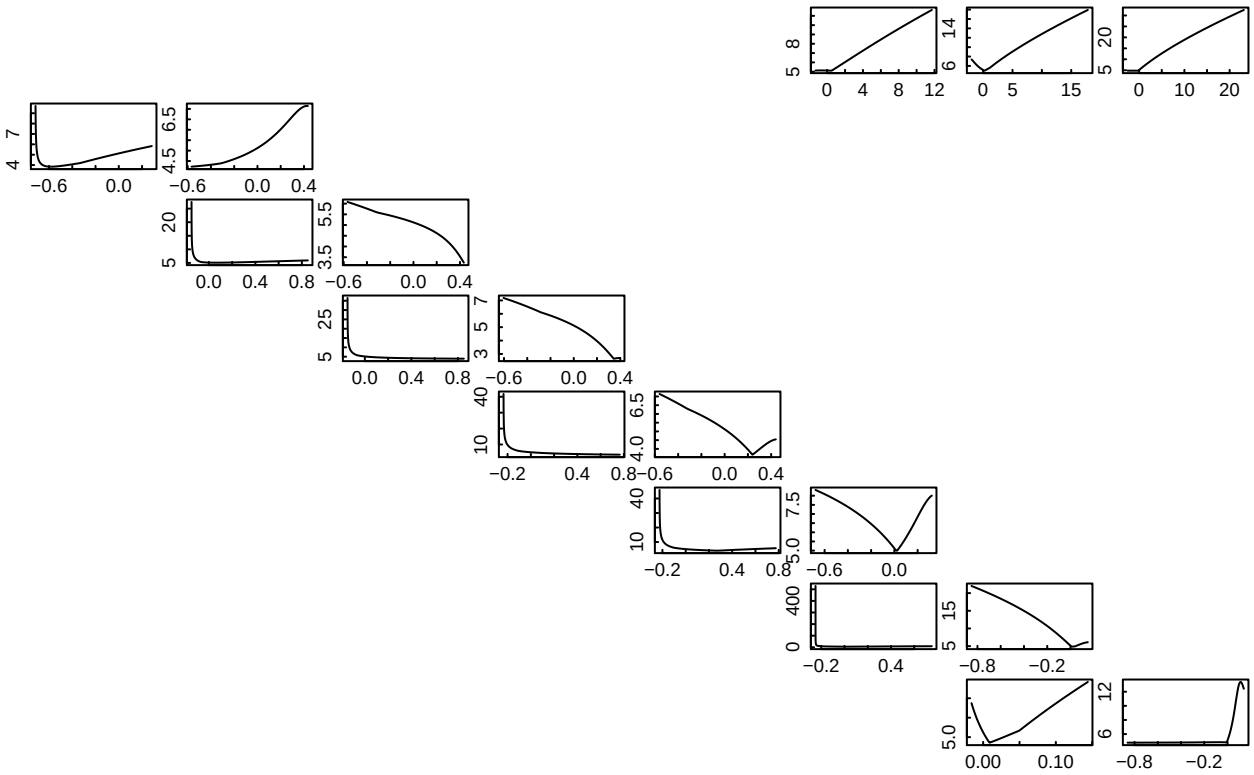
```
TFmat1<-tfam_inertia(Tort, vector=Tortvec)
plot(TFmat1)
```



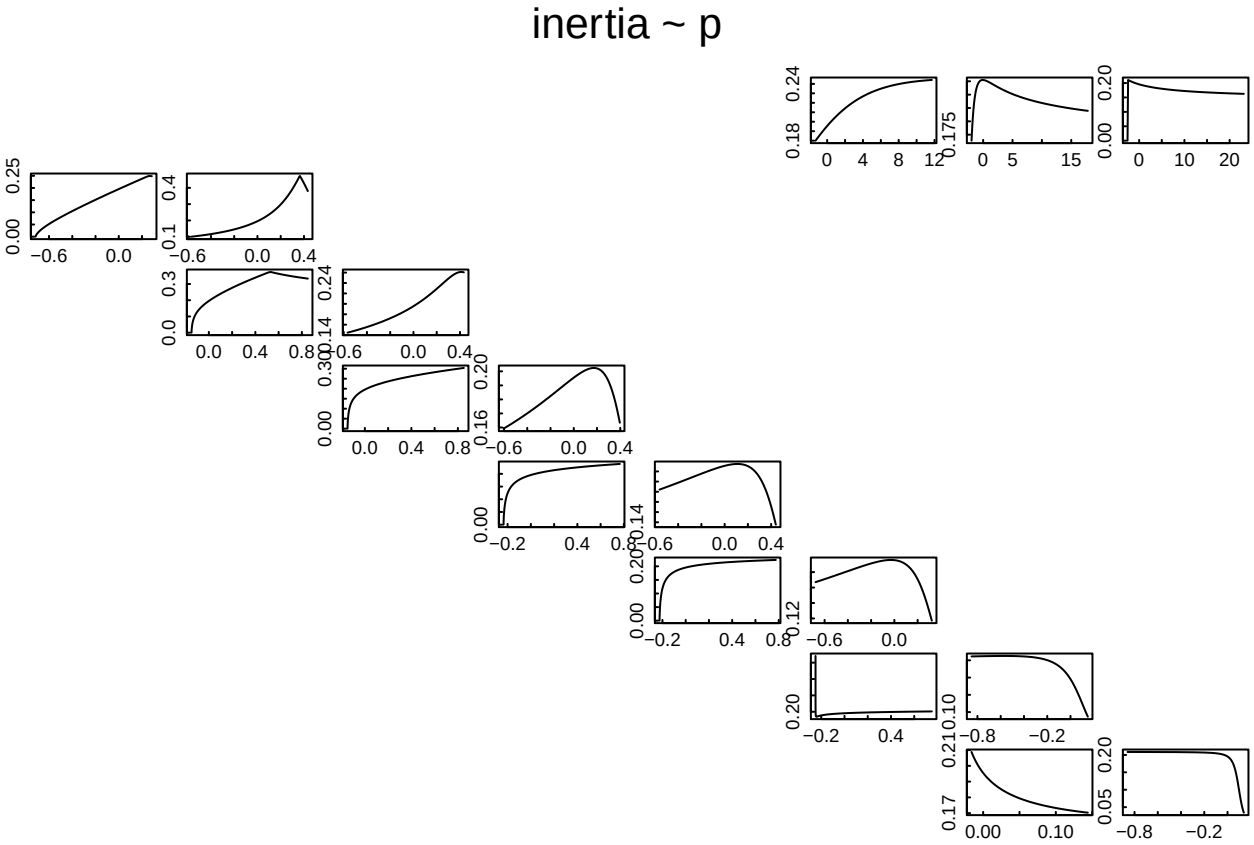
On the x axes are the perturbation magnitudes (how much you change the entry of the matrix by), and on the y axis is the value of inertia. Note how nonlinear the curves can be. The sensitivities are tangents to these curves at p=0.

We can do the same thing for the bounds:

```
TFmatupr<-tfam_inertia(Tort, bound="upper")
plot(TFmatupr)
```




```
TFmatlwr<-tfam_inertia(Tort, bound="lower")
plot(TFmatlwr)
```



These can be incredibly nonlinear, because there can be breaks in the function where a different stage-biased population structure takes over as achieving the bound on inertia. Note where there are kinks in the lines, or switches in the direction of the slope.

Exercise 3: COM(P)ADRE tasks

You can download the COMPADRE and COMADRE databases here. Save the data somewhere on your computer and load it up using 'r load(your/file/path/here/compadre.Rdata). Then, pick a matrix to work with. For example:

```
M<-comadre$mat[comadre$metadata$SpeciesAccepted=="Ursus_maritimus"][[1]]$matA
```

This chooses us a matrix for the polar bear *Ursus maritimus*.

1. Calculate a sensitivity matrix for inertia for your chosen matrix, and a made-up population vector.
2. Choose a different population vector. Make it different to the one you used first: for example, if you had a vector that had lots of adults in it, now choose one that has lots of juveniles. Compare the sensitivity matrices. How different are they? Which entries have a negative impact on inertia?
3. Calculate sensitivity matrices for the bounds on inertia for your chosen matrix.
4. Plot the transfer functions for all the matrix entries for your first vector. How nonlinear are they?
5. Now do the same for your second vector. Do the transfer functions look the same for both?
6. Plot the transfer functions for both the upper and the lower bounds on inertia. Are there any kinks in the lines where there's a switch in the stage-biased vector that achieves the bounds?

7. Rinse and repeat 1-6 with different matrices. Do you come across any problems or errors? What causes them? Can they be fixed?

Extras

Transfer function analysis is quite flexible. You don't have to calculate transfer functions to single matrix entries: you can perturb multiple matrix entries at once, you can trade off perturbations of one matrix entry against another, you can make perturbations of one matrix entry proportional to that of another, and so on. It's also possible to calculate sensitivites of these more complex perturbations, for example to compare sensitivities of specific management plans. If you're interested in these, then the functions tfa_inertia and tfs_inertia may be worth taking a look at.

Details on how transfer function analysis of inertia works, including the math, can be found in Stott et al. 2012 Methods Ecol Evol, 3, 673-684.