Produce ternary plots of elasticities

6 Oct 2015

This script produces a ternary plot a la Silvertown & Franco (1993) with various life history traits such as mean life expectancy, population growth rate or reactivity as the "fourth" dimension. We will use Caswell (2001) formulation of mean life expectancy from the fundamental matrix (`N`), and the packages `popdemo` for the reactivity:

```
require(popdemo)
```

```
> Loading required package: popdemo
```

```
meanLifeExpectancy <- function(matU = matU, startLife = 1){
  uDim=dim(matU)[1]
  N = solve(diag(uDim[startLife])-matU)
  eta = colSums(N)[startLife]
  return(eta)
}
```

As an example for the chosen data, subset COMADRE to studies with a matrix dimension >= 3, that represent mean, unmanipulated conditions duration > 3 years, where sexual reproduction has been modeled explicitly, the matrices are split into U, F and C, and there are no issues with stage-specific survival >1.

```
tempMetadata <- subset(comadre$metadata, MatrixDimension >= 3 & MatrixComposite == "Mean" & MatrixTr
```

Use the row names from the subsetted dataframe to subset the matrices.

```
keep <- as.numeric(rownames(tempMetadata))
```

Define the object tempMat as the list object containing matrices in the same order that their metadata appears in tempMetadata.

```
tempMat <- comadre$mat[keep]
```

These matrices can now be analyzed by applying functions in a loop, or by using lapply.

To calculate elasticities, population growth rate, reactivity and mean life expectancy for the subset matrices, first crate a bummy variable to accommodate the output

```
output <- data.frame(species= rep(NA, length(tempMat)),
                     lambdas = rep(NA, length(tempMat)),
                     eta = rep(NA, length(tempMat)),
                     react = rep(NA, length(tempMat)),
                     EStasis = rep(NA, length(tempMat)),
                     EProgression = rep(NA, length(tempMat)),
                     ERetrogression = rep(NA, length(tempMat)),
                     EFecundity = rep(NA, length(tempMat)),
                     EClonality = rep(NA, length(tempMat)))
```

Use the following function to calculate element-level perturbations:

```r
matrixElementPerturbation <- function(matU, matF, matC=NULL,pert=0.001){
  #Function to calculate matrix element level sensitivities and elasticities

  matA=matU+matF+matC
  aDim=dim(matA)[1]
  fakeA=matA
  sensA=elasA=matrix(NA,aDim,aDim)
  lambda=Re(eigen(matA)$values[1])

  propU=matU/matA
    propU[is.nan(propU)]=NA
    propProg=propRetrog=propU
    propProg[upper.tri(propU,diag=T)]=NA
    propRetrog[lower.tri(propU,diag=T)]=NA
    propStasis=matrix(diag(aDim)*diag(propU),aDim,aDim)
  propF=matF/matA
    propF[is.nan(propF)]=NA
  propC=matC/matA
    propC[is.nan(propC)]=NA

  for (i in 1:aDim){
    for (j in 1:aDim){
       fakeA=matA
       fakeA[i,j]=fakeA[i,j]+pert
       lambdaPert=eigen(fakeA)$values[1]
       sensA[i,j]=(lambda-lambdaPert)/(matA[i,j]-fakeA[i,j])
    }
  }

  sensA=Re(sensA)
  elasA=sensA*matA/lambda

  out = data.frame("SStasis"=NA,"SProgression"=NA,"SRetrogression"=NA,"SFecundity"=NA,"SClonality"=N
                   "EStasis"=NA,"EProgression"=NA,"ERetrogression"=NA,"EFecundity"=NA,"EClonality"=NA

    out$SStasis=sum(sensA*propStasis,na.rm=T)
    out$SRetrogression=sum(sensA*propRetrog,na.rm=T)
    out$SProgression=sum(sensA*propProg,na.rm=T)
    out$SFecundity=sum(sensA*propF,na.rm=T)
    out$SClonality=sum(sensA*propC,na.rm=T)
    out$EStasis=sum(elasA*propStasis,na.rm=T)
    out$EProgression=sum(elasA*propProg,na.rm=T)
    out$ERetrogression=sum(elasA*propRetrog,na.rm=T)
    out$EFecundity=sum(elasA*propF,na.rm=T)
    out$EClonality=sum(elasA*propC,na.rm=T)

  return(out)
}
```

Loop to examine each matrix:

```r
for (i in 1:length(tempMat)){
 tryCatch({
    matA=tempMat[[i]]$matA
    matU=tempMat[[i]]$matU
    matF=tempMat[[i]]$matF
    matC=tempMat[[i]]$matC
    output$species[i] <- tempMetadata$SpeciesAuthor[i]
    output$lambdas[i] <- max(Re(eigen(matA)$value))
    output$eta[i] = meanLifeExpectancy(matU=matU,startLife=1)
    output$react[i] <- reactivity(matA)
    output[i,c("EStasis","EProgression","ERetrogression","EFecundity","EClonality")]=matrixElementPe
      }, error = function(e){})
}
```

Group elasticities of population growth rate to various demographic processes into three main axes:

```
output$S=output$EStasis+output$ERetrogression
output$G=output$EProgression
output$R=output$EFecundity+output$EClonality
```

Scale to 1 the coordinates of each point - this is necessary due to possible rounding issues, although note that the function to create the ternary plot below can do this automatically with the argument 'scale'.

```
output$S=output$S/rowSums(output[,c("S","G","R")])
output$G=output$G/rowSums(output[,c("S","G","R")])
output$R=output$R/rowSums(output[,c("S","G","R")])
```

Eliminate the couple of MPMs where the code did not run correctly, as it produced all NAs:

```
output=output[-which(is.na(output$eta)),]
```

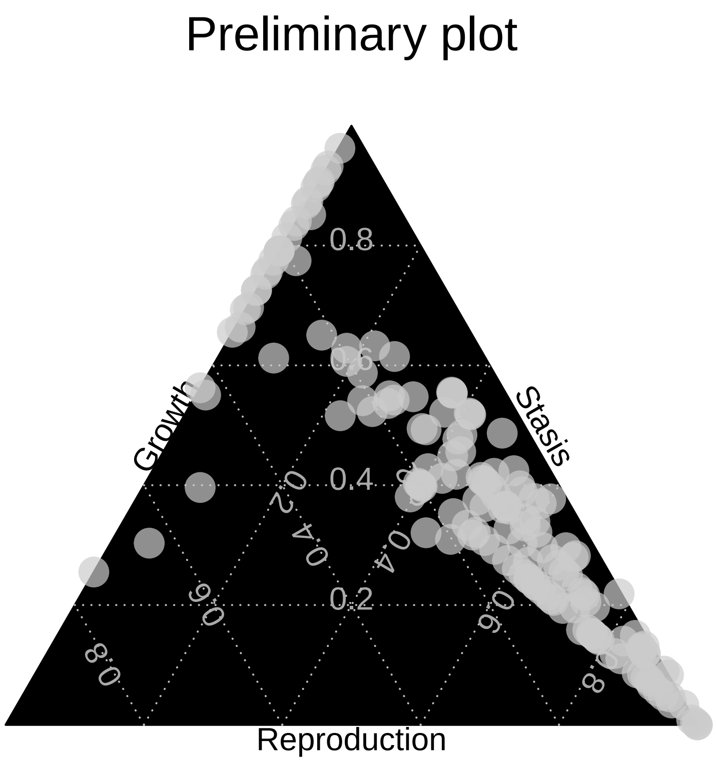
Plot the locations of the chosen matrices in a preliminary ternary plot, for which the following libraries will also be necessary

```
library(vcd)
```

```
> Loading required package: grid
```

```
library(scales)
```

```
ternaryplot(output[,c("R","S","G")],scale=1,col=alpha("gray80",0.7),bg="black", dimnames=c("Stasis",
```



Color-code the points in the ternary plot according to range of lambda, eta and reactivity, respectively, on different plots:

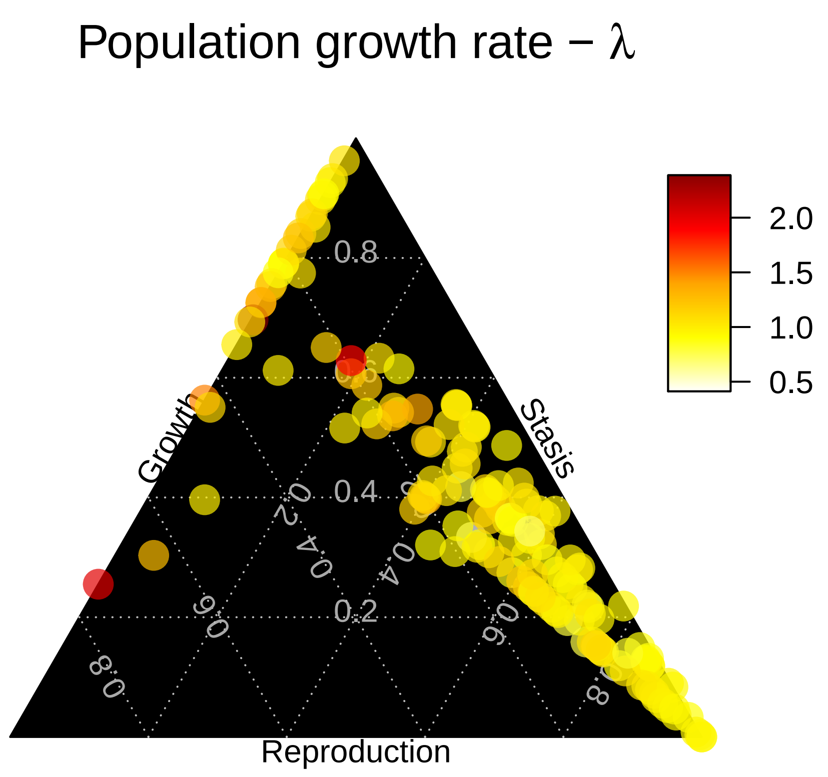
```r
lambdaData=output[which(log(output$lambdas)<=2),]
x_norm=log(lambdaData$lambda)
x_norm = (lambdaData$lambda - min(lambdaData$lambda)) / (max(lambdaData$lambda) - min(lambdaData$lam
col_fun <- colorRamp(c("white","yellow","orange","red","dark red"))
rgb_cols <- col_fun(x_norm)
colsLambda <- rgb(rgb_cols, maxColorValue = 256)

etaData=output
etaData$etalog=log(etaData$eta)
x_norm = (etaData$etalog - min(etaData$etalog)) / (max(etaData$etalog) - min(etaData$etalog))
rgb_cols <- col_fun(x_norm)
colsEta <- rgb(rgb_cols, maxColorValue = 256)

reactData=output[which(log(output$react)<=5),]
reactData$reactlog=log(reactData$react)
x_norm = (output$react - min(output$react)) / (max(output$react) - min(output$react))
rgb_cols <- col_fun(x_norm)
colsReact <- rgb(rgb_cols, maxColorValue = 256)
```

And next plot all three ternary plots. The legend will need the library `fields`

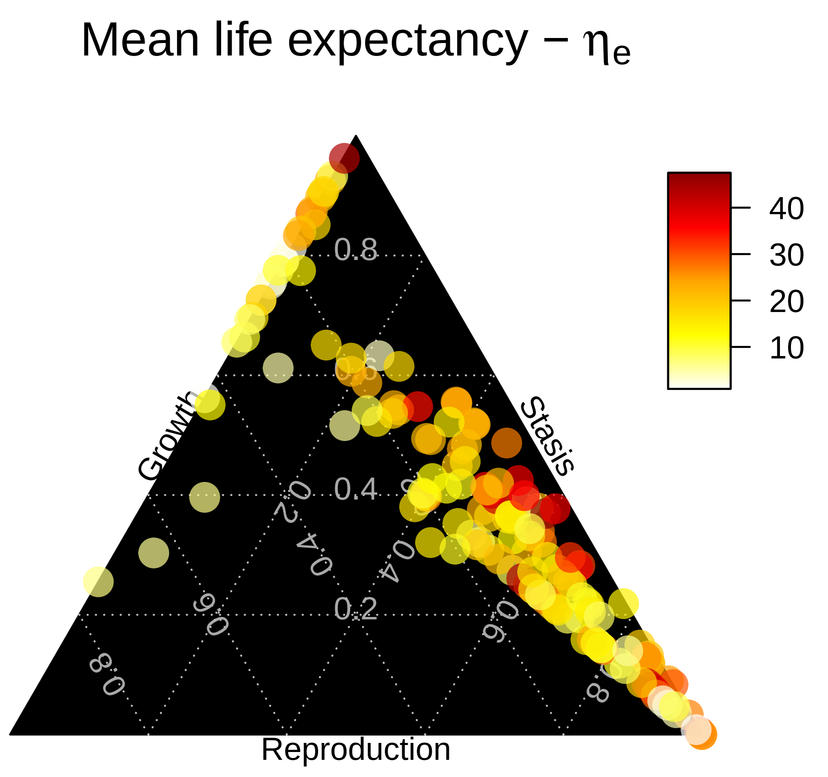
```r
library(fields)
```

```
> Loading required package: spam
> Spam version 1.0-1 (2014-09-09) is loaded.
> Type 'help( Spam)' or 'demo( spam)' for a short introduction
> and overview of this package.
> Help for individual functions is also obtained by adding the
> suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
>
> Attaching package: 'spam'
>
> The following objects are masked from 'package:base':
>
>     backsolve, forwardsolve
>
> Loading required package: maps
```

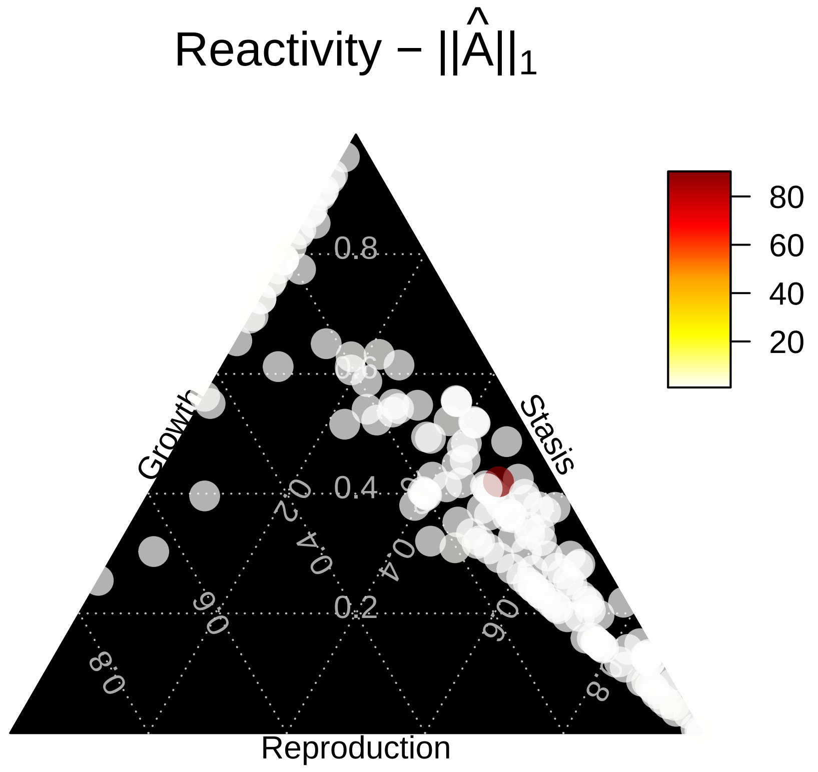
```r
  zr <- range(c(lambdaData$lambda,na.rm=T))
  colCode <- colorRampPalette(c("white","yellow","orange","red","dark red"))(n = 999)
  image.plot(legend.only=TRUE, zlim= zr, col=colCode, smallplot=c(.75,.8, .5,.75),cex.axis=0.2)
  ternaryplot(lambdaData[,c("R","S","G")],scale=1,col=alpha(colsLambda,0.7),bg="black", newpage=F, d
```



```r
zr <- range(c(etaData$eta,na.rm=T))
  colCode <- colorRampPalette(c("white","yellow","orange","red","dark red"))(n = 999)
  image.plot(legend.only=TRUE, zlim= zr, col=colCode, smallplot=c(.75,.8, .5,.75),cex.axis=0.2)
  ternaryplot(etaData[,c("R","S","G")],scale=1,col=alpha(colsEta,0.7),bg="black", newpage=F, dimname
```



```r
  zr <- range(c(reactData$react,na.rm=T))
  colCode <- colorRampPalette(c("white","yellow","orange","red","dark red"))(n = 999)
  image.plot(legend.only=TRUE, zlim= zr, col=colCode, smallplot=c(.75,.8, .5,.75),cex.axis=0.2)
  ternaryplot(reactData[,c("R","S","G")],scale=1,col=alpha(colsReact,0.7),bg="black", newpage=F, dim
```



Reactivity − $||\hat{A}||_1$
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
0.4
0.6
0.8
Growth
Stasis
Reproduction
80
60
40
20