Produce ternary plots of elasticities

2016-06-18

Here we produce 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 package `popdemo` for the reactivity. Other packages we need include `Mage`, `fields`, `vcd` and `scales`:

```
library(popdemo)
library(fields)
library(Mage)
library(vcd)
library(scales)
```

This is the function to calculate mean life expectancy from Caswell (2001):

```
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.

```
x <- subsetDB(comadre, MatrixDimension >= 3
                       & MatrixComposite == "Mean" & MatrixTreatment == "Unmanipulated"
                       & StudyDuration > 3 & MatrixFec == "Yes"
                       & MatrixSplit == "Divided" & SurvivalIssue < 1)
```

This object (`x`) is now a copy of the database that contains ONLY the matrices of interest.

We can ask how many matrices this is by looking at the size of the metadata part.

```
nrow(x$metadata)
```

```
> [1] 163
```

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 create an empty `data.frame` to accommodate the output:

```
output <- data.frame(species= rep(NA, nrow(x$metadata)),
                     lambdas = rep(NA, nrow(x$metadata)),
                     eta = rep(NA, nrow(x$metadata)),
                     react = rep(NA, nrow(x$metadata)),
                     EStasis = rep(NA, nrow(x$metadata)),
                     EProgression = rep(NA, nrow(x$metadata)),
                     ERetrogression = rep(NA, nrow(x$metadata)),
                     EFecundity = rep(NA, nrow(x$metadata)),
                     EClonality = rep(NA, nrow(x$metadata)))
```

We will use the following function to calculate element-level perturbations:

```
matrixElementPerturbation <- function(matU, matF, matC=NULL,pert=0.001){
  matA <- matU + matF + matC
  aDim <- nrow(matA)
  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 = TRUE)] <- NA
  propRetrog[lower.tri(propU, diag = TRUE)] <- 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" = NA, "EStasis" = NA,
                    "EProgression" = NA, "ERetrogression"=NA,
                    "EFecundity" = NA, "EClonality" = NA)

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

  return(out)
}
```

Now we can use a loop to examine each matrix:

```
for (i in 1:nrow(x$metadata)){
 tryCatch({
    matA <- x$mat[[i]]$matA
    matU <- x$mat[[i]]$matU
    matF <- x$mat[[i]]$matF
```

```
    matC <- x$mat[[i]]$matC
    output$species[i] <- x$metadata$SpeciesAuthor[i]
    output$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")] <-
      matrixElementPerturbation(matU = matU, matF = matF, matC = matC)[6:10]
      }, 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`.

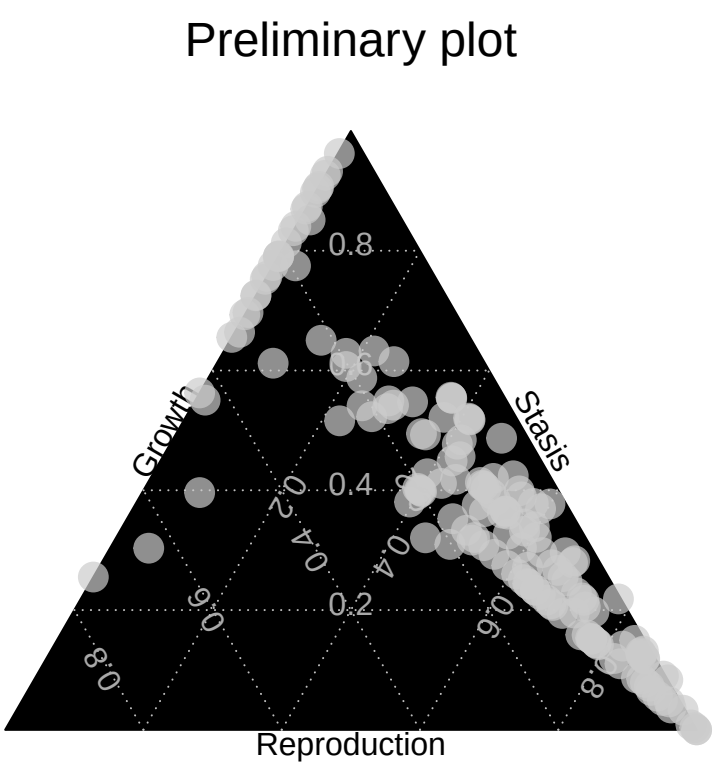
```
output$S <- output$S / rowSums(output[, c("S","G","R")])
output$G <- output$G / rowSums(output[, c("S","G","R")])
output$R <- output$R / rowSums(output[, c("S","G","R")])
```

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.

```
ternaryplot(output[,c("R","S","G")],scale=1,col=alpha("gray80",0.7),bg="black",
            dimnames=c("Stasis","Growth","Reproduction"),dimnames_position="edge",
            main="Preliminary plot")
```



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

```
lambdaData <- output[which(log(output$lambdas)<=2),]
x_norm <- log(lambdaData$lambda)
minL <- min(lambdaData$lambda)
maxL <- max(lambdaData$lambda)
x_norm <- (lambdaData$lambda-minL) / (maxL-minL)
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)
minE <- min(etaData$etalog)
maxE <- max(etaData$etalog)
x_norm <- (etaData$etalog - minE) / (maxE - minE)
rgb_cols <- col_fun(x_norm)
colsEta <- rgb(rgb_cols, maxColorValue = 256)

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

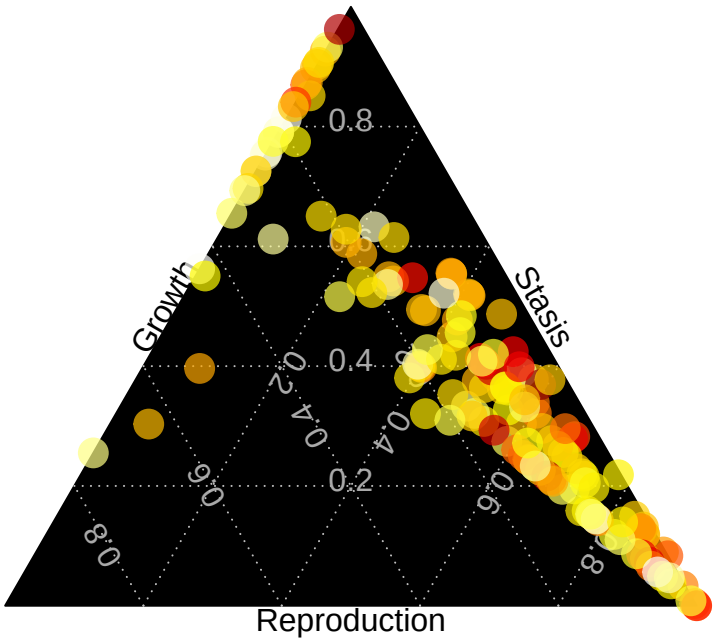
And next plot all three ternary plots.

We can use the same colour palette for each of the plots:

```
colCode <- colorRampPalette(c("white", "yellow", "orange", "red", "dark red"))(n = 999)
```

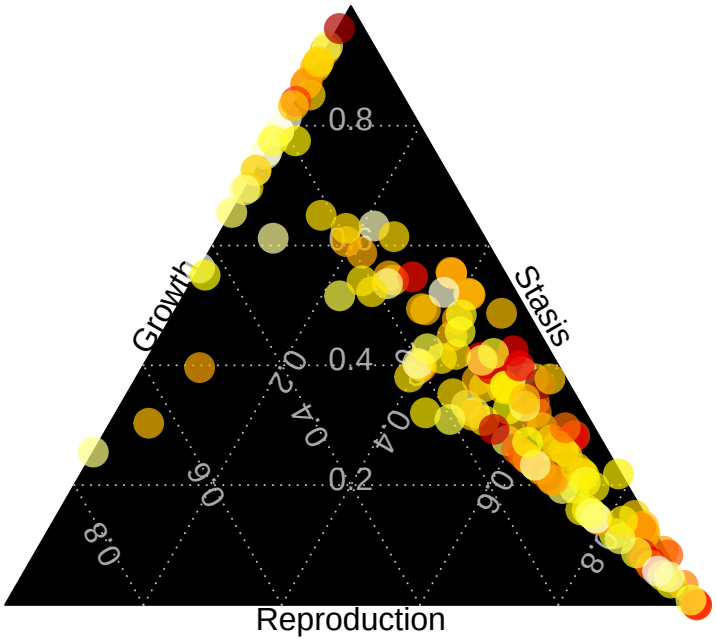
```
zr <- range(c(lambdaData$lambda, na.rm = TRUE))
ternaryplot(lambdaData[, c("R", "S", "G")], scale = 1, col = alpha(colsEta, 0.7),
            bg = "black", dimnames = c("Stasis", "Growth", "Reproduction"),
            dimnames_position = "edge",
            main = expression(paste("Population growth rate - ", lambda)))
```

Population growth rate − λ



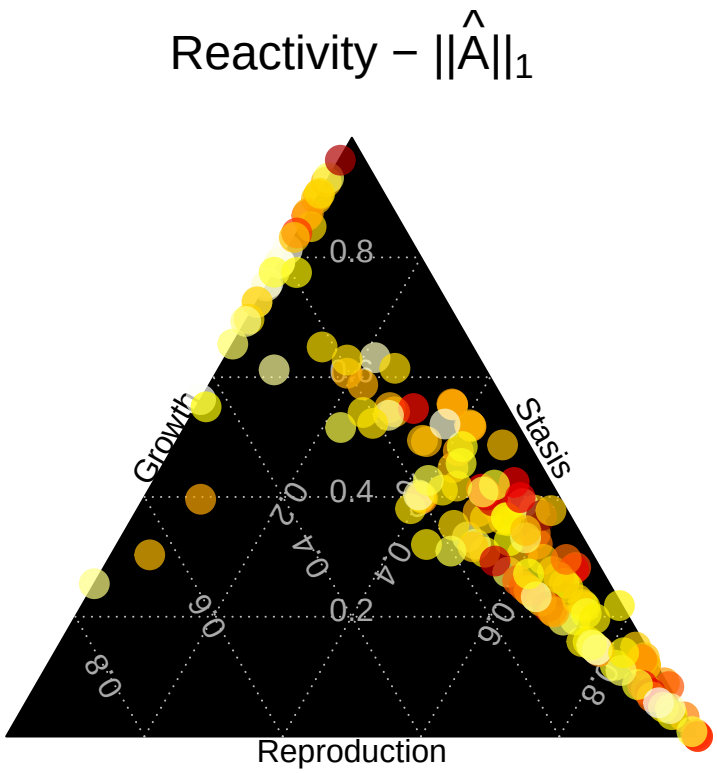

```
zr <- range(c(etaData$eta,na.rm=T))
ternaryplot(lambdaData[, c("R", "S", "G")], scale = 1, col = alpha(colsEta, 0.7),
            bg = "black", dimnames = c("Stasis", "Growth", "Reproduction"),
            dimnames_position = "edge",
            main=expression(paste("Mean life expectancy - ", eta["e"])))
```

Mean life expectancy − η_e




```
zr <- range(c(reactData$react,na.rm=T))
ternaryplot(lambdaData[, c("R", "S", "G")], scale = 1, col = alpha(colsEta, 0.7),
            bg = "black", dimnames = c("Stasis", "Growth", "Reproduction"),
```

```
            dimnames_position = "edge",
            main=expression(paste("Reactivity - ||", hat(A),"||"[1])))
```



One might add the color scale legend using the following code to add to the same plot:

```
image.plot(legend.only = TRUE, zlim = zr, col = colCode,
           smallplot = c(.75, .8, .5, .75), cex.axis=0.2)
```