Geographic distribution

6 Oct 2015

This example produces on a world map the viability (population growth rate $\lambda > 1$, $\lambda = 1$, $\lambda < 1$) of a subset of studied populations given some selection criteria, and color-codes the location of each population according to the value of λ.

First, subset mean matrices for all Carnivora in the wild in the Northern hemisphere, with no issues for survival >1, for which matrices have been split into $\mathbf{A} = \mathbf{U} + \mathbf{F} + \mathbf{C}$, and for which reproduction was explicitly modeled.

Load the data:

```
load("COMADRE_v.1.0.0.RData")
```

```
tempMetadata <- subset(comadre$metadata,
                       MatrixComposite == "Mean" &
                         Order == "Carnivora" &
                         MatrixCaptivity == "W" &
                         LatNS == "N" &
                         SurvivalIssue < 1 &
                         MatrixSplit == "Divided" &
                         MatrixFec == "Yes")
```

The row names from the subsetted dataframe can now be used to subset the entire `comadre` database using the function `subsetDB` which is available via GitHub (using `source_url` from `devtools` to download it).

```
library(devtools)
source_url("https://raw.githubusercontent.com/jonesor/compadreDB/master/Functions/subsetDB.R")
```

```
## SHA-1 hash of file is 1b6085d7b556cf550c8425c76f6078a8e2f59452
```

§

```
id <- as.numeric(rownames(tempMetadata))
x<-subsetDB(comadre,id)
```

The object `x` is now a version of the `comadre` database object that contains only the matrices that match the search criteria. To calculate population growth rate for the subset matrices, we can first create an empty `data.frame` to accommodate the output:

```
output <- data.frame(lambdas = rep(NA, length(x$mat)))
```

Now create dummy variables to convert geographic information to be plotted on the map (below):

```
x$metadata$LAT <- NA
tempMetadata$LON <- NA

for(i in 1:nrow(x$metadata)){
  if(x$metadata$LatNS[i] == "S") {
    x$metadata$LatDeg[i] <- -x$metadata$LatDeg[i]
  }
  if(x$metadata$LonWE[i] == "W") {
    x$metadata$LonDeg[i] <- -x$metadata$LonDeg[i]
```

```
  }
  x$metadata$LAT[i] <- x$metadata$LatDeg[i] +
    x$metadata$LatMin[i]/60 + x$metadata$LatSec[i]/3600
  x$metadata$LON[i] <- x$metadata$LonDeg[i] +
    x$metadata$LonMin[i]/60 +
    x$metadata$LonSec[i]/3600
}
```

Create an empty variable to accommodate output from lambda calculations:

```
x$metadata$lambdas <- NA
```

Then, create a `for` loop to examine each matrix in turn. Here it may be advisable to use the function `tryCatch` as a wrapper to cope with the situation if/when the function in the loop fails:

```
for (i in 1:length(x$mat)){
  tryCatch({
    x$metadata$lambdas[i] <- Re(eigen(x$mat[[i]]$matA)$value)[1]
  }, error = function(e){})
}
```

Now we can create a vector of color hex codes that can be applied according to the estimate of λ. THis is done using the `colorRampPalette` function to go from green for high values of λ, to red for low values of λ. Here `paste` is used to append a value of `90` to the hex codes to allow transparency for aesthetic reasons.

```
rampfunc <- colorRampPalette(c("green", "red"))
colVect <- rampfunc(100)
colVect <- paste(colVect,"90",sep="")
s1 <- seq(min(x$metadata$lambdas,na.rm=TRUE),max( x$metadata$lambdas,na.rm=TRUE),length.out = 100)
```

First, load the `maps` package (and install it if necessary). Then plot the world map and overlay the points from our data, color coded by value of λ. In this case, the points are jittered slightly to improve visibility of nearby populations.

```
library(maps)
map("world", col = "gray", fill = TRUE, bg = "light blue",
    xlim=c(-175,176),ylim=c(-60, 85), border = "white")

points(jitter(x$metadata$LON,amount=0.6),jitter(x$metadata$LAT,amount = 0.6),col = colVect[findInter
```