

# Geographic distribution

2016-08-12

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  $\lambda$ .

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.2.0.0.RData")

x<-subsetDB(comadre, MatrixComposite == "Mean" &
            Order == "Carnivora" &
            MatrixCaptivity == "W" &
            Lat > 0 &
            SurvivalIssue < 1 &
            MatrixSplit == "Divided" &
            MatrixFec == "Yes")
```

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)))
```

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  $\lambda$ . This is done using the `colorRampPalette` function to go from green for high values of  $\lambda$ , to red for low values of  $\lambda$ . 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  $\lambda$ . In this case, the points are jittered slightly to improve visibility of nearby populations.

```
library(maps)
```

```
>  
> # maps v3.1: updated 'world': all lakes moved to separate new #  
> # 'lakes' database. Type '?world' or 'news(package="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, 0.6), jitter(x$metadata$Lat, 0.6),  
       col = colVect[findInterval(x$metadata$lambda_s, s1)], cex = 2, pch = 16)
```

