Reponse Function Lab

By Vinícius Manvailer

For a visual explanation go to the video of this class here

1. INTRODUCTION

Response Function analysis is a multivariate technique based on Principal Components Analysis that was introduced in the field of dendrochronology by Fritz in 1971. This technique was implemented to overcome the issues associated with dealing with a great number of predictor variables (monthly precipitation (PPT) and temperature (TMP) data) in correlation analysis. With many variables, multiple correlation tests are necessary which in turn increases your chances of type-I error (finding a false positive). Although this confidence intervals can be adjusted for multiple testing this often reduces your power of detecting climate signals in the analysis.

Another issue related to simple correlation analysis is dealing with time series data. Auto-correlation between adjacent months can often hinder correct interpretation of climate patterns. Furthermore, precipitation and temperature are variables that are correlated as well.

Although simple correlation can still be informative, all these issues make it hard to untangle the effect of climate on tree growth. Thus, Fritz addressed those issues by doing a regression analysis of the TRW on the principal components of the climate data. So instead of performing a multiple regression on TRW to PPT of January, February and so on, we regress TRW on PC1, PC2 and etc.

The advantage of doing this is that Principal Components are, by definition, not correlated to each other thus solving the issue related to auto-correlation and inter-correlation between temperature and precipitation.

2. DOING YOUR OWN RESPONSE FUNCTION ANALYSIS

Start by importing your tree ring and climate data and standardizing it using the `scale()` function.

Tree ring data:

```
tr <- read.csv("chr_spruce.csv")

tr_scaled <- scale(tr$mucstd)

tr_scaled <- as.data.frame(tr_scaled)
```

Climate data:

```
clim <- read.csv("clim_spruce.csv")

clim_scaled <- scale(clim[2:25])

clim_scaled <- as.data.frame(clim_scaled)
```

Now run a principal component analysis on the scaled climate data

```
pca_clim <- princomp(clim_scaled)
```

Ok now we will regress the tree ring data on the principal components, but we do not need all of them so we will select the ones that accounts for the largest variation in climate. *Note: there are different ways to do this, but we will use the PVP criterion (CITATION) which is the most used.*

To do this we will calculate the product of eigenvalues until they get below 1. So, eigen1*eigen2, then eigen1*eigen2*eigen3 and so on. Start with the first 15 and keep adding eigenvalues until the results drops below one.

Our eigenvalues are stored in:

```
pca_clim[["sdev"]]
```

Now, extract the first 15 and do the product. What is the result? How many do you need?

```
prod(pca_clim[["sdev"]][1:15])
```

Found it? Awesome! Now we know which PCs are most important. The rest is probably random variability. We will now regress the tree ring data on the principal components that matter.

First get the principal components scores.

```
scores <- as.data.frame(pca_clim$scores)
```

Those scores represent almost all our climate data – minus some noise we removed – but, unlike our original data, each column (PC) is a variable completely independent from each other. This is important part of this technique. Now get only the columns (scores) related to the PCs that you chose in the previous step and let's regress the standardized tree ring data on those scores.

```
lm_rf <- lm(tr_scale$V1 ~ Comp.1 + Comp.2 + Comp.3 + Comp.4 + Comp.5 + Comp.6
+ Comp.7 + Comp.8 + Comp.9 + Comp.10 + Comp.11 + Comp.12 + Comp.13 + Comp.14 +
Comp.15 + …)
```

Great! You now have a linear model that uses principal components instead of original variables. Usually, we would look at the coefficients for each variable and calculate the R2 to understand how much variation is explain by each variable, but since principal components do not represent a direct measure of the real world the coefficients don't mean much to us. So, to make this meaningful we will get those coefficients and multiply by the original principal components loadings. This part is not at all intuitive, but it will yield our response function coefficients which are interpretable. Basically, we are trying to revert these principal components coefficients back to coefficients of our original variables.

Extract the coefficients from the linear model and transform it into a data frame:

```
coef <- as.data.frame(lm_rf$coefficients)

coef2 <- coef[,1] #extract just the coefficients
```

Now, because we are about to perform a matrix multiplication in both of our matrices the number of rows in one must match the number of columns on the other one. They are different because we removed some components in the previous step. So let's just add some zeros at the end.

```
coef3 <- as.matrix (c(coef2, c(0,0,0,0)))
```

Now multiple the loadings (original variable weights) by the coefficients we calculated in our linear model (I think we can call these our "new weights"). We basically want to find a single set of weights for our original variables that will best describe the variability in our tree ring data.

```
loadings <- pca_clim$loadings

rf_coef <- as.data.frame (loadings%*%coef3)
```

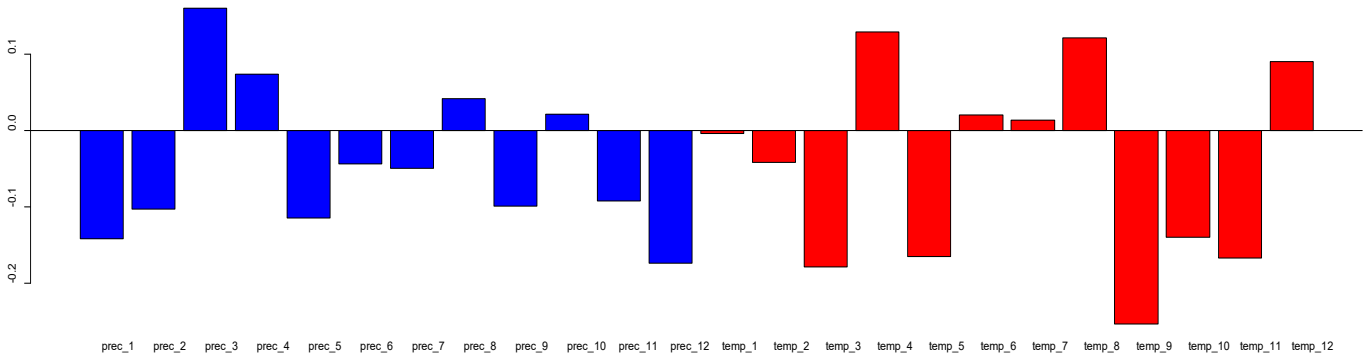
Let's retrieve the variable names:

```
rf_coef <- rownames_to_column(rf_coef, "varnames")
```

Congratulations, you are done! You calculated your own response coefficients! Now let's plot our work.

```
barplot(rf_coef$V1, col = c(rep("blue", 12), rep("red", 12)) , names.arg = rf_coef$varnames)

abline(h = 0)
```



The last step would be to calculate the confidence intervals for our weight estimates by repeating this 1000 times using cross-validation. We could do this manually but, luckily, we have a nice package that will do this for us.

3. RESPONSE FUNCTION ANALYSIS WITH THE *TREECLIM* PACKAGE.

```
library(treeclim)
```

Let's import our data again

```
tr <- read.csv("chr_spruce.csv")

clim <- read.csv("clim_spruce.csv")
```

This packages is rather straight forward and does everything simply by calling the function *dcc()*. The only thing you must check is the format of the data. The tree ring data needs to have the years as row names. This is easily done with the command below.

```
tr <- column_to_rownames(tr, "year")
```

Now let's adjust the climate data, we will use *gather()* to transform from wide to long format.

```
library(tidyr)
```

Split the dataset in two according to the variables.

```
a1 <- muc_clim_spread[1:13]

a2 <- muc_clim_spread[,c(1,14:25)]
```

Use *gather()* to put it in the long format. Then use *separate()* to extract the numbers regarding the months.

```
a1 <- gather(a1, key = "month", value = "temp", 2:13)

a1 <- separate(a1, col = 2, into = c("d", "month")) %>% select(year, month, temp)

a2 <- gather(a2, key = "month", value = "prec", 2:13)

a2 <- separate(a2, col = 2, into = c("d", "month")) %>% select(year, month, prec)
```

Merge both datasets and make the month variable an integer so we can properly order this dataset.

```
a <- merge(a1, a2)

a$month <- as.integer(a$month)

clim_long <- arrange(a, year, month)
```

Done! No you are ready to go. This package does everything for you, you just have to call the function *dcc()*.

```
rf_treeclim <- dcc(tr, clim_long)
```

To visualize your coefficients simply call.

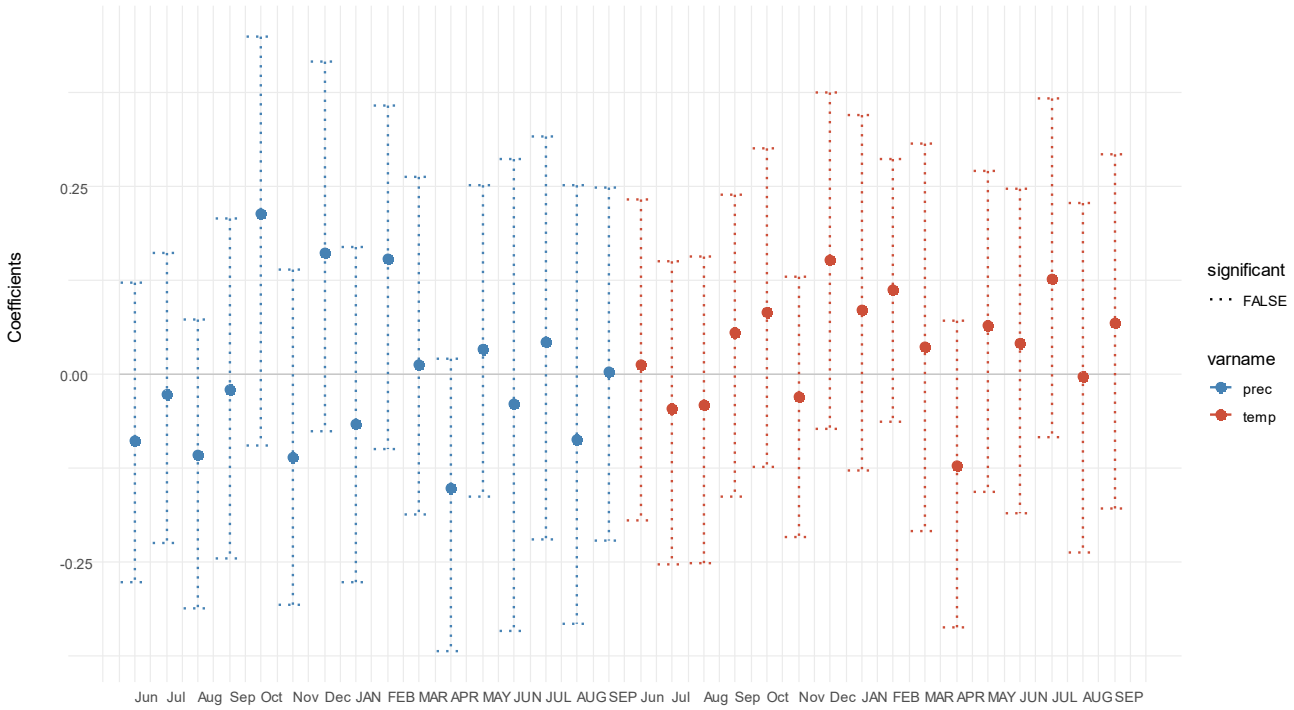
```
rf_treeclim$coef
```

You can extract the information from there and make your own custom plots. However, the function already uses ggplot style to plot everything for you. To plot the coefficients, just call the plot command.

```
plot(rf_treeclim)
```

To adjust colors, you can use the ggplot syntax, just try:

```
plot(rf_treeclim) + scale_color_manual(values = c("steelblue", "tomato3"))
```



There you go!