

R tutorial: basic use of package ‘rioja’ for Modern Analogue reconstructions

23 May 2016

- Aim: To provide training in the use of R and package rioja for analogue-based palaeoenvironmental reconstructions.
- 1. Introduction
 - 1.1 Your first R session
 - 1.2 Closing RStudio
 - 1.3 Using the script editor
- 2. Importing data into R
 - 2.1 Importing the modern pollen data
 - 2.2 Importing the fossil pollen data
 - 2.3 View the pollen data
- 3. Environmental reconstruction using MAT
 - 3.1 Select the variables to be used with the Modern Analog method.
 - 3.2 Apply the Modern Analogue method using the package “rioja”.
 - 3.3 Predict T_ANN using fossil data
 - 3.4 How good are the analogues?
 - 3.5 Where are the analogues located?
- 4. Tree cover reconstructions
- 5. References

Aim: To provide training in the use of R and package rioja for analogue-based palaeoenvironmental reconstructions.

1. Introduction

R is a language and environment for statistical computing and graphics (<http://www.r-project.org/> (<http://www.r-project.org/>)). It has a number of features that make it a very effective tool for analysing environmental and ecological data, including a wide range of statistical and graphical techniques, easy addition of new methods, excellent facilities for data handling and manipulation, publication quality graphics, and a large, rapidly-expanding and helpful community of users. Versions are available for Windows, MacOSX, and Linux and it is free.

Recently R has been enhanced by the development of RStudio. RStudio is an IDE (Integrated Development Environment) for R available for Windows and MacOSX, and Linux. It provides a single interface with all the functionality for running and using R effectively, including a multi-file script editor, graphics windows, help system, workspace browser, and html / presentation viewer.

You can download and install R and RStudio from the following websites.

R: [<http://cran.r-project.org/> (<http://cran.r-project.org/>)]

RStudio: [<http://www.rstudio.com/> (<http://www.rstudio.com/>)]

RStudio is an interface on top of the base R installation so you should install R first then RStudio. Once installed you only need to run RStudio (this will start R automatically). You do not need to run R separately.

In this handout commands that you should enter at the R console or into the script editor are formatted like this:

```
plot(x, y)
```

Output from R is preceded by ##, for example:

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

Function names in the text are shown in monospace font, (e.g. `plot(x, y)`).

1.1 Your first R session

Start RStudio and click in the R Console (bottom left pane). The R prompt in the console (“>”) is waiting for a command. Type the following at the prompt:

```
rnorm(20)
```

```
## [1] 0.84340037 -2.09294148 0.06502729 1.00227077
## [5] 1.44425499 0.76910077 0.39779268 -1.11466388
## [9] 0.84271368 0.01566203 2.02221748 1.75998077
## [13] -0.52499927 -0.11382235 -1.20591627 0.48726253
## [17] 0.18014635 1.94649957 -1.25085246 0.96997508
```

`rnorm` is a **function** that generates random numbers from a normal distribution (with mean=0 and standard deviation=1 by default). 20 is an **argument** to this function and instructs `rnorm` to generate 20 random numbers. Since we have not told R to do anything with the output from `rnorm` it **prints** the 20 random numbers on the screen.

Now type:

```
x <- rnorm(20)
```

The `<-` is an **assignment** operator that says take the output from `rnorm` and assign it to an object called "x". `<-` is equivalent to `=` (equals) in other languages. You can also use `=` in R to make assignments but most texts use the `<-` notation so we will also follow this convention.

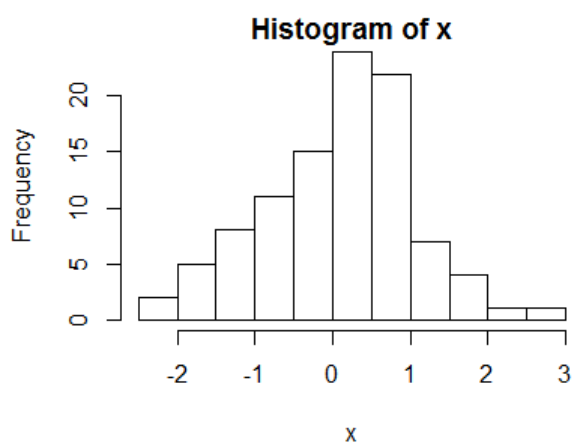
Now type the following at the console:

```
x
```

```
## [1] -0.5185203 0.6416658 1.0277842 -1.1344577 -1.8209771
## [6] -0.9491615 -0.1921532 -1.4207798 0.3201229 0.8531206
## [11] -0.4809296 -1.6787675 1.2625557 -1.1165794 0.5477355
## [16] -1.2610984 -1.0059436 0.5421939 -1.1382905 0.8693772
```

This will display (or more precisely **print**) the contents of x, in this case twenty random numbers. Now enter:

```
x <- rnorm(100)
hist(x)
```



`hist` is a function that plots a histogram, and x is its argument. To get help on a function type `?function_name`, for example:

```
?hist
```

This will load the help text for the `hist` function into the Help window in the lower right pane of RStudio.

The help text probably looks pretty daunting so close it again quickly and let's move on! Now enter:

```
Hist(x)
```

```
## Error in eval(expr, envir, enclos): could not find function "Hist"
```

You have just discovered that R is case sensitive. That is, `Hist` is different from `hist`.

You may have already discovered that if you do make a mistake you can use the up / down arrow keys to scroll back through previous commands, correct and re-run them.

The prompt (`>`) at the console means R is waiting for a command. If you see a `+` (plus) this means you have entered an unfinished expression. That is, R has parsed the command but needs some additional input to make sense of it. For example, enter `rmnorm(50`. You will see a `+` because the expression is not complete - it is missing the closing parenthesis. Now enter `)`. If you get the `+` and can't figure out what is wrong just press the escape key to quit the expression and start again.

1.2 Closing RStudio

When you close RStudio the program will ask if you want to save the workspace. The **workspace** is everything you have created in the current session. You can see a list of these objects and can inspect them under the Environment tab in the top right pane. Answer no and quit. In R (and RStudio) you will save your commands as scripts that read the data, do some analysis and/or create some graphics. There is no need to save the workspace as running the script will re-create all your analyses, usually in a few seconds. There is usually no need to save and reload the workspace unless you have very large datasets or complex analysis that takes a long time to run, or you want to send the workspace file to a colleague. You can turn off the prompt to save the workspace under Tools > Global options > General.

1.3 Using the script editor

You will have realised by now that the main difference between R and most other programs you will have used is that in R the analysis proceeds by entering commands at the console and not by selecting actions from menus and dialog boxes. This does make R harder to learn but it also means it is far more flexible as you can string lists of commands together to perform almost any combination of analyses in one go. It also means that you have a permanent record of your analyses. That is, your work is **reproducible**. You, or someone else, can come back to the script in the future and recreate exactly what you did.

Sequences of saved commands are called **scripts** in R and RStudio has a very nice built-in script editor for creating, modifying and running scripts. In RStudio click the New Document icon on the main toolbar (the left-most icon on the toolbar below the File menu option), and select **R Script**. Type the following into the empty file:

```
x <- rnorm(100)
y <- rnorm(100)
plot(x, y)
```

Notice that the function names and any quoted text are coloured in the editor. The script editor knows about R functions and the colour coding makes writing and understanding R code much easier. Notice also that as you place the cursor next to the parentheses they change highlight. The editor highlights the parenthesis or brace and what it thinks is the matching one. Most errors in R are simple and caused by incorrectly matching parentheses, square brackets or braces. This feature is invaluable for identifying these common errors.

Now save the file to your R **working directory**. What is your R working directory? This is a directory you have created to contain the datasets and R scripts for this tutorial. Now click "Source" button (or Cmd-Shift-S in MacOSX / Ctrl-Shift-S in Windows). This will send the commands to R and produce the plot.

You can build up lists of commands but what if you don't want to run them all? You can comment out a line using the `#` (hash) character. Notice that the colour of commented out lines changes, again making it easier to read. You should also use comments to annotate your code to remind you why you did what you did when you come back to it later. There are also options in the script editors to run the current line (Cmd-Enter in MacOSX / Ctrl-Enter in Windows). To run a block of code simply select it and click Run.

2. Importing data into R

2.1 Importing the modern pollen data

Our training dataset of choice is the European Modern Pollen Database (EMPD), freely available online at http://www.europeanpollendatabase.net/wiki/doku.php?id=empd_database (http://www.europeanpollendatabase.net/wiki/doku.php?id=empd_database).

The EMPD comprises over 4000 surface pollen samples –available as raw pollen counts- resulting in a collection of several thousand taxa, including pollen and Non Pollen Palynomorphs. Such taxonomic richness is not suitable to be used in an analogue based reconstruction, since an excessive diversity between samples can exacerbate the dissimilarities between pollen spectra and hinder the identification of suitable analogues. It is therefore needed to reduce this complexity by grouping different taxa under a broader, more suitable taxonomic level. This process largely depends on specific research questions and can be extremely time consuming. In order to focus on the analogue reconstruction algorithm, we already proceeded with this taxonomic simplification, grouping the EMPD pollen taxa in broader categories and eventually limiting our selection to a total of 45 variables.

As an example, the EMPD entries:

- Carpinus
- Carpinus betulus
- Carpinus betulus-type
- Carpinus-type

have all been grouped under the same variable “Carpinus”. Similarly, all members of the family Umbelliferae, including generic taxa such as “Umbelliferae undifferentiated”, have been gathered under the broad category “Umbelliferae”.

The dataset of modern pollen taxa, expressed as percentages, together with modern climate data is available in the tab-delimited file “EMPD_climate.txt”. We can read this using the function `read.table`. When we import a file we need to tell R where to look for it, that is, we need to specify the folder or path. We can include the full path in the call to `read.table` but if we do this we would have to type the whole path for every file we read. It is easier to set the *working directory* to the folder that contains your data files. `read.table` (and other input/output functions) will then look here for any files if the full path is not given.

Open a new script for the practical. At the top of your script set the working directory to your data directory. For example:

```
setwd("c:\\data\\analogue_workshop")
```

Of course, you should change this to your own working directory! If you have created the folder on your desktop your working directory will be something like this:

```
setwd("C:\\users\\[your_username]\\Desktop\\analogue_workshop\\") ## On PC
setwd("~/Users/[your_username]\\Desktop\\analogue_workshop\\") ## On Mac
```

Note: File names and paths are quoted. These are text strings and should be quoted in R. Note also that we use double backslash (\\) in the path. This is because in a quoted string a single backslash has special meaning (it ‘escapes’ or modifies the following character). Double backslashes resolve to a single backslash when the string is evaluated. We can also use a forward slash to separate folders. Both of the following are correct:

```
setwd("c:\\data\\StatsCourse")
setwd("c:/data/StatsCourse")
```

Pay special attention to this rule - it is the source of many simple but frustrating errors! You can also quote text strings using single or double quotes but cannot mix the two in the same string. For example, “I am a text string” and ‘So am I’ are both correct, but “What about me” will generate an error.

We will swap back and forth between your script and the R prompt. **As a rule of thumb, anything that performs an analysis that you want to keep or come back to goes into a script, and any on-the-fly commands to inspect objects etc. are typed at the console.** This is an important rule so remember it! You should also comment your scripts so you have a record of what your code means. By the time you have finished this practical you should have a commented script file that contains all the code to re-run all the examples and analyses.

Now import the modern pollen counts and climate data:

```
EMPD_climate <- read.table("EMPD_climate.txt", sep="\t", row.names=1, header=TRUE)
```

You can view the data by clicking on the Data name in the RStudio Environment viewer (top right window), or using the function `head()` to print the first 6 lines in the console:

```
head(EMPD_climate)
```

We can also view a list of column or variable names using the function `colnames`:

```
colnames(EMPD_climate)
```

```
## [1] "Abies"          "Acer"           "Alnus"          "Apiaceae"
## [5] "Artemisia"      "Asteraceae"     "Betula"         "Carpinus"
## [9] "Caryophyllaceae" "Castanea"       "Cedrus"         "Chenopodiaceae"
## [13] "Cichorioideae"  "Cistaceae"      "Corylus"        "Cruciferae"
## [17] "Cupressaceae"   "Dipsacaceae"    "Ephedra"        "Ericaceae"
## [21] "Fabaceae"       "Fagus"          "Fraxinus"       "Hedera"
## [25] "Hippophae"      "Ilex"           "Juglans"        "Lamiaceae"
## [29] "Larix_Pseudotsuga" "Olea_Phillyrea" "Ostrya_type"     "Picea"
## [33] "Pinus"          "Pistacia"       "Plantaginaceae" "Quercus"
## [37] "Ranunculaceae"  "Rosaceae"       "Rumex"          "Salix"
## [41] "Tilia"          "Ulmus_Zelkova"  "Urticaceae"     "Vitis"
## [45] "Poaceae"        "P_DJF"          "P_JJA"          "P_ANN"
## [49] "T_DJF"         "T_JJA"          "T_ANN"          "londd"
## [53] "latdd"         "altitude"       "Elevation"
```

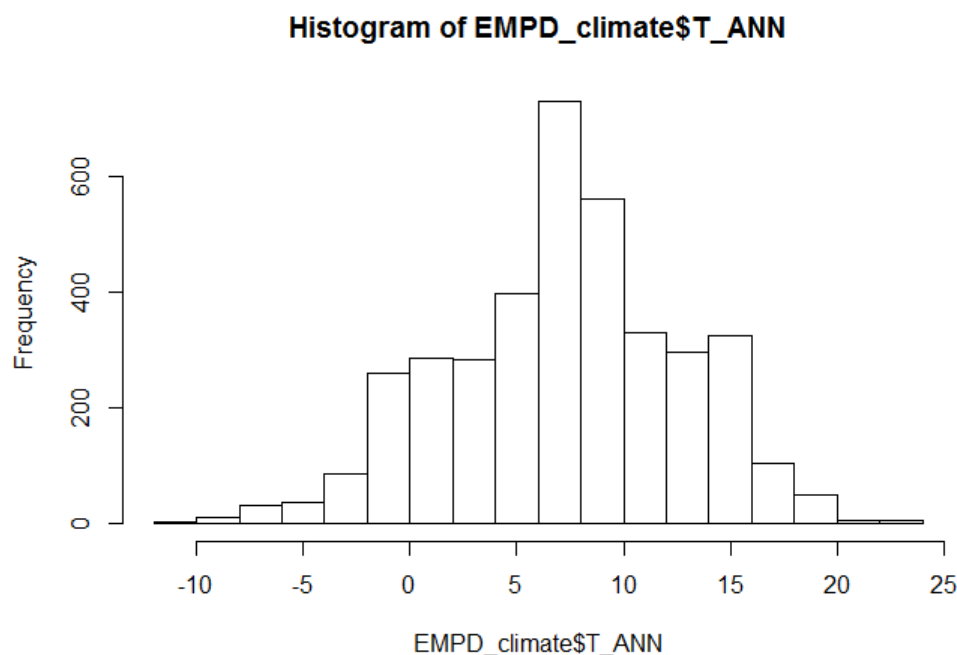
We can also ask how many rows (samples) and columns (variables) our data has:

```
dim(EMPD_climate)
```

```
## [1] 3813 55
```

We can also produce some simple summaries of the climate variables. For example, to produce a histogram of the Mean Annual temperature values for all sites (stored in column T_ANN):

```
hist(EMPD_climate$T_ANN)
```



EMPD_climate is imported into R as a *data frame* containing the pollen counts and climate variables. The data are arranged with variables as columns and sites or samples as rows. The dollar ("") above allows us to reference the column T_ANN within this data frame. We can also reference specific columns using their numeric index or name with square-bracket notation:

```
hist(EMPD_climate[, 51])
```

```
hist(EMPD_climate[, "T_ANN"])
```

2.2 Importing the fossil pollen data

The fossil pollen record for a single site (Lake Trummen, S. Sweden) was selected from the publicly available EPD. The MAT function in the package `rioja` will match pollen taxa between the modern and fossil data using the column names. The order of names in the two files can differ but the variables names (ie. pollen taxa) must be spelt exactly the same (and with the same case) in the two files.

```
fossil <- read.table("Lake_Trummen_pollen_perc.txt", header=TRUE, row.names=1, sep="\t")
```

```
fossil.age <- read.table("Lake_Trummen_agebp.txt", header=TRUE, row.names=1, sep="\t")
```

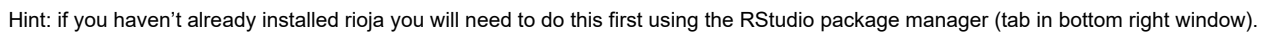
We can produce a simple stratigraphic diagram using the function `strat.plot` in package `rioja`. The pollen data contains a total of 45 taxa. We first remove rare taxa using the following steps:

- ```
mx <- apply(fossil, 2, max)
fossil.sub <- fossil[, mx > 2]
```

```
library(rioja)
```

```
This is rioja 0.9-7
```

```
strat.plot(fossil.sub, yvar=fossil.age$agebp, scale.percent=TRUE, y.rev=TRUE)
```



```
strat.plot(fossil.sub, yvar=fossil.age$agebp, scale.percent=TRUE, y.rev=TRUE, plot.poly=TRUE, plot.bar="Full", col.poly.line=NA, exag=TRUE, col.exag="auto", col.poly="darkgreen")
```

### 3. Environmental reconstruction using MAT

#### 3.1 Select the variables to be used with the Modern Analog method.

Three datasets are required to perform an analogue-based reconstruction using 'rioja': - a table containing modern pollen samples (contained in R object EMPD\_climate) - the environmental variable for every fossil sample (also contained in R object EMPD\_climate) - a table containing fossil samples (R object fossil)

While the table "fossil" contains only taxa percentages, the table EMPD\_climate contains both percentages and modern climate data. rioja requires that the pollen and climate data are in different tables so we extract these data from the EMPD\_climate table. The first 45 columns contain pollen taxa so we using column square-bracket indexing to extract these. We will reconstruct mean annual temperature (in column T\_ANN):

```
train <- EMPD_climate[, 1:45]
T_ANN <- EMPD_climate[, "T_ANN"]
```

Before proceeding with the Modern Analogue algorithm, pollen data can undergo additional refinements. As an example, percentages lower than a defined threshold can be interpreted as noise (e.g. due to long distance transport or identification uncertainties) and removed from the pollen spectra. In the following commands, we use a threshold value of 0.5% as suggested by Prentice et al. (1996).

```
fossil[fossil <= 0.5] <- 0
train[train <= 0.5] <- 0
```

By convention, dissimilarity coefficients are defined for proportional data (Simpson, 2007). Therefore, all pollen data must be reduced to proportions:

```
fossil <- (fossil/rowSums(fossil))
train <- (train/rowSums(train))
```

The variable sums and names can be checked for consistency using some basic R tools. The following command verifies that the taxa sum for every sample amounts to 1 after converting to proportions.

```
rowSums(fossil)
rowSums(train)
```

---

#### 3.2 Apply the Modern Analogue method using the package "rioja".

Fit a Modern Analogue model to the training data set using squared chord distance as a measure of dissimilarity:

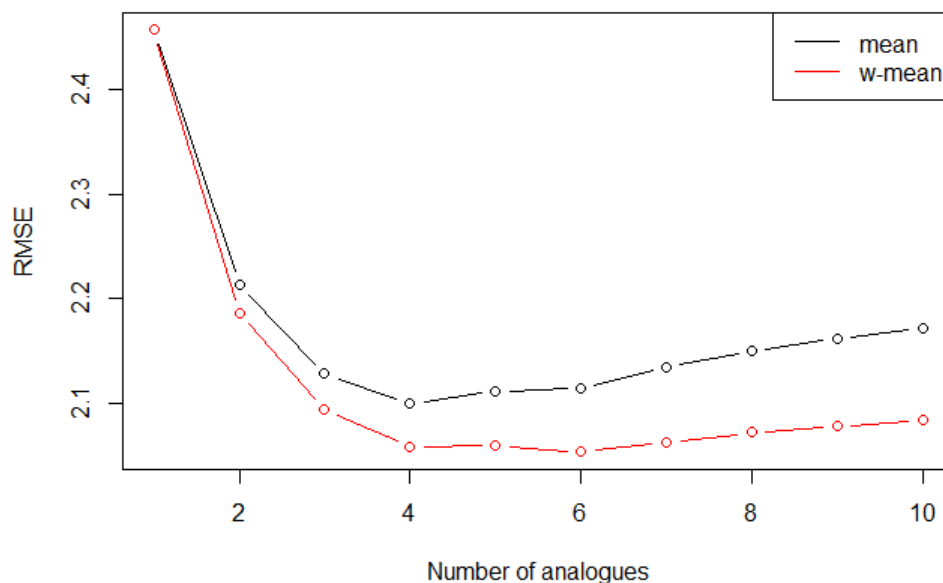
```
Warning in MAT(train, T_ANN, dist.method = "sq.chord", k = 10, lean = FALSE): Inter-sample distances
< 1.0E-6 found (duplicate samples?)
These have been replaced by 1.0E-6
```

Rioja provides different tools to verify the performance of the model. The default is to show errors based on leave-one-out cross-validation.

```
MAT.T_ANN
```

```
##
Method : Modern Analogue Technique
Call : MAT(y = y, x = x, dist.method = "sq.chord", k = 10, lean = FALSE)
##
Distance : sq.chord
No. samples : 3813
No. species : 45
Cross val. : none
##
##
Performance:
RMSE R2 Avg.Bias Max.Bias Skill
N01 2.4575 0.8088 0.0414 3.9867 79.9973
N02 2.2125 0.8399 0.0617 5.1500 83.7871
N03 2.1275 0.8505 0.0531 5.5956 85.0092
N04 2.0986 0.8543 0.0595 6.2383 85.4131
N05 2.1111 0.8525 0.0643 6.1187 85.2396
N06 2.1137 0.8523 0.0751 6.0867 85.2029
N07 2.1338 0.8497 0.0693 6.1286 84.9202
N08 2.1499 0.8476 0.0722 6.4017 84.6916
N09 2.1614 0.8462 0.0779 6.6422 84.5279
N10 2.1720 0.8448 0.0729 6.9620 84.3758
N01.wm 2.4575 0.8088 0.0414 3.9867 79.9973
N02.wm 2.1856 0.8438 0.0538 4.8415 84.1787
N03.wm 2.0934 0.8554 0.0482 5.1797 85.4857
N04.wm 2.0575 0.8600 0.0546 5.7357 85.9797
N05.wm 2.0586 0.8598 0.0603 5.6818 85.9639
N06.wm 2.0527 0.8606 0.0648 5.7100 86.0442
N07.wm 2.0615 0.8595 0.0583 5.7533 85.9246
N08.wm 2.0714 0.8584 0.0606 5.9982 85.7895
N09.wm 2.0774 0.8577 0.0648 6.2021 85.7071
N10.wm 2.0829 0.8571 0.0600 6.4829 85.6305
```

```
screepplot(MAT.T_ANN)
```



The above plot shows the RMSE or prediction error for different numbers of analogues.

Spatially structured training sets are likely to exhibit spatial dependency in both the species and environment fields. In such situations nearby samples cannot be considered independent and leave-one-out (LOO) cross-validation will underestimate the likely errors. A more robust form of CV is so-called h-block CV, in which training samples a distance "h" from the target are omitted from the reconstruction. The problem is to estimate the distance "h". Detailed discussion is beyond the scope of this workshop (see Trachsel & Telford, 2015). For now we will use a figure of 300kmdetermined for an Arctic pollen dataset (Trachsel & Telford, 2015).



To use h-block CV we need to generate a matrix of geographical distances between training set sites. We extract the grid references from our original data table and calculate geographic distances using the function `rdist.earth` in package `fields`:

```
suppressPackageStartupMessages(library(fields))
```

```
library(fields)
LatLong <- EMPD_climate[, c("londd", "latdd")]
geo.dist <- as.matrix(as.dist(rdist.earth(LatLong, miles=FALSE)))
```

```
MAT.T_ANN.h <- crossval(MAT.T_ANN, k=10, cv.method="h-block", h.dist=geo.dist, h.cutoff=300)
MAT.T_ANN.h
```

The errors are larger than those from leave-one-out CV (the default with MAT). Estimating errors in the case of spatially structured training sets is a current area of research and these results should be used as a guide only (see Trachsel & Telford, 2015).

### 3.3 Predict T\_ANN using fossil data

The MAT model can now be applied to the fossil data. In this case, the results are calculated by selecting the 6 best modern analogues for every fossil sample (see screeplot):

```
fossil.MAT.T_ANN <- predict(MAT.T_ANN, fossil, k=6)
```

The output from MAT is a list containing the reconstructions and various diagnostics. We can combine these into a single data frame:

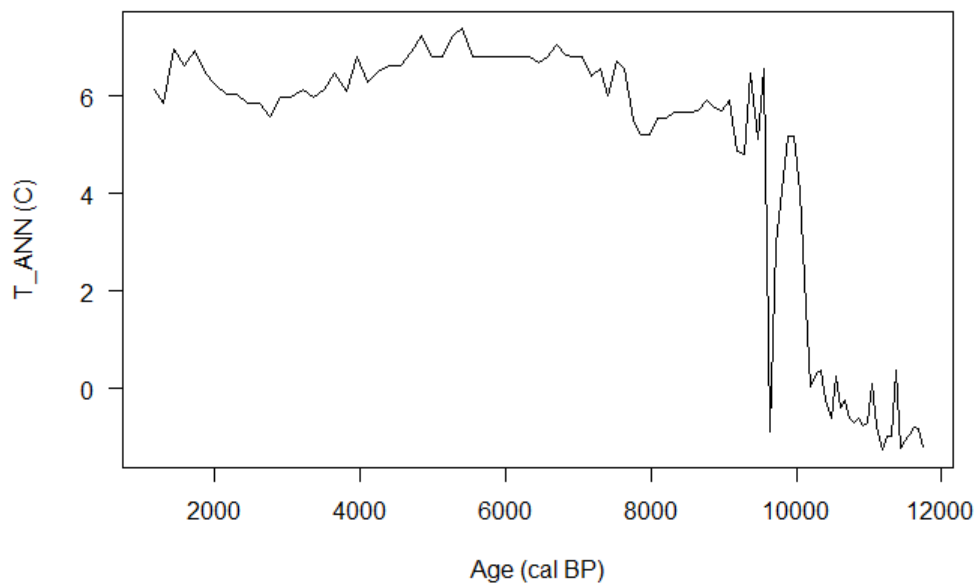
```
fossil.MAT.T_ANN <- data.frame(fossil.MAT.T_ANN)
```

The resulting data frame will now contain the following columns:

|                                     |                                                                                                      |
|-------------------------------------|------------------------------------------------------------------------------------------------------|
| - Sample                            | Name of the fossil sample.                                                                           |
| - k                                 | Number of modern analogues used.                                                                     |
| - fit.MAT                           | Predicted environmental value for fossil data based on the mean values of the k closest analogues.   |
| - fit.MAT.wm                        | Predicted environmental value for fossil data based on the weighted mean of the k closest analogues. |
| - diagnostics.Stdev                 | Standard deviation of the k closest analogues.                                                       |
| - diagnostics.minD                  | Distance of the closest analogue.                                                                    |
| - dist.n                            | Dissimilarities of the k closest analogues.                                                          |
| - x.n.N                             | Environmental value of the k closest analogues.                                                      |
| - match.name.N01-<br>match.name.N06 | Sample names of the k closest analogues.                                                             |

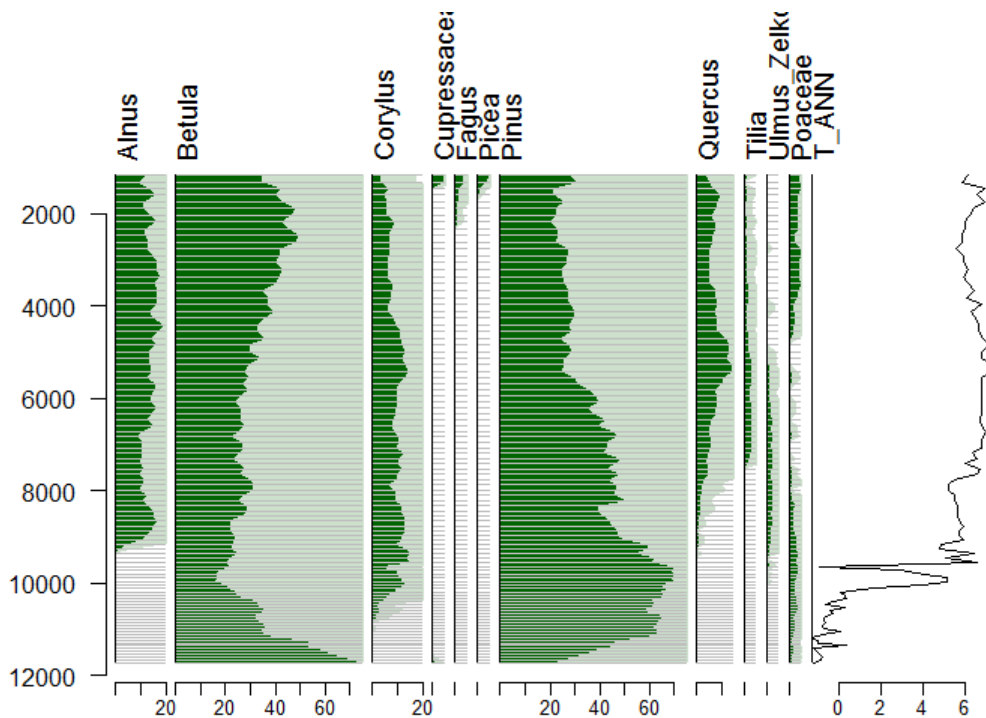
We can then plot the reconstructed values vs. sample age:

```
plot(fossil.age$agebp, fossil.MAT.T_ANN$fit.MAT, type="l", xlab="Age (cal BP)", ylab="T_ANN (C)", las=1)
```



We can also add the reconstruction to the stratigraphic diagram:

```
strat.plot(fossil.sub, yvar=fossil.age$agebp, scale.percent=TRUE, y.rev=TRUE, plot.poly=TRUE, plot.bar="Full", col.poly.line=NA, exag=TRUE, col.exag="auto", col.poly="darkgreen", xRight=0.8)
strat.plot(fossil.MAT.T_ANN[, "fit.MAT", drop=FALSE], yvar=fossil.age$agebp, y.rev=TRUE, xLeft=0.8, add=TRUE, plot.bar=FALSE, y.axis=FALSE, scale.minmax=FALSE, x.names="T_ANN")
```



### 3.4 How good are the analogues?

We can get an idea of which samples have good analogues by plotting minDC vs age, that is the distance to each fossil sample to its closest modern analogue.

```
plot(fossil.age$agebp, fossil.MAT.T_ANN$diagnostics.minD, type="l", xlab="Age (cal BP)", ylab="min DC", las=1)
```

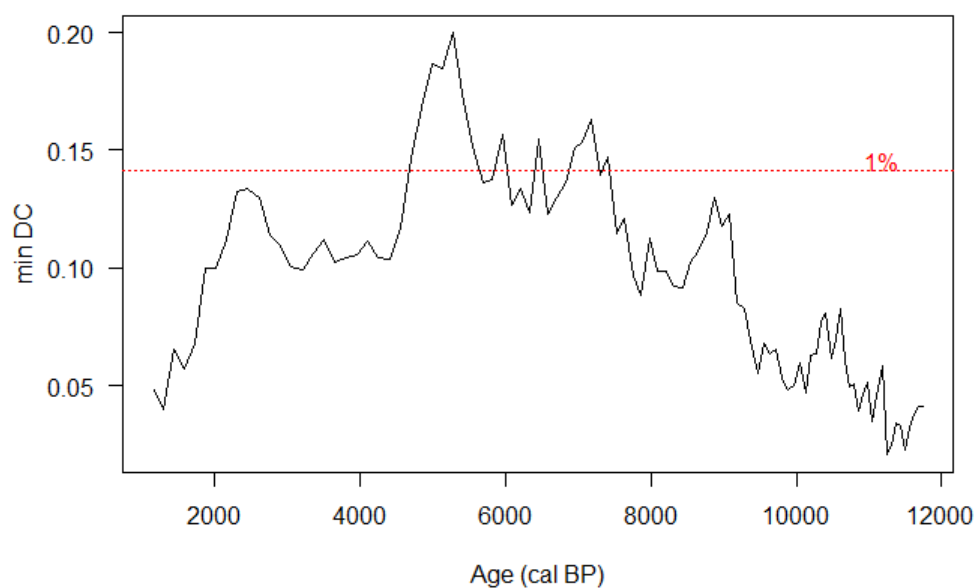


The problem is to define what value of minDC defines a “good” analogue. Some authors (e.g. Bartlein and Whitlock 1993) take the percentiles of the distribution of distance between modern samples. We can calculate these from our MAT results:

```
MAT.percentiles <- quantile(MAT.T_ANN$dist[lower.tri(MAT.T_ANN$dist)], probs=c(0.01, 0.025, 0.05, 0.1))
```

Now add these to our plot:

```
plot(fossil.age$agebp, fossil.MAT.T_ANN$diagnostics.minD, type="l", xlab="Age (cal BP)", ylab="min DC",
 las=1)
abline(h=MAT.percentiles, col="red", lty="dotted")
us <- par("usr")
text(us[2] * 0.90, MAT.percentiles, labels = paste(c(0.01, 0.025, 0.05, 0.1)*100, "%", sep=""), adj=c
 (0, 0), col="red")
```

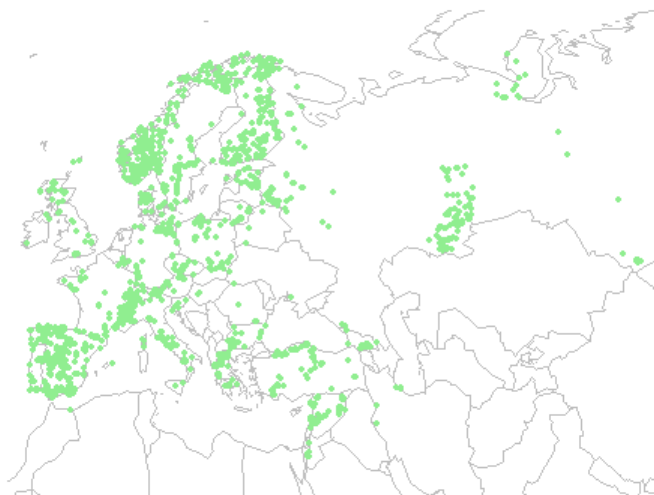


### 3.5 Where are the analogues located?

There are many ways to draw maps in R. Probably the easiest is to use the package `maps`. First create a base map, and set x-axis

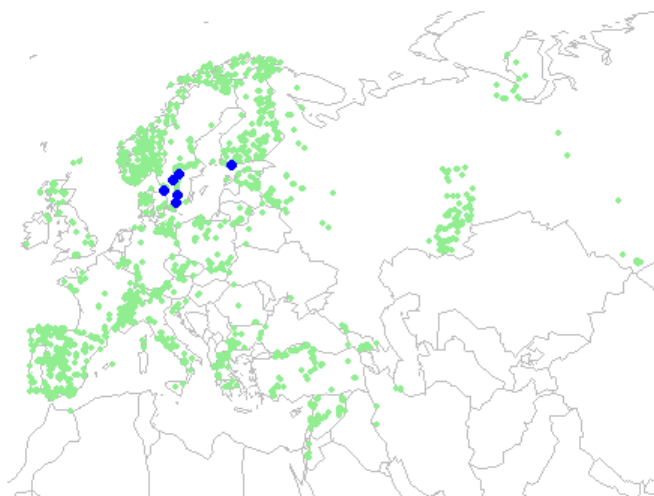
and y-axis limits (in degrees) to plot Europe, then use `points` to add the coordinates of the surface samples.

```
library(maps)
map("world", xlim=c(-12, 90), ylim=c(27, 75), col="grey")
points(EMPD_climate$londd, EMPD_climate$latdd, pch=19, cex=0.5, col="lightgreen")
```



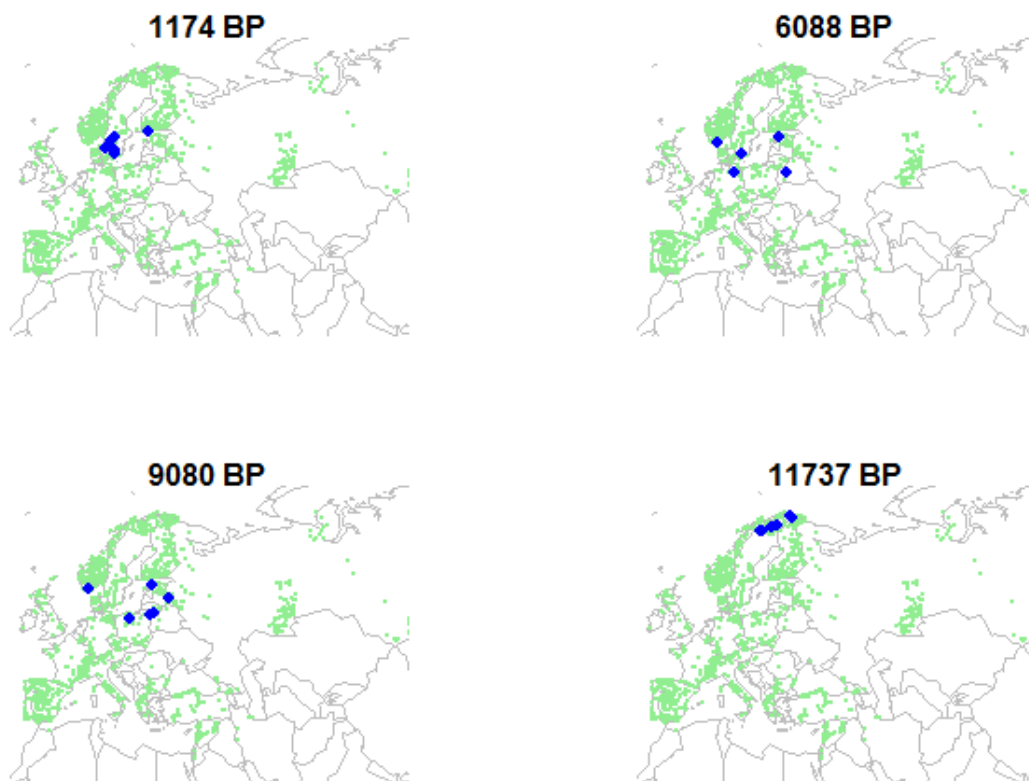
Then extract the coordinates of the analogues:

```
level <- 1 # plot analogues for the uppermost sample in the core
analogues <- fossil.MAT.T_ANN[, 18:23]
names <- sapply(analogues[level,], as.character)
coords <- EMPD_climate[names,]
map("world", xlim=c(-12, 90), ylim=c(27, 75), col="grey")
points(EMPD_climate$londd, EMPD_climate$latdd, pch=19, cex=0.5, col="lightgreen")
points(coords$londd, coords$latdd, pch=19, cex=1, col="blue")
```



Put it all in a loop and plot analogues for different time slices:

```
par(mfrow=c(2,2))
par(mar=c(0, 0, 1, 0))
levels <- c(1, 35, 61, 98)
for (i in 1:4) {
 level <- levels[i]
 analogues <- fossil.MAT.T_ANN[, 18:23]
 names <- sapply(analogues[level,], as.character)
 coords <- EMPD_climate[names,]
 map("world", xlim=c(-12, 90), ylim=c(27, 75), col="grey")
 points(EMPD_climate$londd, EMPD_climate$latdd, pch=19, cex=0.3, col="lightgreen")
 points(coords$londd, coords$latdd, pch=19, cex=1, col="blue")
 title(paste(round(fossil.age[level, "agebp"], 0), "BP"))
}
```

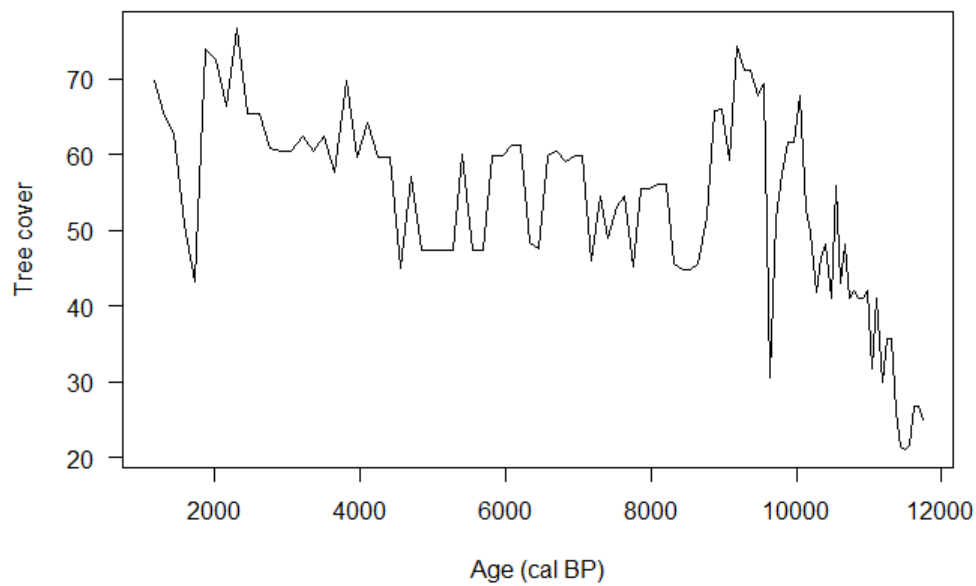


You can repeat the analysis for the other climate variables in the “EMPD\_climate.txt” file, including T\_DJF (winter temperature), T\_JJA (summer temperature), P\_ANN (annual precipitation), P\_DJF (winter precipitation), P\_JJA (summer precipitation),

## 4. Tree cover reconstructions

We can also use MAT to produce tree cover reconstructions from the pollen data. This approach has been previously used by Tarasov et al. (2007) and Williams et al. (2011). The first step consists of building a training dataset by coupling modern pollen spectra with the corresponding present day, quantifiable forest cover data. The selected source for tree cover data is the Global Forest Change Map produced by Hansen et al. (2013) and available at <http://earthenginepartners.appspot.com/science-2013-global-forest> (<http://earthenginepartners.appspot.com/science-2013-global-forest>). The geographic location of every modern pollen sample is available within the EMPD. For training purpose, a search window with a radius of 500m was placed on the map at the location of every pollen sample. The values of every pixel falling within each window were then averaged and assigned to the corresponding sample. The results of this process give tree cover for each modern pollen surface sample, expressed as percentage tree cover within a 500m radius. The training dataset of modern pollen counts and tree-cover data is available in the tab-delimited file “EMPD\_treecover.txt”. Pollen taxa are in columns 1-45 and tree cover is in the column named “treecover\_500m”.

Import the file “EMPD\_treecover.txt” into R and repeat the above instructions using these data to produce a tree cover reconstruction. That is, use “EMPD\_treecover.txt” instead of “EMPD\_climate.txt” and column “treecover\_500m” instead of “T\_ANN”. Your output should look like this:



## 5. References

- Bartlein PJ, Whitlock C. Paleoclimatic interpretation of the Elk Lake pollen Record. In: Bradbury JP, Dean WE, editors. Elk Lake, Minnesota: Evidence for Rapid Climatic Change in the North-Central United States. Geological Society of America, Boulder Colorado, 1993, pp. 275-293.
- Hansen M.C., Potapov P.V., Moore R., Hancher M., Turubanova S.A., Tyukavina A., Thau D., Stehman S.V., Goetz S.J., Loveland T.R., Kommareddy A., Egorov A., Chini L., Justice C.O., Townshend J.R.G. 2013. High-Resolution Global Maps of 21st-Century Forest Cover Change. *Science* 342, 850-853.
- Prentice C., Guiot J., Huntley B., Jolly D., Cheddadi R. 1996. Reconstructing biomes from palaeoecological data: a general method and its application to European pollen data at 0 and 6 ka. *Climate Dynamics*, 12, 185-194.
- Simpson G.L. 2007. Analogue Methods in Palaeoecology: Using the analogue Package. *Journal of Statistical Software*, 22.
- Tarasov P., Williams J.W., Andreev A., Nakagawa T., Bezrukova E., Herzschuh U., Igarashi Y., Müller S., Werner K., Zheng Z. 2007. Satellite- and pollen-based quantitative woody cover reconstructions for northern Asia: Verification and application to late-Quaternary pollen data. *Earth and Planetary Science Letters*, 264, 284-298.
- Trachsel M, Telford RJ. Technical Note: Estimating unbiased transfer-function performances in spatially structured environments. *Climate of the Past Discussions* 2015; 11: 4729-4749.
- Williams JW, Tarasov P, Brewer S, Notaro M. Late Quaternary variations in tree cover at the northern forest-tundra ecotone. *Journal of Geophysical Research* 2011; 116.