

DTW examples for Covid-19 datasets for ECDC

toni.giorgino@cnr.it - CNR Institute of Biophysics & Dept. of Biosciences UniMi

Tue Nov 3 14:44:19 2020

Contents

Generalities (which can be deleted)	1
Data loading	1
Preprocessing and test alignment	2
All-vs-all alignment	3
Clustering	3

Generalities (which can be deleted)

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Data loading

Note: check if the COVID19 package <https://cran.r-project.org/web/packages/COVID19/index.html> has similar data and it is more convenient.

```
library(dplyr)
library(dtw)
library(readxl)
```

Load all sheets in list elements. **Important:** sort by dates.

```
xn <- "example_data.xlsx"
countries <- excel_sheets(xn)
Ncountries <- length(countries)

data<-list()
```

```
for (c in countries) {
  data[[c]] <- read_excel(xn, sheet=c) %>% arrange(Date)
}
```

Preprocessing and test alignment

Just one test alignment, for demonstration purposes. **Note.** Using columns as-is is probably simplistic, because columns may have different lags depending on the local country characteristics (e.g. infection-to-report delays, which may not even be constant).

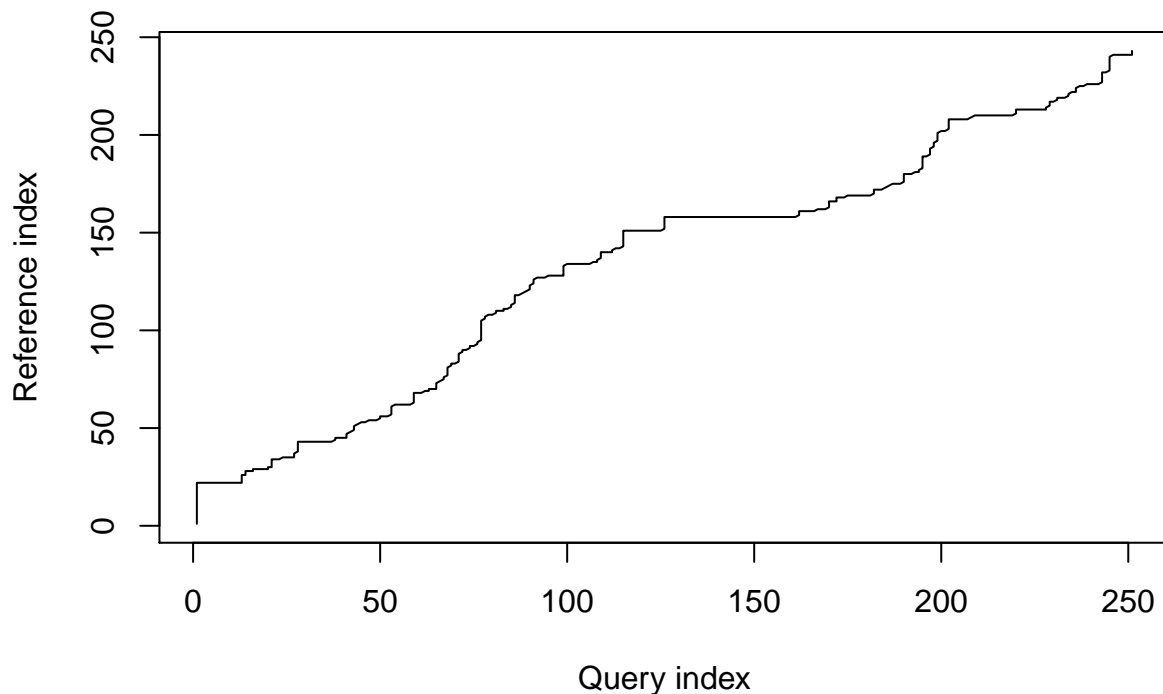
The columns must be matched, of course (e.g. column 1 is new cases for all countries).

Here I define a preprocessing function which returns a matrix. The preprocessing includes scaling. This implies that columns will get the same “weight” - which may not be desirable! If different weights are desired, just multiply the columns.

```
df.preprocess <- function(df) {
  tmp <- df %>% select(New_Cases, New_Deaths, Daily_Incidence) %>% as.matrix
  tmp[is.na(tmp)] <- 0 # NAs become 0
  sc <- scale(tmp) # Standardize by column
  sc
}
```

```
tmp.at <- df.preprocess(data[["Austria"]])
tmp.fr <- df.preprocess(data[["France"]])
```

```
dtw.at.fr <- dtw(tmp.at, tmp.fr) # The most basic format, all default parameters.
plot(dtw.at.fr)
```



```
dtw.at.fr$normalizedDistance # This is the distance (dissimilarity) to be used in clustering
```

```
## [1] 0.3470184
```

All-vs-all alignment

Now compute the all-vs-all dissimilarity matrix via DTW. Note that the default `step.pattern=symmetric2` which gives a symmetric distance matrix. This is not true in general.

```
dmat <- matrix(NA, nrow=Ncountries, ncol=Ncountries)
colnames(dmat) <- countries
rownames(dmat) <- countries

for (c1 in countries) {
  pp.c1 <- df.preprocess(data[[c1]])
  for (c2 in countries) {
    pp.c2 <- df.preprocess(data[[c2]])
    aln <- dtw(pp.c1, pp.c2, distance.only = T)
    dmat[c1,c2] <- aln$normalizedDistance
  }
}
```

Clustering

Now any hierarchical algorithm may be used.

```
dist.dmat <- as.dist(dmat)
hclust(dist.dmat)

##
## Call:
## hclust(d = dist.dmat)
##
## Cluster method      : complete
## Number of objects: 3
```