data\_covid\_ts <- fread("\_rmd/data\_covid\_time\_series\_2020-05-24.csv")

data\_covid\_ts[, DateRep := as.Date(DateRep)] # transform character to Date class

# what we got

str(data\_covid\_ts)

## Classes 'data.table' and 'data.frame': 19344 obs. of 17 variables:

## $ Country : chr "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...

## $ DateRep : Date, format: "2020-01-22" "2020-01-23" "2020-01-24" "2020-01-25" ...

## $ Cases\_cumsum : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Cases : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Deaths\_cumsum : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Deaths : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Recovered\_cumsum : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Recovered : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Active\_cases\_cumsum : int 0 0 0 0 0 0 0 0 0 0 ...

## $ New cases per 1 million population : int 0 0 0 0 0 0 0 0 0 0 ...

## $ New deaths per 1 million population : int 0 0 0 0 0 0 0 0 0 0 ...

## $ New recovered cases per 1 million population : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Death rate (%) : num NA NA NA NA NA NA NA NA NA NA ...

## $ Total active cases per 1 million population : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Total deaths per 1 million population : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Total cases per 1 million population : int 0 0 0 0 0 0 0 0 0 0 ...

## $ Total recovered cases per 1 million population: int 0 0 0 0 0 0 0 0 0 0 ...

## - attr(\*, ".internal.selfref")=

Since I want to analyze (cluster) trajectories of countries’ active cases spread, I need to set **starting position for every countries’ time series** – in this case (and in other many analyses out there) *100-th* cumulative confirmed case is set as starting point.  
I will also use only top *82* affected countries (+ Slovakia as my home country) for the whole analysis.  
Let’s transform our data for ‘since first 100-th case’ countries’ trajectories (with the same lengths!).

n\_cases <- 100 # you can vary

statistic <- 'Total active cases per 1 million population' # you can also choose other stat

# Cases greater than threshold

data\_covid\_100\_cases <- copy(data\_covid\_ts[,

.SD[DateRep >= .SD[Cases\_cumsum >= n\_cases,

min(DateRep,

na.rm = T)]],

by = .(Country)])

# Order data

setorder(data\_covid\_100\_cases,

Country,

DateRep)

# Create 'days since' column

data\_covid\_100\_cases[, (paste0("Days\_since\_first\_",

n\_cases, "\_case")) := 1:.N,

by = .(Country)]

# top N countries selection for an analysis

data\_cases\_order <- copy(data\_covid\_100\_cases[,

.SD[DateRep == max(DateRep),

get(statistic)],

by = .(Country)

])

setnames(data\_cases\_order, "V1", statistic)

setorderv(data\_cases\_order, statistic, -1)

# subset data based on selected parameter - top 82 countries and analyzed columns

data\_covid\_cases\_sub <- copy(data\_covid\_100\_cases[.(c(data\_cases\_order[1:82][![is.na](http://is.na)(Country),

Country],

"Slovakia")),

on = .(Country),

.SD,

.SDcols = c("Country",

paste0("Days\_since\_first\_",

n\_cases, "\_case"),

statistic)

])

# Make same length time series from countries data

data\_covid\_trajectories <- dcast(data\_covid\_cases\_sub,

get(paste0("Days\_since\_first\_", n\_cases, "\_case")) ~

Country,

value.var = statistic)

setnames(data\_covid\_trajectories,

colnames(data\_covid\_trajectories)[1],

paste0("Days\_since\_first\_", n\_cases, "\_case"))

# see interactively what we got with DT package

dt <- DT::datatable(data\_covid\_trajectories,

class = "compact",

extensions = 'Scroller',

options = list(

dom = 't',

deferRender = TRUE,

scrollY = 270,

scroller = TRUE,

scrollX = TRUE

))

htmlwidgets::saveWidget(dt, "dt\_traj\_1.html", selfcontained = T)

You can see that we got nicely the same length time series for every country.

Now, preparation of **trajectories’ data for clustering** is coming…  
We have to remove missing rows/ columns if there are so + I will preprocess time series with **Simple Moving Average** (SMA) to little bit smooth our trajectories (removes noise) – the function repr\_sma is implemented

# save stats of dt

n\_col <- ncol(data\_covid\_trajectories)

n\_row <- nrow(data\_covid\_trajectories)

n\_row\_na <- rowSums(data\_covid\_trajectories[, lapply(.SD, [is.na](http://is.na))])

n\_col\_na <- colSums(data\_covid\_trajectories[, lapply(.SD, [is.na](http://is.na))])

# remove all NA rows and cols - for sure

if (length(which(n\_row\_na %in% n\_col)) != 0) {

data\_covid\_trajectories <- copy(data\_covid\_trajectories[-which(n\_row\_na == n\_col)])

}

if (length(which(n\_col\_na %in% n\_row)) != 0) {

data\_covid\_trajectories <- copy(data\_covid\_trajectories[, -which(n\_col\_na %in% n\_row),

with = FALSE])

}

# use SMA for preprocessing time series

q\_sma <- 3 # order of moving average

data\_covid\_trajectories[,

(colnames(data\_covid\_trajectories)[-1]) :=

lapply(.SD, function(i)

c(rep(NA, q\_sma - 1),

ceiling(repr\_sma(i, q\_sma))

)),

.SDcols = colnames(data\_covid\_trajectories)[-1]]

# again see data interactively

dt <- DT::datatable(data\_covid\_trajectories,

class = "compact",

extensions = 'Scroller',

options = list(

dom = 't',

deferRender = TRUE,

scrollY = 270,

scroller = TRUE,

scrollX = TRUE

))

htmlwidgets::saveWidget(dt, "dt\_traj\_2.html", selfcontained = T)

**Clustering trajectories with the hierarchical method with DTW distance**

Since we use data with different lengths, we have to use different distance measures than Euclidean (or Manhattan, etc.).  
Here comes very handy [**Dynamic Time Warping**](https://en.wikipedia.org/wiki/Dynamic_time_warping) distance measure that can compute distances between time series with various lags and different lengths.

As a clustering method, I picked [**hierarchical clustering with Ward criterion**](https://en.wikipedia.org/wiki/Ward%27s_method) for its next nice post-analysis tools as **dendrograms**.

Let’s define clustering function with DTW distance with additional data preprocessing steps necessary for dtwclust package. I allow user also vary number of clusters and normalization of time series before clustering.

cluster\_trajectories <- function(data, k, normalize = FALSE) {

# transpose data for clustering

data\_trajectories\_trans <- t(data[, .SD,

.SDcols = colnames(data)[-1]])

# create list of time series

data\_trajectories\_trans\_list <- lapply(1:nrow(data\_trajectories\_trans), function(i)

na.omit(data\_trajectories\_trans[i,]))

names(data\_trajectories\_trans\_list) <- colnames(data)[-1]

# remove data with length <= 1

n\_list <- sapply(1:length(data\_trajectories\_trans\_list), function(i)

length(data\_trajectories\_trans\_list[[i]]))

names(n\_list) <- names(data\_trajectories\_trans\_list)

if (length(which(n\_list %in% 0:1)) != 0) {

data\_trajectories\_trans\_list <- data\_trajectories\_trans\_list[-which(n\_list %in% 0:1)]

}

list\_names <- names(data\_trajectories\_trans\_list)

# normalization

if (normalize) {

data\_trajectories\_trans\_list <- lapply(names(data\_trajectories\_trans\_list),

function(i)

norm\_z(data\_trajectories\_trans\_list[[i]])

)

names(data\_trajectories\_trans\_list) <- list\_names

}

# clustering with dtwclust package function

hc\_res <- tsclust(data\_trajectories\_trans\_list,

type = "hierarchical",

k = k,

distance = "dtw\_basic",

centroid = dba,

trace = FALSE,

seed = 54321,

control = hierarchical\_control(method = "ward.D2"),

args = tsclust\_args(dist = list(norm = "L2"))

)

return(hc\_res)

}

Let’s cluster data with *14* clusters and **normalization of countries’ trajectories** for extracting clusters with **same trends (curves)** – not magnitudes! It is very important thing before every clustering/ classification task.

clust\_res <- cluster\_trajectories(data = data\_covid\_trajectories,

k = 14,

normalize = TRUE)

# results

clust\_res

## hierarchical clustering with 14 clusters

## Using dtw\_basic distance

## Using dba centroids

## Using method ward.D2

##

## Time required for analysis:

## user system elapsed

## 0.39 0.00 0.08

##

## Cluster sizes with average intra-cluster distance:

##

## size av\_dist

## 1 21 0.5011793

## 2 6 0.5170467

## 3 2 0.9447618

## 4 10 0.4015355

## 5 11 0.5774944

## 6 5 0.6441501

## 7 7 0.4719638

## 8 2 0.5682452

## 9 5 0.7722092

## 10 2 0.8360796

## 11 4 0.4361994

## 12 5 0.5527005

## 13 2 0.5363868

## 14 1 0.0000000

Let’s prepare clustered data for visualization:

# prepare time series

data\_clust\_id <- data.table(Cluster = clust\_res@cluster,

Country = names(clust\_res@cluster))

data\_plot <- melt(data\_covid\_trajectories,

id.vars = colnames(data\_covid\_trajectories)[1],

[variable.name](http://variable.name) = "Country",

variable.factor = FALSE,

[value.name](http://value.name) = statistic,

value.factor = FALSE)

data\_plot <- copy(data\_plot[.(data\_clust\_id$Country), on = .(Country)])

data\_plot[data\_clust\_id,

on = .(Country),

Cluster := i.Cluster]

# again see what we got interactively

dt <- DT::datatable(data\_plot,

class = "compact",

extensions = 'Scroller',

filter = "top",

options = list(

dom = 't',

deferRender = TRUE,

scrollY = 270,

scroller = TRUE,

scrollX = TRUE

))

htmlwidgets::saveWidget(dt, "dt\_data\_plot.html", selfcontained = T)

You can also search for your preferred country in the datatable.

Here comes finally **plot of cluster members** with ggplot2 package (log scale is used for better comparison of trends):

theme\_my <- theme(panel.border = element\_rect(fill = NA,

colour = "grey10"),

panel.background = element\_blank(),

panel.grid.minor = element\_line(colour = "grey85"),

panel.grid.major = element\_line(colour = "grey85"),

panel.grid.major.x = element\_line(colour = "grey85"),

axis.text = element\_text(size = 12, face = "bold"),

axis.title = element\_text(size = 13, face = "bold"),

plot.title = element\_text(size = 16, face = "bold"),

strip.text = element\_text(size = 12, face = "bold"),

strip.background = element\_rect(colour = "black"))

ggplot(data\_plot,

aes(get(colnames(data\_plot)[1]),

get(statistic),

group = Country)) +

facet\_wrap(~Cluster,

ncol = ceiling(data\_plot[, sqrt(uniqueN(Cluster))]),

scales = "free\_y") +

geom\_line(color = "grey10",

alpha = 0.75,

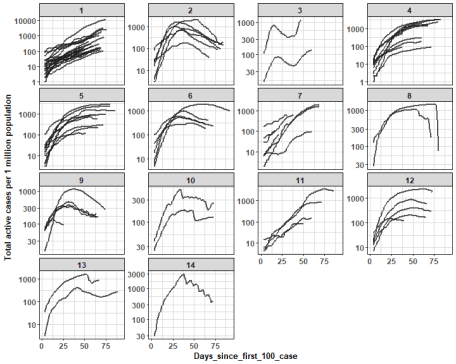
size = 0.8) +

scale\_y\_continuous(trans = 'log10') +

labs(x = colnames(data\_plot)[1],

y = statistic) +

theme\_my



We can see nicely distinguishable clusters with various active cases trends (settled, rapid/ steady increase/ decrease).

Let’s check some clusters interactively with dygraphs package:

data\_clust\_focus <- dcast(data\_plot[.(c(2,6)), on = .(Cluster)],

Days\_since\_first\_100\_case ~ Country,

value.var = statistic)

dyg <- dygraph(data\_clust\_focus,

main = "Clusters n. 2 and 6") %>%

dyAxis("x", label = colnames(data\_clust\_focus)[1]) %>%

dyAxis("y", label = statistic) %>%

dyOptions(strokeWidth = 2,

drawPoints = F,

pointSize = 3,

pointShape = "circle",

logscale = TRUE,

colors = RColorBrewer::brewer.pal(ncol(data\_clust\_focus)-1, "Set2")) %>%

dyHighlight(highlightSeriesOpts = list(strokeWidth = 2.5,

pointSize = 4)) %>%

dyLegend(width = 150, show = "follow",

hideOnMouseOut = TRUE, labelsSeparateLines = TRUE)

saveWidget(dyg, "dyg\_focus\_clust\_2\_6.html", selfcontained = T)

Here, I picked two clusters (2 and 6) with nice decreasing trends – there are countries mostly from Central/ West Europe.

Let’s see also clusters with increasing trends of active cases per 1 mil. population:

data\_clust\_focus <- dcast(data\_plot[.(7), on = .(Cluster)],

Days\_since\_first\_100\_case ~ Country,

value.var = statistic)

dyg <- dygraph(data\_clust\_focus,

main = "Cluster n. 7") %>%

dyAxis("x", label = colnames(data\_clust\_focus)[1]) %>%

dyAxis("y", label = statistic) %>%

dyOptions(strokeWidth = 2,

drawPoints = F,

pointSize = 3,

pointShape = "circle",

logscale = TRUE,

colors = RColorBrewer::brewer.pal(ncol(data\_clust\_focus)-1, "Set2")) %>%

dyHighlight(highlightSeriesOpts = list(strokeWidth = 2.5,

pointSize = 4)) %>%

dyLegend(width = 150, show = "follow",

hideOnMouseOut = TRUE, labelsSeparateLines = TRUE)

saveWidget(dyg, "dyg\_focus\_clust\_7.html", selfcontained = T)

data\_clust\_focus <- dcast(data\_plot[.(1), on = .(Cluster)],

Days\_since\_first\_100\_case ~ Country,

value.var = statistic)

dyg <- dygraph(data\_clust\_focus,

main = "Cluster n. 1") %>%

dyAxis("x", label = colnames(data\_clust\_focus)[1]) %>%

dyAxis("y", label = statistic) %>%

dyOptions(strokeWidth = 2,

drawPoints = F,

pointSize = 3,

pointShape = "circle",

logscale = TRUE,

colors = RColorBrewer::brewer.pal(ncol(data\_clust\_focus)-1, "Set2")) %>%

dyHighlight(highlightSeriesOpts = list(strokeWidth = 2.5,

pointSize = 4)) %>%

dyLegend(width = 150, show = "follow",

hideOnMouseOut = TRUE, labelsSeparateLines = TRUE)

saveWidget(dyg, "dyg\_focus\_clust\_1.html", selfcontained = T)

We can see that on this day, the increasing trend of active cases has countries mostly in Western Asia, South America, and Africa.

**Post-analysis visualizations with dendrograms and MDS**

In order to see whole connectivity between countries’ clusters as a tree, we can use for example **dendrogram**.  
Here, we can simply use object of clustering result to generate the tree:

dend <- as.dendrogram(clust\_res)

dend <- dend %>%

color\_branches(k = 14) %>%

color\_labels(k = 14) %>%

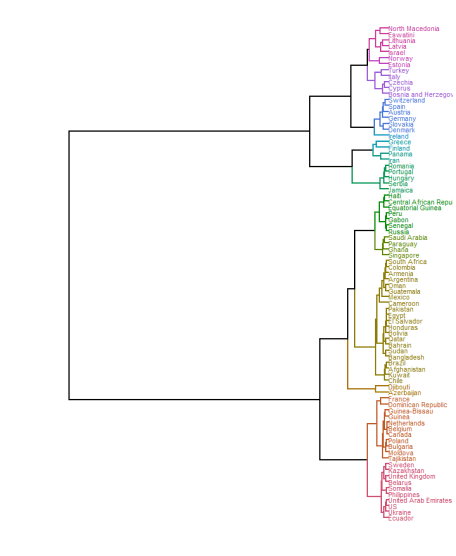
set("branches\_lwd", 1) %>%

set("labels\_cex", 0.8)

ggd1 <- as.ggdend(dend)

ggplot(ggd1,

horiz = T)



In order to see for example connections between countries in 2D scatter plot, we can use [**dimensionality reduction method Multidimensional scaling (MDS)**](https://en.wikipedia.org/wiki/Multidimensional_scaling). It uses (stored) distance matrix between objects – and we have it in our clustering result object (Yey clust\_res@distmat!). For countries labels, I use great package ggrepel.

mds\_classical <- cmdscale(clust\_res@distmat, eig = FALSE, k = 2)

data\_plot <- data.table(mds\_classical,

Country = row.names(mds\_classical),

Cluster = clust\_res@cluster)

ggplot(data\_plot, aes(x = get("V1"),

y = get("V2"),

label = Country,

color = as.factor(Cluster))) +

geom\_label\_repel(size = 4.2,

alpha = 0.95,

segment.alpha = 0.35,

label.r = 0.1,

box.padding = 0.25,

label.padding = 0.3,

label.size = 0.35,

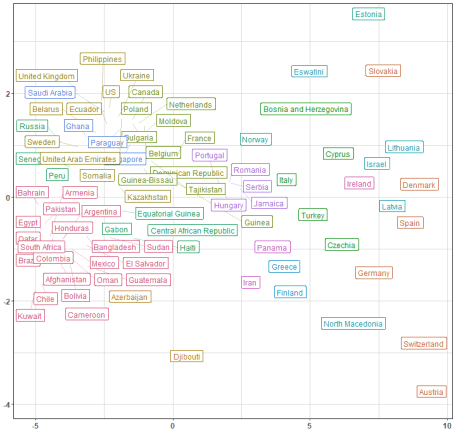
max.iter = 2500) +

scale\_color\_manual(values = colorspace::rainbow\_hcl(14, c = 90, l = 50)) +

labs(x = NULL, y = NULL, color = NULL) +

guides(color = FALSE) +

theme\_my



In both graphs (dendrogram and MDS scatter plot), we can see clearly how far (or close) are countries from each other based on **DTW distance**.