Assignment 2

Cesaro Giada

23/12/2019

Contents

1 Run and explain the commands **2**
1.1 Commands for ggplot2 graphs 7

2 Some more exploratory data analysis **18**

3 Cluster analysis **26**

4 Time series clustering **29**
4.1 Dynamic Time Warping 29

5 Cluster analysis based on the issue **37**
5.1 Clustering issues based on Countries' "Yes"-proportions 37
5.2 Clustering issues based on yearly "Yes"-proportions 40
5.3 Clustering Countries based on the issue 41

6 Principal Component Analysis and visualization **43**

1 Run and explain the commands

The `un_votes` dataset (`un_votes` package) will be used for this task. Let's load the library and have a look at the data.

```
library(unvotes)
head(un_votes)
```

```
# A tibble: 6 x 4
   rcid country                  country_code vote
  <int> <chr>                    <chr>        <fct>
1     3 United States of America US           yes
2     3 Canada                   CA           no
3     3 Cuba                     CU           yes
4     3 Haiti                    HT           yes
5     3 Dominican Republic       DO           yes
6     3 Mexico                   MX           yes
```

The following commands all use the *pipeline* notation, which the whole `dplyr` library is based on. The pipeline (`%>%`) allows to lighten the notation when you have to perform several operations in a row. For example, in order to apply the function `f` first and then the function `g`, you can write `data %>% f(...) %>% g(...)` instead of nestling functions into functions (`g(f(data))`), which can be quite heavy especially when you have a large number of functions.
In what follows, the command will first be explained and then it will be displayed and executed in order to see its output.

The function `count` is used to count data in each group. It looks therefore for any `group_by` clause (or any grouping of the data, if `data` has already been grouped according to some variable) and returns the number of item for each group. If the data object has not got any grouping, the command will treat the dataset as if it was a unique large group, like in this case. The `count` command therefore returns the total number of instances in `un_votes`.

```
un_votes %>% count
```

```
# A tibble: 1 x 1
       n
   <int>
1 738764
```

There is however a difference with the usual command `nrow(un_votes)`: despite returning the same value, `count` returns a 1x1 `tibble` (one item with one row, named `n`), while `nrow` returns an integer value.

```
nrow(un_votes)
```

```
[1] 738764
```

```
str(nrow(un_votes))
```

```
 int 738764
```

```
str(un_votes %>% count)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   1 obs. of  1 variable:
 $ n: int 738764
```

The `filter()` command allows to select a subset of the data based on the condition within braces. It returns a `tibble` with the same columns (attributes) of the original `dataset`, but with a lower number of rows, since only the rows matching the filter condition are returned. In the following case the condition is based on an equality.

```
un_votes %>% filter(vote=="yes")
```

```
# A tibble: 588,800 x 4
    rcid country                   country_code vote
   <int> <chr>                     <chr>        <fct>
 1     3 United States of America US           yes
 2     3 Cuba                    CU           yes
 3     3 Haiti                   HT           yes
 4     3 Dominican Republic      DO           yes
 5     3 Mexico                  MX           yes
 6     3 Guatemala               GT           yes
 7     3 Honduras                HN           yes
 8     3 El Salvador             SV           yes
 9     3 Nicaragua               NI           yes
10     3 Costa Rica              CR           yes
# ... with 588,790 more rows
```

If a `count` command is called after a `filter` command, the filtered dataset is considered as the new dataset, and the same considerations above apply to this case. The number returned right below is the number of rows satisfying the matching condition.

```
un_votes %>% filter(vote=="yes") %>% count
```

```
# A tibble: 1 x 1
       n
   <int>
1 588800
```

The `summarize()` command creates one or more scalar variables summarizing the variables of a dataset. It the tables has groups created by `group_by()`, `summarize()` will result in one row (or more, if multiple arguments are used within braces, separated by commas) for each group. With no groups, `summarize()` will result in one row. The following two commands are applied to an ungrouped and grouped dataset respectively. The summarized measures are the proportion of observations with value `yes` on the `vote` column, and the total number of observations. They return a `tibble` with as many rows as groups, and as many columns as summarize conditions.

```
un_votes %>% summarize(total = n(), percent_yes = mean(vote =="yes"))
```

```
# A tibble: 1 x 2
   total percent_yes
   <int>       <dbl>
1 738764       0.797
```

```
by_country <- un_votes %>%
  group_by(country) %>%
  summarize(votes = n(),
            percent_yes = mean(vote == "yes"))
by_country
```

```
# A tibble: 200 x 3
   country             votes percent_yes
   <chr>               <int>       <dbl>
 1 Afghanistan          4972       0.842
 2 Albania              3514       0.716
 3 Algeria              4527       0.898
 4 Andorra              1564       0.645
 5 Angola               3075       0.922
```

```
 6 Antigua and Barbuda  2658       0.919
 7 Argentina            5361       0.779
 8 Armenia              1629       0.759
 9 Australia            5399       0.552
10 Austria              4939       0.633
# ... with 190 more rows
```

Let's dig a little bit deeper into what `group_by` does. A dataset which is group is seen by `R` as a `grouped_df` object, which has the structure displayed below. It is a list of different objects: the first four are the variables of the original dataframe, which remain unchanged, and then there are 200 sublists (where 200 is the number of different Countries in the dataset, hence different groupings), each containing a set of integer values. Each value is the index position of the item belonging to the corresponding group.
The following output has omitted since it would otherwise list 200 elements, but that is how the structure of any R object can be explored.

```
str(un_votes %>%
  group_by(country))
```

The items of a dataset can be arranged according to one or multiple variables. If multiple variables are listed, the data is first sorted according to the first variable, and the records with the same value for the first variable are then sorted according to the second variable and so on. By default the ordering is ascending (in case of a character variable the ordering is from A to Z), but a descending order can be specified using `desc()`.

```
arrange(by_country, percent_yes)
```

```
# A tibble: 200 x 3
   country                                              votes percent_yes
   <chr>                                                <int>       <dbl>
 1 Zanzibar                                                 2       0
 2 United States of America                              5390       0.284
 3 Palau                                                  896       0.323
 4 Israel                                                4944       0.346
 5 Federal Republic of Germany                           2067       0.396
 6 Micronesia (Federated States of)                      1462       0.414
 7 United Kingdom of Great Britain and Northern Ireland  5372       0.429
 8 France                                                5325       0.434
 9 Marshall Islands                                      1600       0.489
10 Belgium                                               5391       0.495
# ... with 190 more rows
```

```
arrange(by_country, desc(percent_yes))
```

```
# A tibble: 200 x 3
   country               votes percent_yes
   <chr>                 <int>       <dbl>
 1 Seychelles             1790       0.978
 2 Timor-Leste             837       0.970
 3 Sao Tome and Principe  2389       0.967
 4 Cabo Verde             3292       0.960
 5 Djibouti               3345       0.956
 6 Guinea Bissau          3070       0.956
 7 Comoros                2530       0.945
 8 Mozambique             3456       0.943
 9 United Arab Emirates   4031       0.941
10 Suriname               3410       0.941
# ... with 190 more rows
```

The `inner_join()` command allows to collect information from multiple tables. It returns all rows from the first table (here: `un_votes`) where there are matching values[1] in the second table (`un_roll_calls`), and all the columns of the first and the second table are returned. If there are multiple matches between the two tables, all combination of the matches are returned.
Here the `un_votes` is first joined to the `un_roll_calls` and then to the `un_roll_call_issues` table.

```
votes_per = un_votes %>%
  inner_join(un_roll_calls, by = "rcid")

votes_issues <- un_votes %>%
  inner_join(un_roll_call_issues, by = "rcid")
votes_issues
```

```
# A tibble: 768,674 x 6
    rcid country                 country_code vote   short_name issue
   <int> <chr>                   <chr>        <fct>  <chr>      <chr>
 1     6 United States of Ameri... US           no     hr         Human righ...
 2     6 Canada                  CA           no     hr         Human righ...
 3     6 Cuba                    CU           yes    hr         Human righ...
 4     6 Dominican Republic      DO           absta... hr         Human righ...
 5     6 Mexico                  MX           yes    hr         Human righ...
 6     6 Guatemala               GT           no     hr         Human righ...
 7     6 Honduras                HN           yes    hr         Human righ...
 8     6 El Salvador             SV           absta... hr         Human righ...
 9     6 Nicaragua               NI           yes    hr         Human righ...
10     6 Panama                  PA           absta... hr         Human righ...
# ... with 768,664 more rows
```

The following command[2] selects the UN issue "Colonialism" and summarizes, for each Country, the percentage of "yes" votes in the above mentioned matter. The Countries are then ordered (ascendent order) according to this percentage. It can be see that the USA are the Country whith the lowest number of "yes" votes when "Colonialism" issues have been discussed.

```
cosa = votes_issues %>%
  filter(issue == "Colonialism") %>%
  group_by(country) %>%
  summarize(percent_yes = mean(vote == "yes")) %>%
  arrange(percent_yes)
cosa
```

```
# A tibble: 199 x 2
   country                                                 percent_yes
   <chr>                                                         <dbl>
 1 United States of America                                      0.166
 2 Micronesia (Federated States of)                              0.194
 3 Israel                                                        0.233
 4 Palau                                                         0.26
 5 Federal Republic of Germany                                   0.268
 6 France                                                        0.281
 7 United Kingdom of Great Britain and Northern Ireland          0.283
 8 Nauru                                                         0.374
 9 Belgium                                                       0.388
10 Marshall Islands                                              0.396
# ... with 189 more rows
```

[1] it means that all the variable in common of the two dataframes have the same values for two records

[2] Note: it is been modified with respect to the original file of commands, since it contained vote== 1 instead of vote=="yes". It therefore returned all percentages equal to 0

The next commands do the following:

- the first one groups data by "issue" and counts the number of United Nation councils which were made for each "issue".
- the second one does the same thing, except that the previous grouping is broken down into Countries (grouping based on two conditions). It can be seen, therefore, how many times each Country has taken part to a council for each given issue.

```
by_issues = group_by(votes_issues,issue)
summarize(by_issues,count=n())
```

```
# A tibble: 6 x 2
  issue                                count
  <chr>                                <int>
1 Arms control and disarmament        146581
2 Colonialism                         127027
3 Economic development                 68828
4 Human rights                        146441
5 Nuclear weapons and nuclear material 115266
6 Palestinian conflict                164531
```

```
by_issues = group_by(votes_issues,country,issue)
summarize(by_issues,count=n())
```

```
# A tibble: 1,194 x 3
# Groups:   country [199]
   country     issue                                count
   <chr>       <chr>                                <int>
 1 Afghanistan Arms control and disarmament           897
 2 Afghanistan Colonialism                            898
 3 Afghanistan Economic development                   440
 4 Afghanistan Human rights                           904
 5 Afghanistan Nuclear weapons and nuclear material   711
 6 Afghanistan Palestinian conflict                   994
 7 Albania     Arms control and disarmament           620
 8 Albania     Colonialism                            717
 9 Albania     Economic development                   306
10 Albania     Human rights                           735
# ... with 1,184 more rows
```

The `separate()` command allows to break down the first argument, which is of format type `Date` ("YY-MM-DD"), into three columns, which are the ones listed as a character vector and passed to the argument `into`. The three resulting columns are of type `chr`. The variable `year` which has just been created is used as grouping factor; the number of records for each year are counted, and the percentage of "yes" votes is computed.

```
votes_per2 = votes_per %>% separate(date,into=c("year","month","day"))

votes_per2 %>%
  group_by(year) %>%
  summarize(total = n(),
            percent_yes = mean(vote == "yes"))
```

```
# A tibble: 69 x 3
   year  total percent_yes
   <chr> <int>       <dbl>
 1 1946   2143       0.573
```

```
 2 1947   2039       0.569
 3 1948   3454       0.400
 4 1949   5700       0.425
 5 1950   2911       0.497
 6 1951    402       0.657
 7 1952   4082       0.546
 8 1953   1537       0.632
 9 1954   1788       0.622
10 1955   2169       0.695
# ... with 59 more rows
```

`by_country` is a `tbl_df` with 200 rows (one for each Country) and 3 variables: `country`, `total` and `percent_yes`. This table is first arranged according to the `percent_yes` variable, and then the rows are filtered according to the `total` column: thise with `total`<=100 are discarded.
The second command computes the same summary measures, the only difference is that the grouping is done by 'year.

```
by_country <- votes_per2 %>%
  group_by(country) %>%
  summarize(total = n(),
            percent_yes = mean(vote == "yes"))

by_country %>%
  arrange(percent_yes) %>%
  filter(total > 100)
```

```
# A tibble: 199 x 3
   country                                                total percent_yes
   <chr>                                                  <int>       <dbl>
 1 United States of America                                5390       0.284
 2 Palau                                                    896       0.323
 3 Israel                                                  4944       0.346
 4 Federal Republic of Germany                             2067       0.396
 5 Micronesia (Federated States of)                        1462       0.414
 6 United Kingdom of Great Britain and Northern Ireland    5372       0.429
 7 France                                                  5325       0.434
 8 Marshall Islands                                        1600       0.489
 9 Belgium                                                 5391       0.495
10 Canada                                                  5408       0.509
# ... with 189 more rows
```

```
by_year <- votes_per2 %>%
  group_by(year) %>%
  summarize(total = n(),
            percent_yes = mean(vote == "yes"))
```

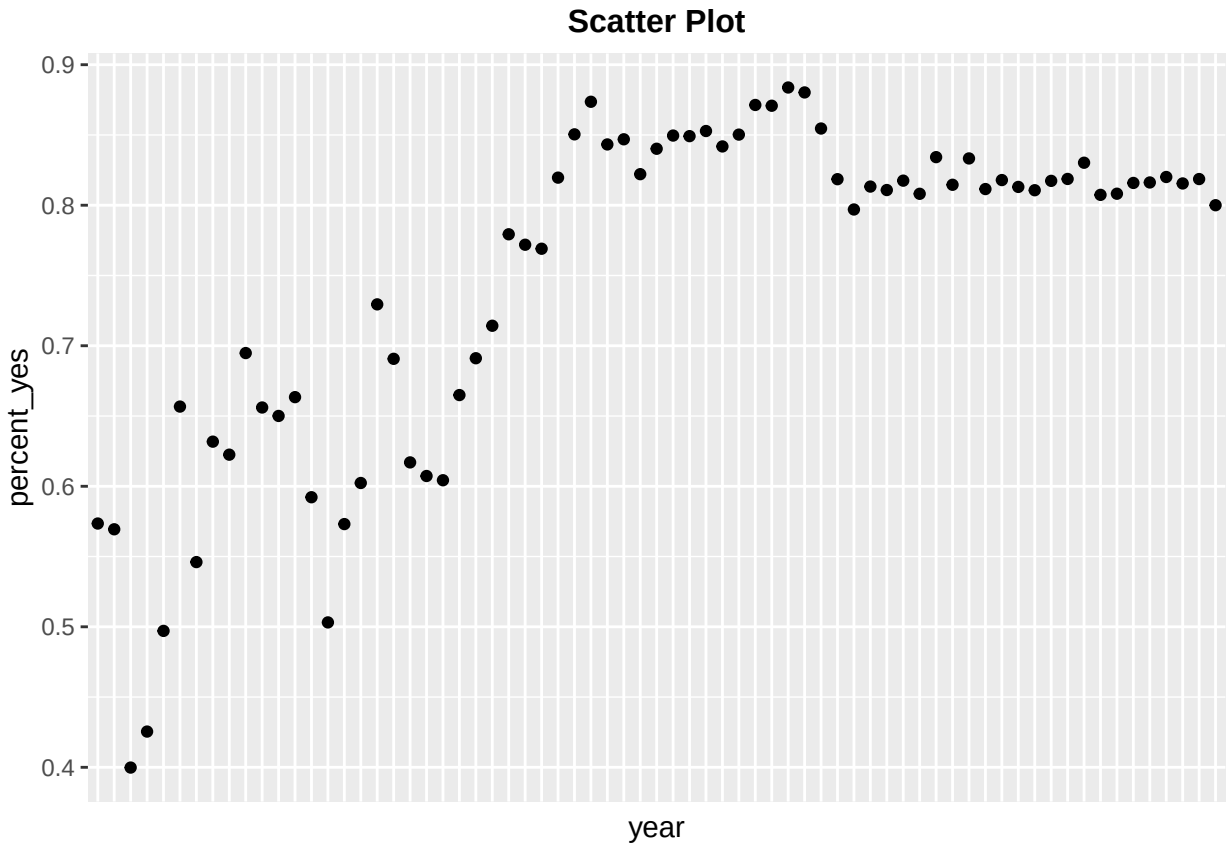
1.1 Commands for ggplot2 graphs

In this section some small changes have been made with respect to the commands listed in the Word sheet of the assignment. The changes are only due to graphic reasons (center the title, eliminate the automatic labels on the axes when they overlap etc...)

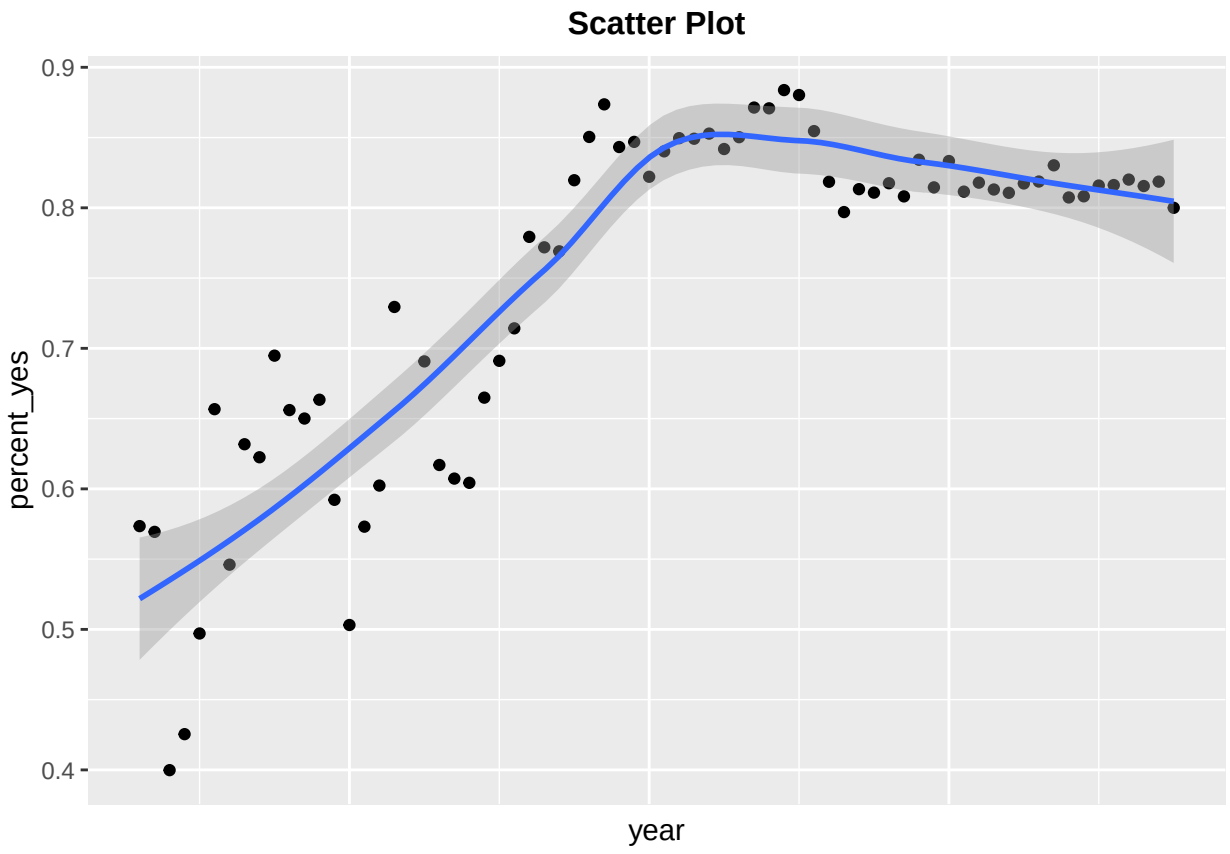
A `ggplot()` graph is reported below. `aes` wraps a list of aesthetic mappings to use for plot: here the mappings are the `x` and `y` axis to the variables `year` and `percent_yes` respectively. The `geom_smooth()` command draws the points. The `geom_smooth()` command does not work in the first case, as it was reported in the assignment. I had to change the `by_year` dataset by turning the `year` column into a column of type `Double` to make the smoother work. It performs a LOESS regression with default parameters (`span`=0.75 and `degree`=2).

Finally, a new table is created (`by_year_country`) by summarizing `total` and `percent_yes`, grouping by both `country` and `year`.

```
mynamestheme <- theme(plot.title= element_text(family = "Helvetica", face = "bold", size = (12)))

ggplot(by_year, aes(year, percent_yes)) +
  geom_point() +
  geom_smooth() + mynamestheme + theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())+
  ggtitle("Scatter Plot")
```



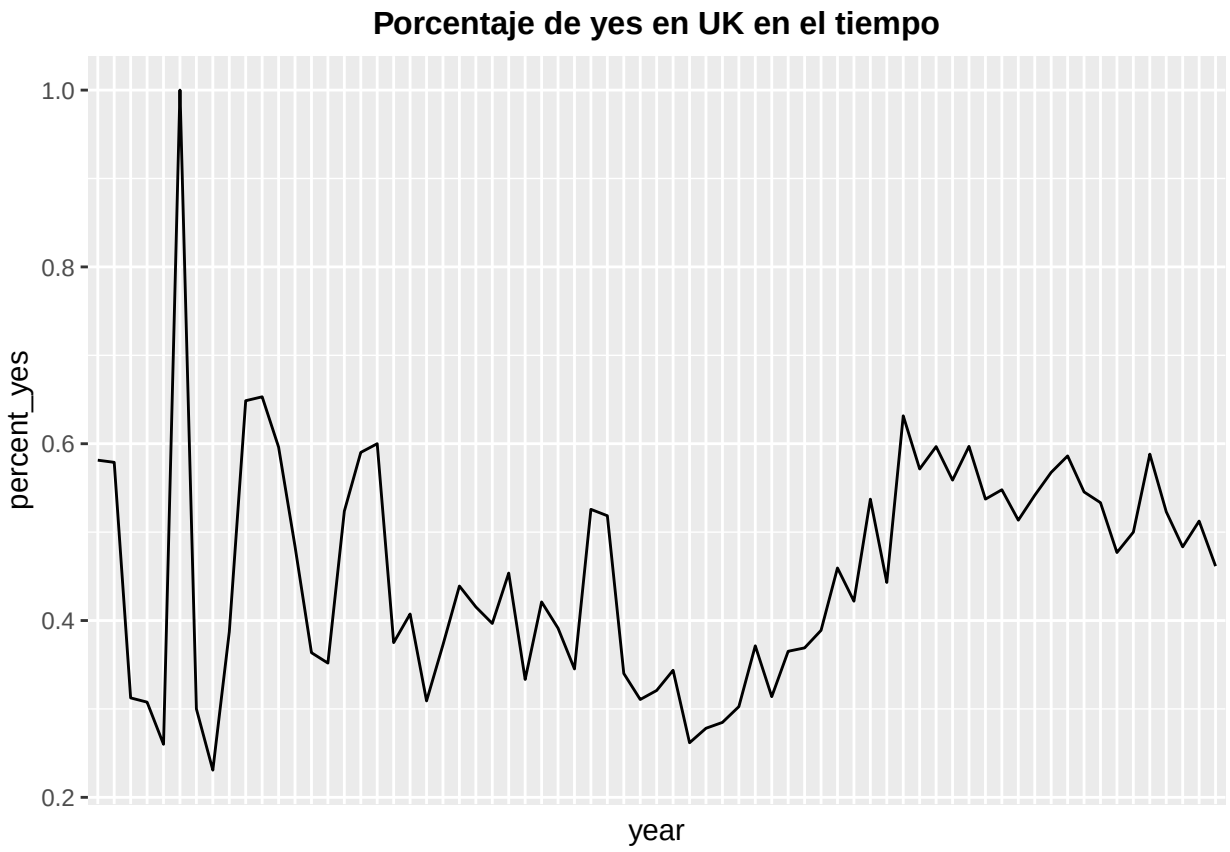

```
by_year <- votes_per2 %>%
  group_by(year) %>%
  summarize(total = n(),
            percent_yes = mean(vote == "yes"))%>%  mutate(year=as.numeric(year))

ggplot(by_year, aes(year, percent_yes)) +
  geom_point() +
  geom_smooth() +  mynamestheme +
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())+
  ggtitle("Scatter Plot")
```




```
by_year_country <- votes_per2 %>%
  group_by(country, year) %>%
  summarize(total = n(),
            percent_yes = mean(vote == "yes"))
```

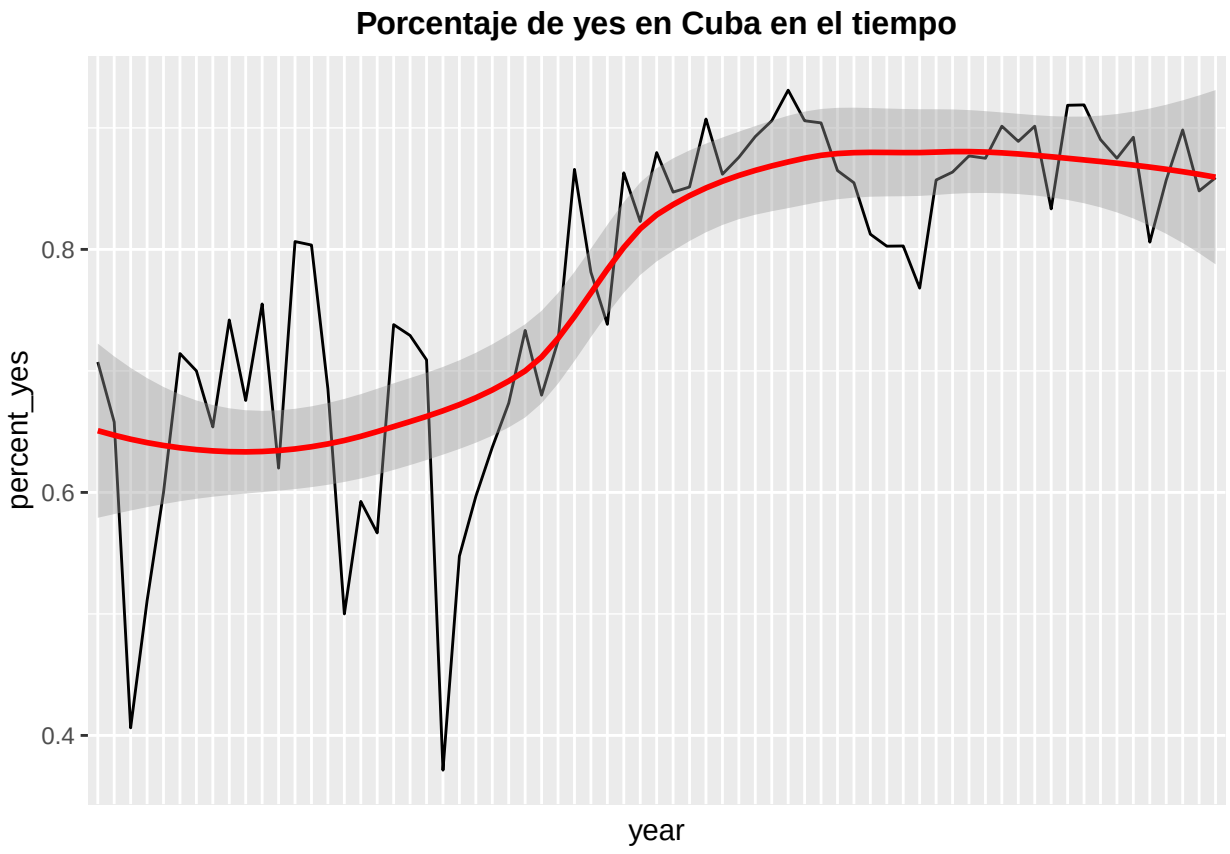
The following graph draws the profile of the `country` "United Kingdom of Great Britain and Northern Ireland" in time in relation to the `percent_yes` variable. The `group=1` parameter wrapped up in `aes` is used to indicate that a single country profile has been measured. `geom_line()` draws a line by joining the (x, y) pairs of the variables in `aes()`.

```
UK_by_year <- by_year_country %>%
  filter(country == "United Kingdom of Great Britain and Northern Ireland")

ggplot(UK_by_year, aes(year, percent_yes,group=1)) +  mynamestheme +
theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())+
     geom_line()+ggtitle("Porcentaje de yes en UK en el tiempo")
```



The same profile is drawn for Cuba. A smoothing curve is overlayed by adding the `geom_smooth()` command after the `geom_line()`. The entire `ggplot2` system is based on the idea of layers and mapping of layers to elements of the graph, therefore elements are added to a plot by using the `+` simbol and the wished graph element.

```
cuba_by_year <- by_year_country %>%
  filter(country == "Cuba")

ggplot(cuba_by_year, aes(year, percent_yes,group=1)) +  mynamestheme +
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())+
  geom_line()+ geom_smooth(col="red") + ggtitle("Porcentaje de yes en Cuba en el tiempo")
```

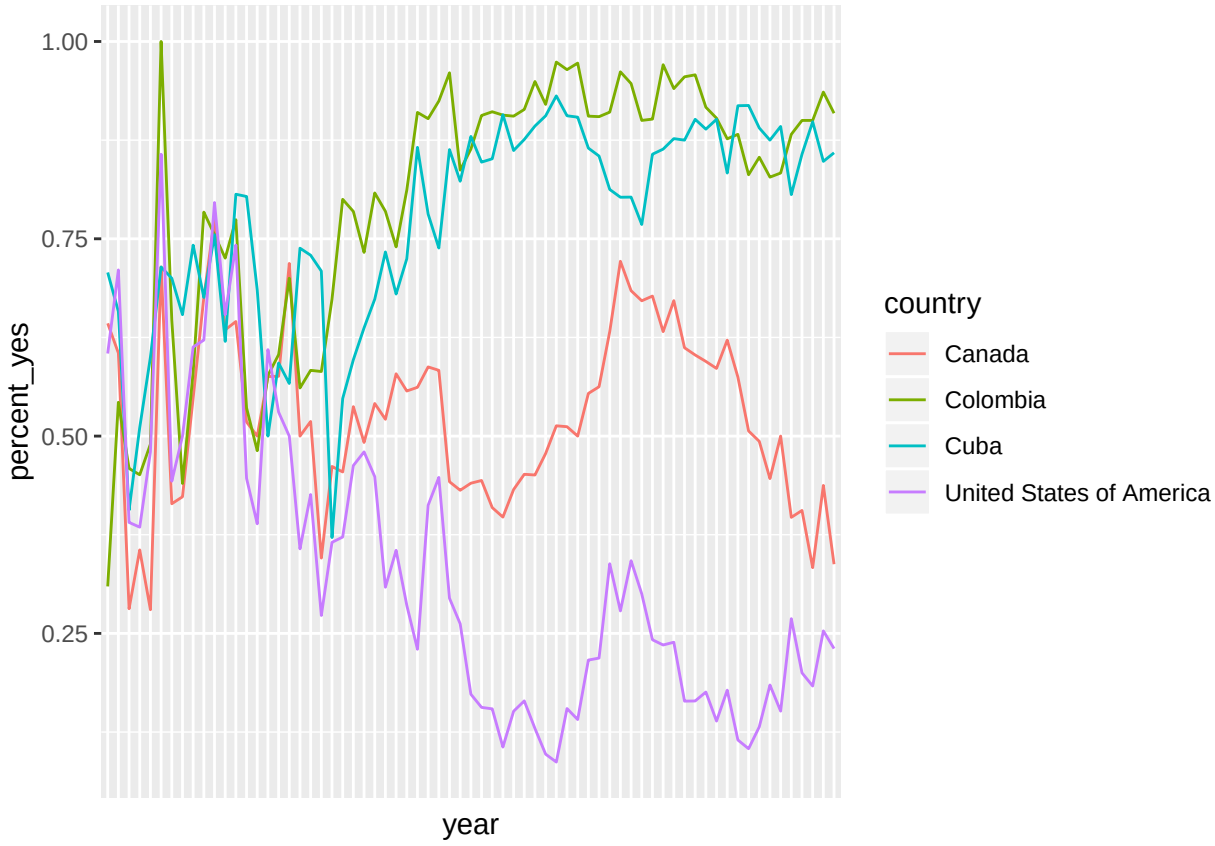


In the following commands a different way of expressing a `filter` condition can be seen. First of all, two vectors of characters are defined. Subsequently, the two vectors are used in the `filter` command: by using the `%in%` condition, records are filtered by looking to the `country` variable (variable to the left of `%in%` symbol) and checking whether its value is contained in the `paises` and `paises2` vector respectively.

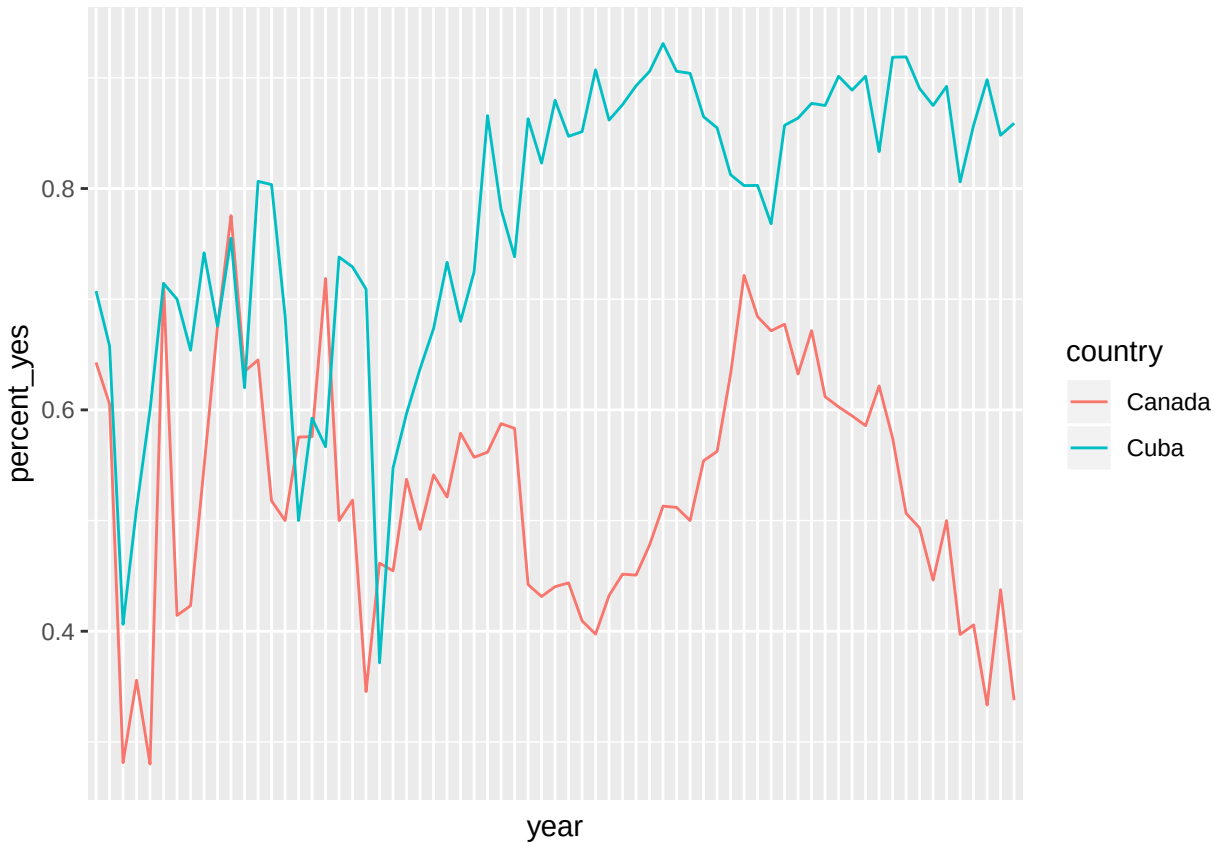
```
paises = c("United States of America","Canada","Cuba","Colombia")

paises2 = c("Canada","Cuba")

filtered_4_countries <- by_year_country %>%
       filter(country %in% paises)

filtered_2_countries <- by_year_country %>%
       filter(country %in% paises2)
```

Below it can be see how to follow different profiles over time by country. Within `aes()` both the `color` and the `group` element of the graph have to be set equal to the variable based on which we want to break down the line to get a more in-depth view of the data. If only `group=country` would be specified, the graph would have multiple lines, however they would all be of the same colour. If colour would not be wrapped in `aes()`, the command would give an error, since `R` would be looking for a `System R` color named `country`, which does not actually exist. If a `geom_smooth()` command would be added, the smoothing curves would also be broken down according to the Country (and the colour set accordingly).

```
ggplot(filtered_4_countries, aes(year, percent_yes, color = country,group=country)) +
       geom_line() + theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())
```

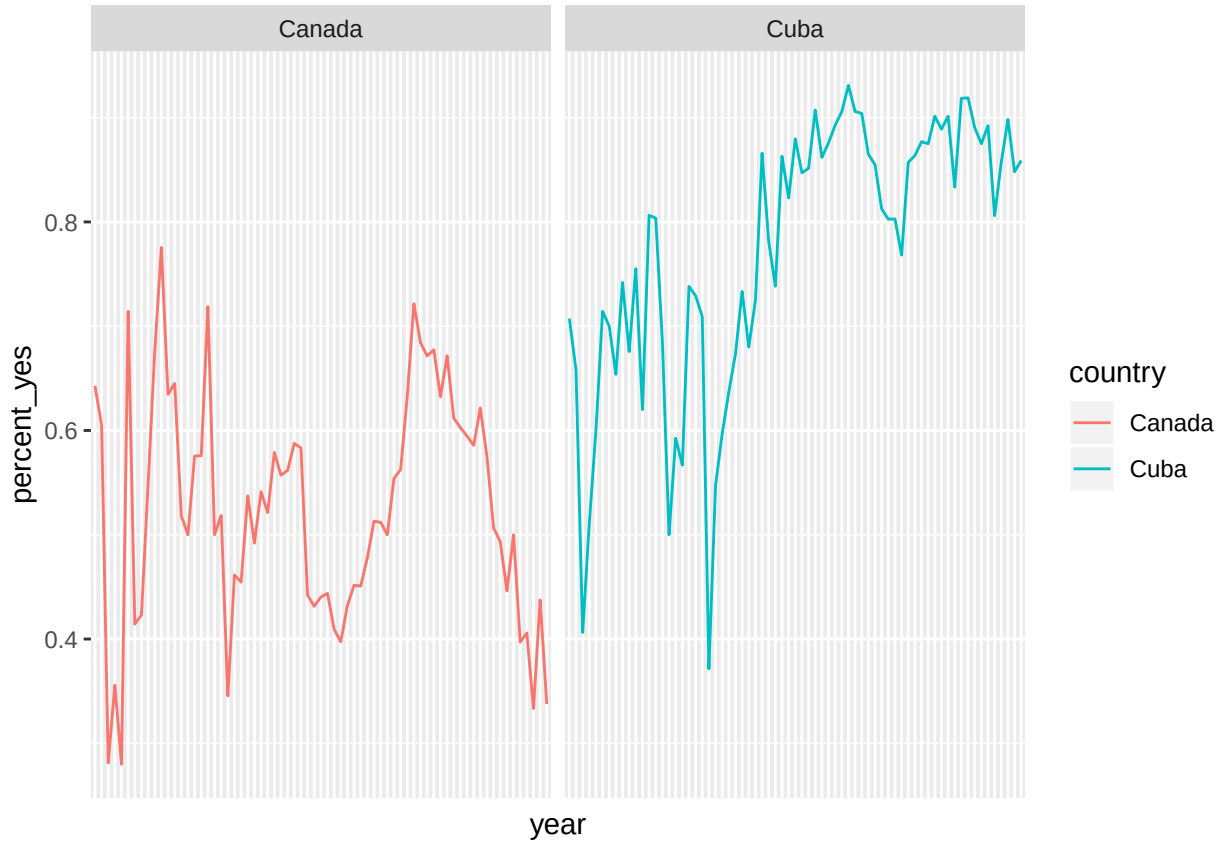



```
ggplot(filtered_2_countries, aes(year, percent_yes, color = country,group=country)) +
  geom_line() +
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())
```

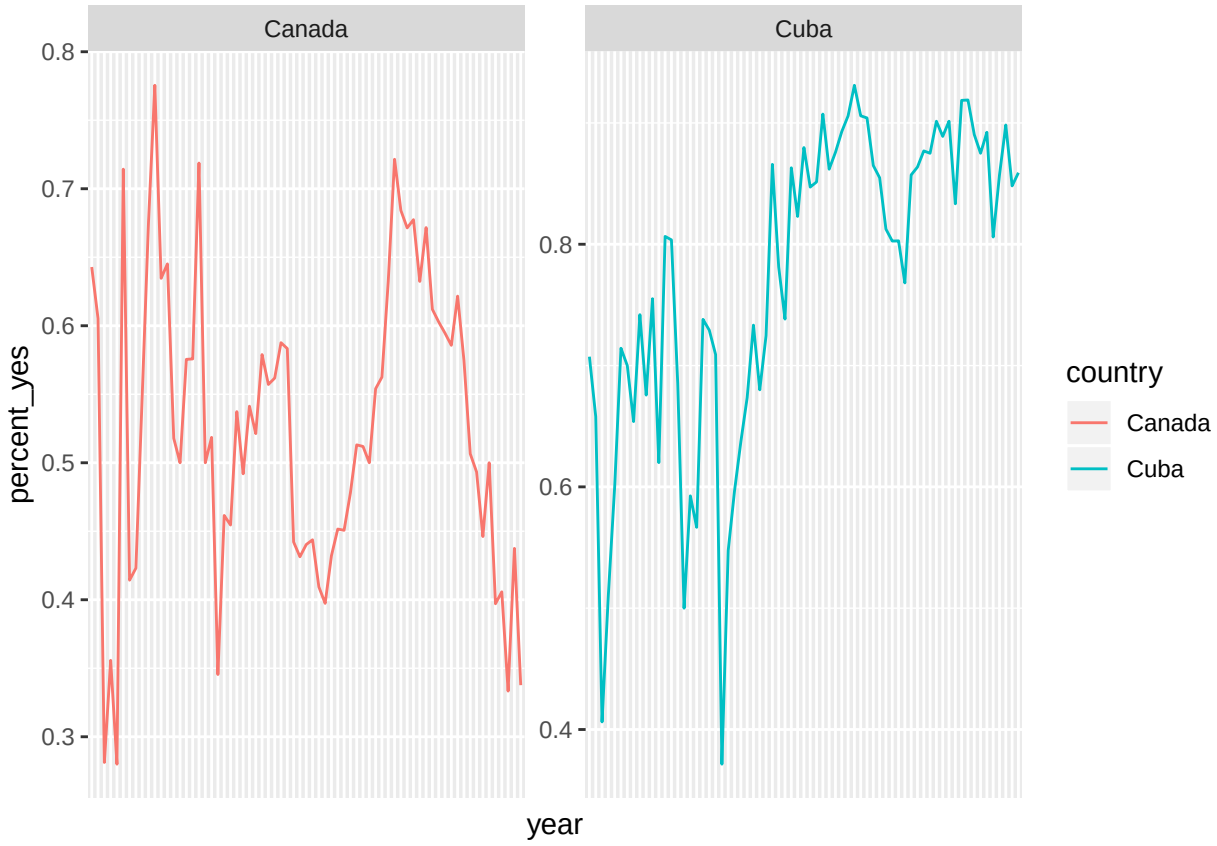


Here below the same graph above is split up into smaller windows, placed side by side, based on the variable in the `facet_wrap(~ ..)` layer. Lines belonging to different countries are showed in different panels. The difference between the first and the second graph is that by setting the option `scales="free_y"` in the second one, the `y` axis is freed from being equal in the two paired graphs. This is particularly helpful when the `y` variable has quite different values among different countries. The same considerations apply for the third and fourth graph.

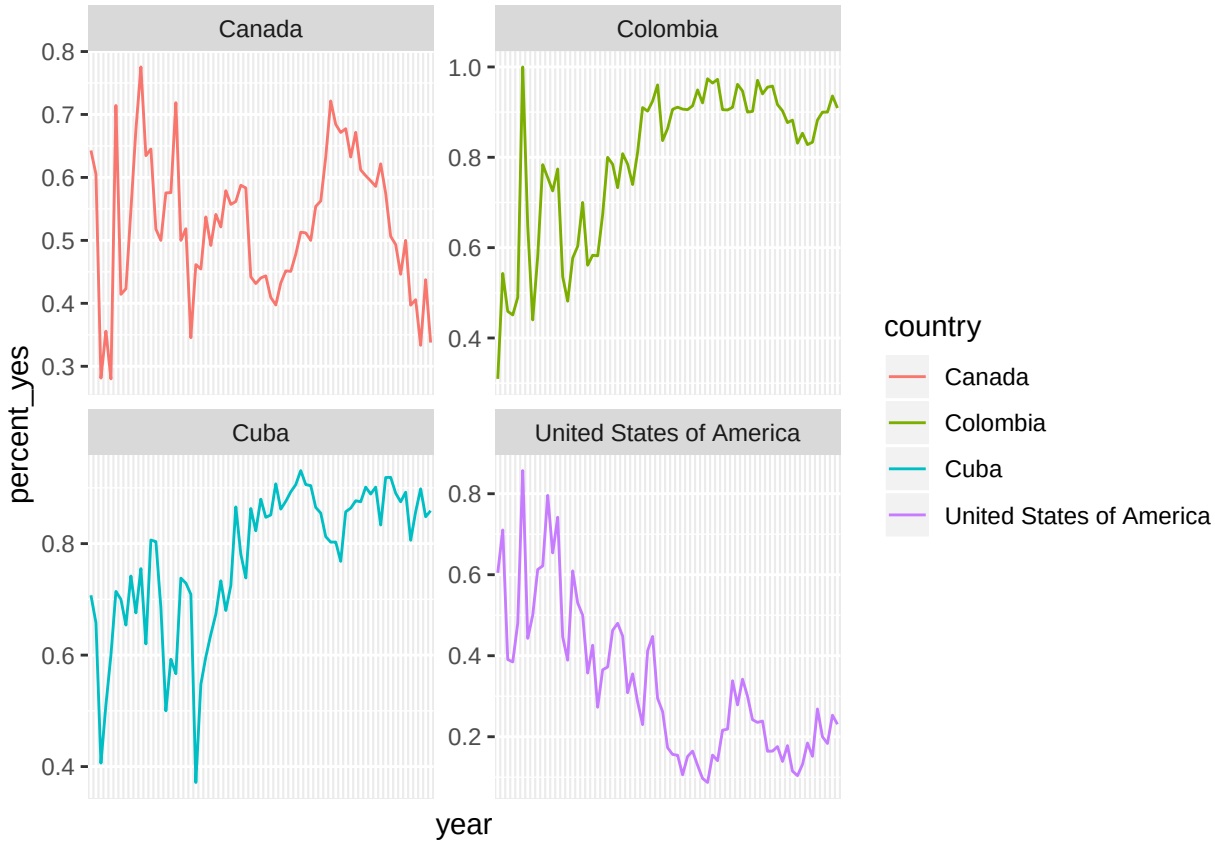
```
ggplot(filtered_2_countries, aes(year, percent_yes, color = country, group = country)) +
  geom_line() + facet_wrap(~ country)+
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())
```



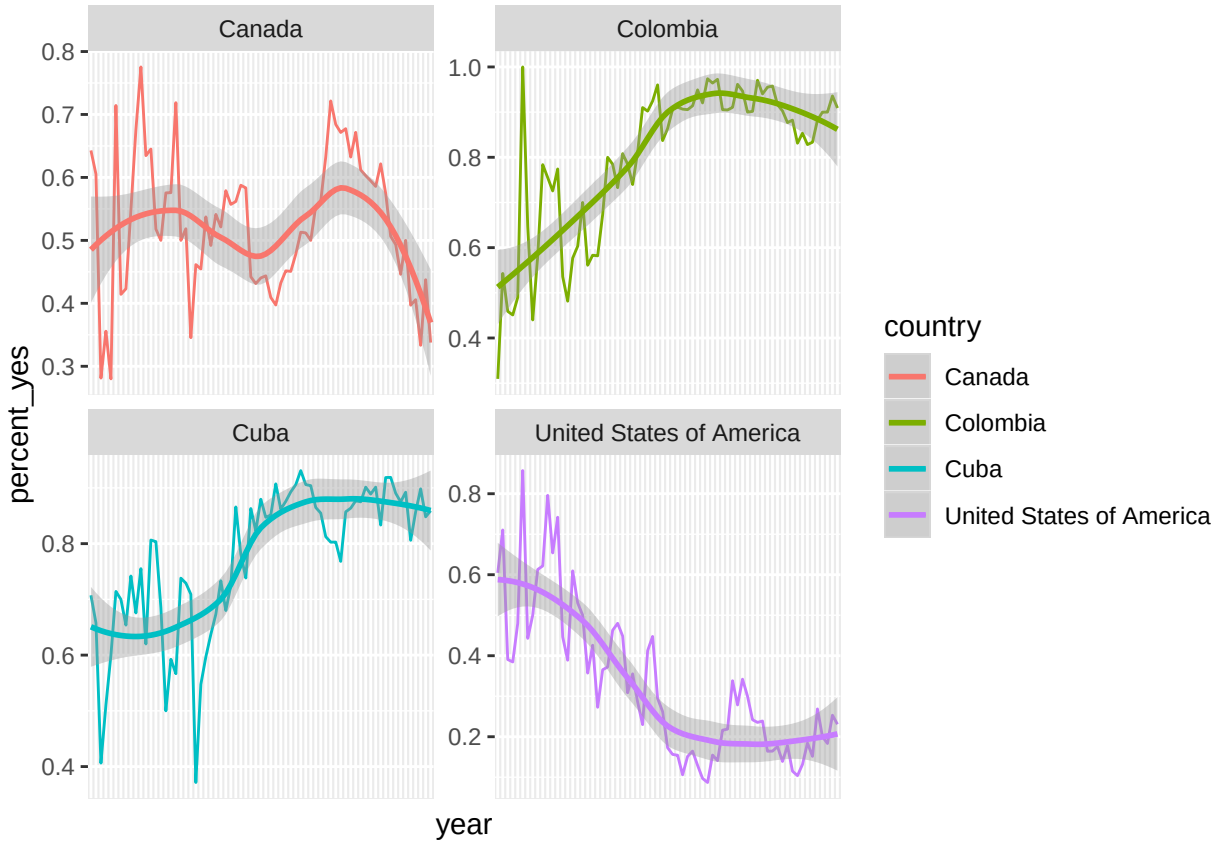

```
ggplot(filtered_2_countries, aes(year, percent_yes, color = country, group = country)) +
  geom_line() + facet_wrap(~ country,scales="free_y") +
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())
```




```
ggplot(filtered_4_countries, aes(year, percent_yes, color = country, group = country)) +
  geom_line() +  facet_wrap(~ country,scales="free_y") +
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())
```



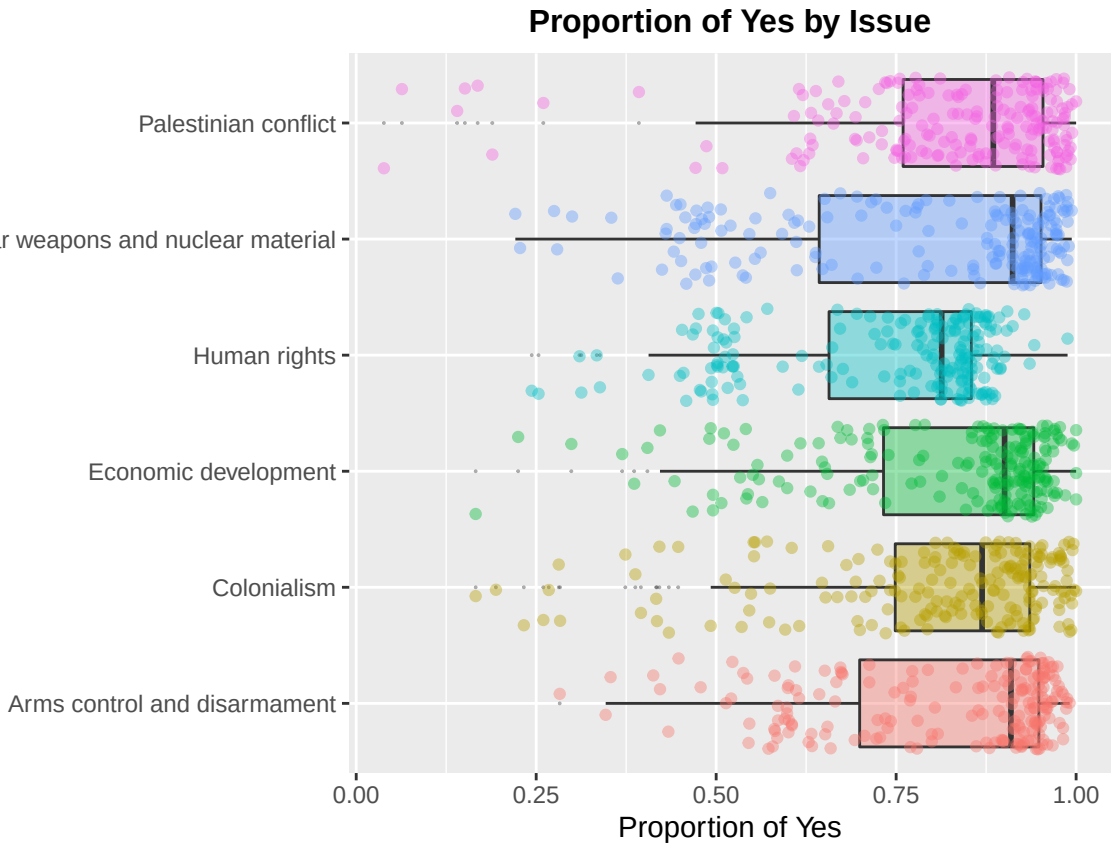

```
ggplot(filtered_4_countries, aes(year, percent_yes, color = country, group = country)) +
  geom_line() + geom_smooth() + facet_wrap(~ country,scales="free_y") +
  theme(plot.title = element_text(hjust = 0.5),
  axis.text.x=element_blank(), axis.ticks.x=element_blank())
```


Canada
Colombia
Cuba
United States of America
percent_yes
year
country
Canada
Colombia
Cuba
United States of America

2 Some more exploratory data analysis

The following graph analyzes how the proportions of "yes" votes[3] changes according to the issue which is discussed. A `jitter()` was used so that the proportions of yes from different Countries for the same issue did not appear as points superimposed on the whiskers of the boxplot. Some noise is therefore randomly added. Transparency (`alpha=0.4`) has been added to the boxplots so that the points behind could be visible.

```
d<- votes_issues%>% group_by (country, issue)%>% summarize(Prop=mean(vote=="yes"))


ggplot(d)+ geom_boxplot(aes(x=issue, y=Prop, fill=issue), alpha=0.4, outlier.size = 0)+
  geom_jitter(alpha=0.4, aes(x=issue, y=Prop, col=issue))+
  theme(plot.title = element_text(hjust = 0.5))+
  ggtitle("Proportion of Yes by Issue")+mynamestheme +
  ylab("Proportion of Yes") + xlab("") +
  theme(legend.position = "none")+ coord_flip()
```



Looking at the median only, we can see that the issues on which the Countries generally agree more are "Arms, control and disaemament", "Nuclear weapons and nuclear material" and "Economic development". Looking at the dispersion of the proportions by focusing on the difference between the first and third quartiles, it can be see that the most dispersed proportions (=most discordant votes) concern "Nuclear weapons and nuclear material" which despite having a very high median also has great dispersion (=many proportions of yes are really low, and "Arms, control and disaemament" (for which the same considerations apply). The greater the dispersion highlighted by the boxplots (and the points plotted on them) the more discordant is the issue discussed. As for the Palestinian conflict, it is an issue which puts many countries in agreement (most of the points are concentrated between 60% and 100%) but there are 7 Countries which stand out, with a very different position in this regard (proportion <26%). In order to see which Countries are those, the following command is run:

[3] 6 proportions for each Country have been computed, one for each issue

```
d%>% filter(issue =="Palestinian conflict" & Prop<=0.27)
```

```
# A tibble: 7 x 3
# Groups:   country [7]
  country                          issue                  Prop
  <chr>                            <chr>                 <dbl>
1 Israel                           Palestinian conflict 0.0636
2 Marshall Islands                 Palestinian conflict 0.189
3 Micronesia (Federated States of) Palestinian conflict 0.151
4 Nauru                            Palestinian conflict 0.169
5 Palau                            Palestinian conflict 0.0385
6 South Sudan                      Palestinian conflict 0.26
7 United States of America         Palestinian conflict 0.140
```

The following command computes and displays the countries which have the highest percentage of abstention votes.

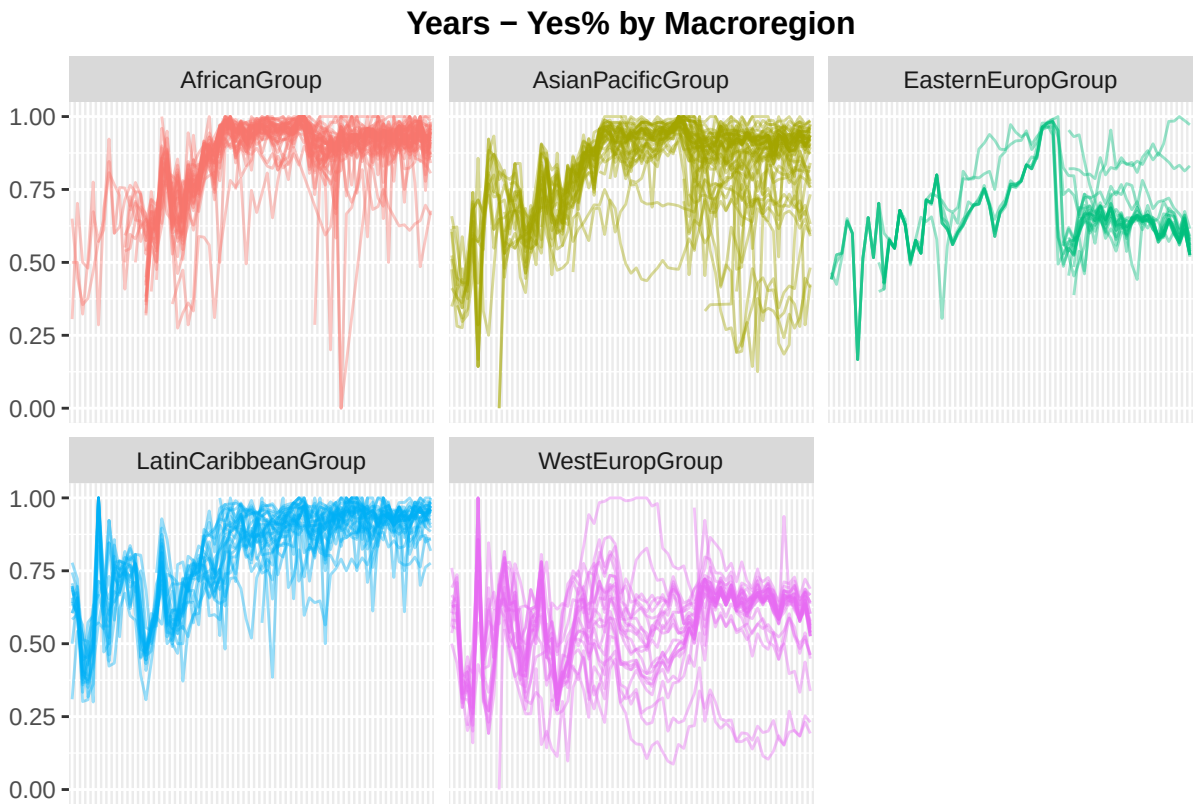
```
votes_issues%>% group_by(country)%>% summarize(PercAbstain=mean(vote =="abstain"))%>%
  arrange(desc(PercAbstain))%>% head(10)
```

```
# A tibble: 10 x 2
   country                                                PercAbstain
   <chr>                                                        <dbl>
 1 Federal Republic of Germany                                  0.430
 2 South Sudan                                                  0.391
 3 France                                                       0.354
 4 Japan                                                        0.344
 5 Tonga                                                        0.322
 6 Italy                                                        0.315
 7 Georgia                                                      0.311
 8 Republic of Korea                                            0.309
 9 United Kingdom of Great Britain and Northern Ireland         0.306
10 Belgium                                                      0.301
```

After creating 5 vectors, where each represents a set of countries belonging to the same macro area[4], some plots are drawn to see how the yes percentage changes over time according to the region.

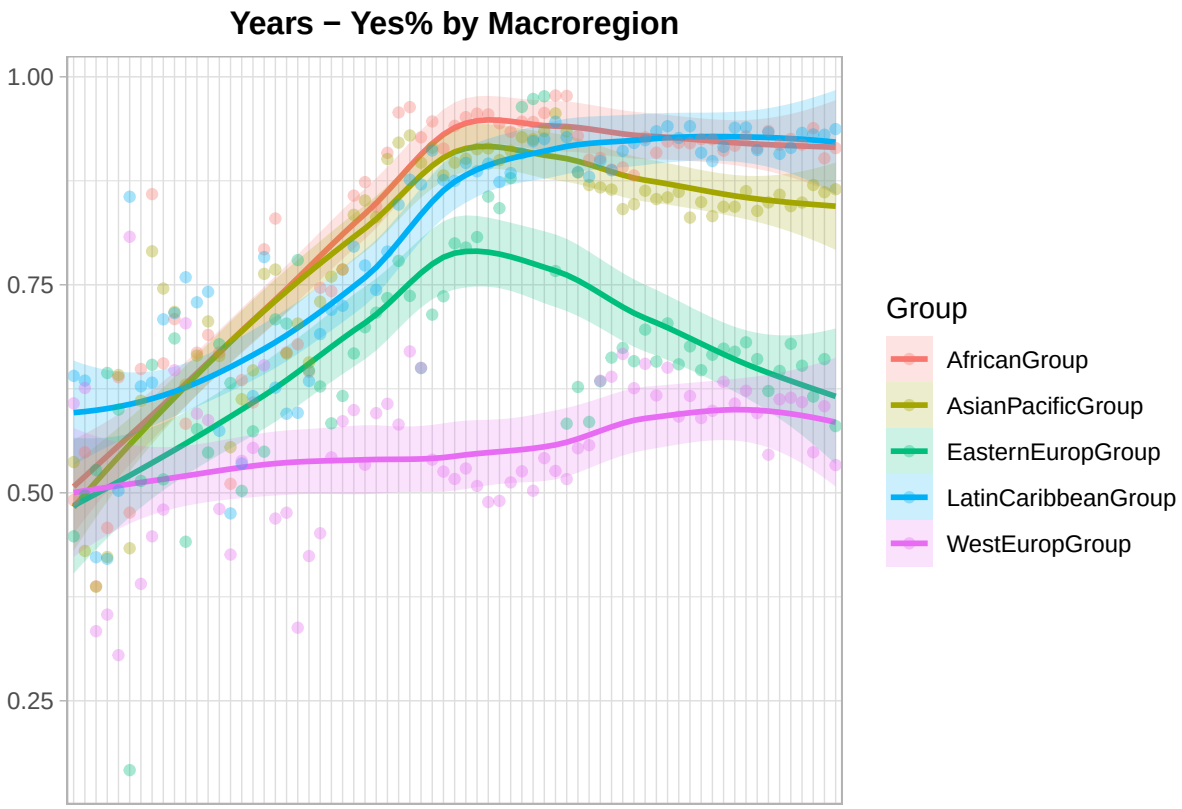
```
withGroup<- votes_per2 %>% mutate(Group = ifelse(country %in% Africa, "AfricanGroup",
            ifelse(country %in% Asia_and_Pacific_Group, "AsianPacificGroup",
          ifelse(country %in% Eastern_European_Group, "EasternEuropGroup",
          ifelse(country %in% LatinAmerican_and_CaribbeanGroup, "LatinCaribbeanGroup",
        ifelse(country %in% WesternEuropeanandOthers, "WestEuropGroup", "other"))))))

withGroup1<- withGroup %>% group_by(year, country, Group) %>%
  summarize(total = n(),
            percent_yes = mean(vote == "yes"))%>% filter(Group!="other")

ggplot(withGroup1, aes(year, percent_yes, color = Group, group = country)) + mynamestheme+
  geom_line(alpha=0.4) + facet_wrap(~ Group)   + xlab("") + ylab("") +
  ggtitle("Years - Yes% by Macroregion") + theme(legend.position = "none") +
  theme(plot.title = element_text(hjust = 0.5), axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```

[4]the original grouping can be found here: https://en.wikipedia.org/wiki/United_Nations_Regional_Groups



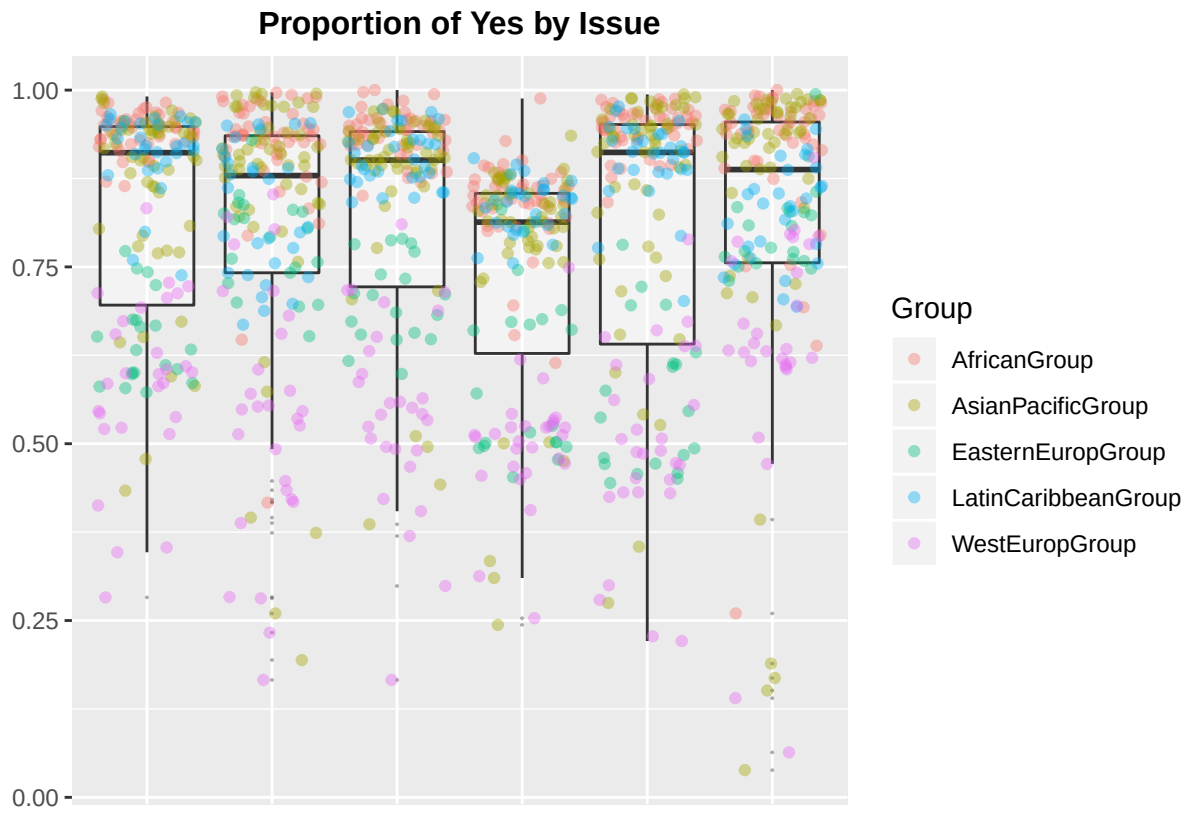
The plot above shows that countries in the same macro-region have similar voting behavior over time. The most uniform behavior seems to be that of the Latin and Caribbean Countries, while the most dissimilar voting behaviour is the one of the Western European Countries.

```
withGroup2<- withGroup %>% group_by(year, Group) %>%
  summarize(
            percent_yes = mean(vote == "yes"))%>% filter(Group!="other")

ggplot(withGroup2, aes(year, percent_yes, color = Group, group = Group)) + mynamestheme+
 xlab("") + ylab("") + theme_light() + mynamestheme + ggtitle("Years - Yes% by Macroregion") +
  theme(plot.title = element_text(hjust = 0.5), axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  geom_smooth(alpha=0.2, aes(fill=Group)) +
  geom_point(alpha=0.35)
```



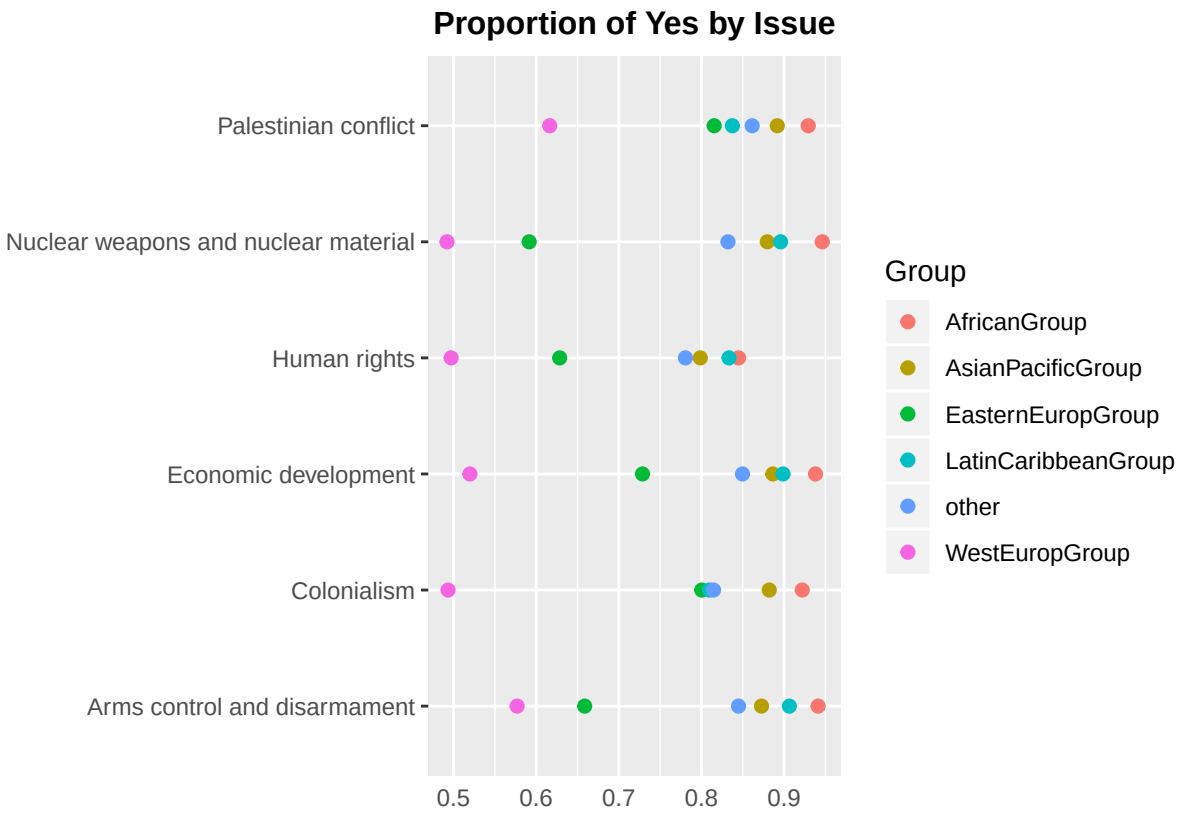
The graph above makes a smoothing of the percentages of "yes" over time for the macro-regions, and depicts them all in a graph in order to better appreciate the differences over time. The graph below shows the same boxplot

commented previously (percentage of yes by issue), however in this case each point is colored based on the macro region to which it belongs. What we are interested in seeing in this graph is whether the proportion of yes by issue changes according to the geographical area.
However, the graph shows that there is not any clear difference in the voting behaviour: the purple dots, which correspond to the West Europe group, is always the one that has lower proportions of favorable votes, whatever the issue, while the Asia-Pacific, the Caribbean and the African groups are those that always have the highest percentage, regardless of the issue.



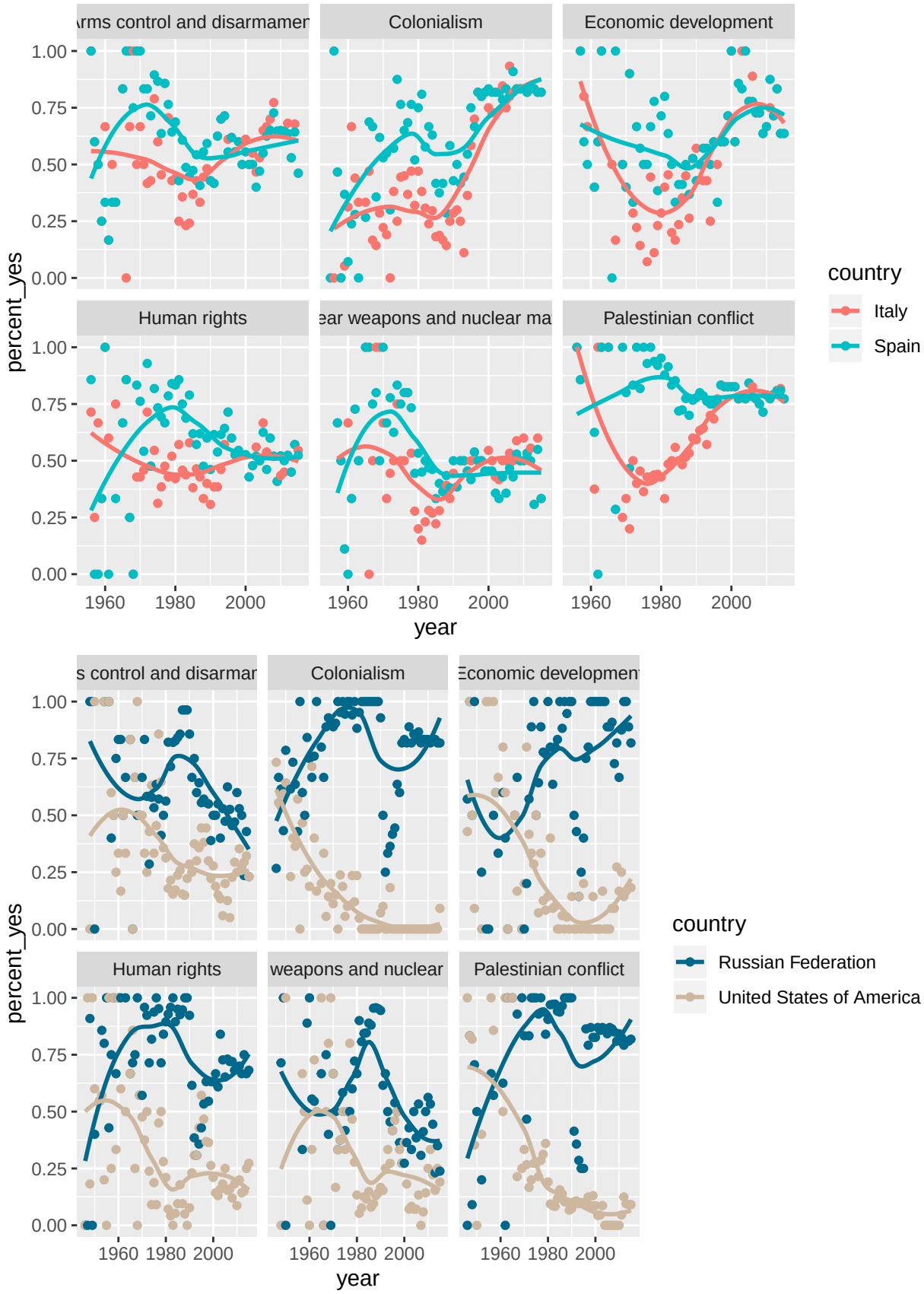
To see more clearly what happens, only the averages of the votes per issue per geographical group are represented. It can be seen that there is a quite clear pattern in the color of the points: the purple dots are the first on the left (lower percentage) and the orange ones always appear at the right-hand side (highest percentage for all the issues). Moreover, the green dots always follow the purple ones. It can be seen, therefore, that there is not a different pattern based on the issue.

```
summary<- wG%>% group_by(issue, Group)%>% summarize(Perc=mean(vote == "yes"))

  ggplot(summary)+ geom_point(aes(x=issue, y=Perc, col=Group), size=2)+
  theme(plot.title = element_text(hjust = 0.5),
        axis.ticks.x=element_blank())+
  ggtitle("Proportion of Yes by Issue")+
  mynamestheme + xlab("") + ylab("")+
  coord_flip()
```



Another interesting exploratory analysis is to follow the time profile of voting divided by issue and compare these trends for different countries. The pairs of countries which are compared are Italy-Spain and Russian Federation-USA. In the **first plot**, the voting pattern is quite similar, especially for "Colonialism" and "Economic Development" (the lines have a similar pattern).
In the **second plot**, it can be seen that the Russian Federation and the USA have quite a different stance over time, particularly for issues concearning "Colonialism", "Economic development" and "Palestinian Conflict".

```
votes_per2$year<- as.numeric(votes_per2$year)

 votes_per2 %>%
  filter(country== "Spain" | country== "Italy") %>% inner_join(un_roll_call_issues, by = "rcid") %>%
 group_by(year, issue, country) %>%
  summarize(percent_yes = mean(vote == "yes")) %>%
  ggplot(aes(year, percent_yes)) +
  geom_point(aes(col=country)) +
  geom_smooth(se = FALSE, aes(group=country, color=country)) +
  facet_wrap(~ issue)
```



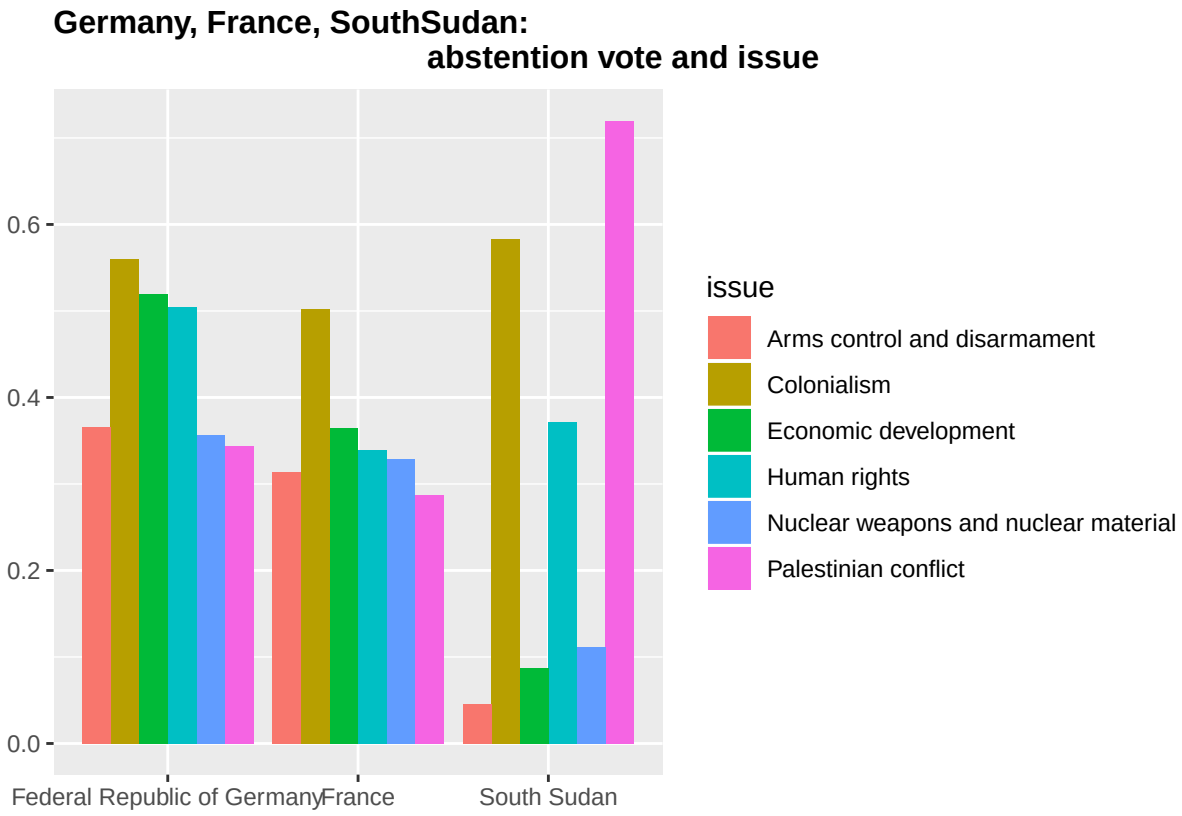
Finally, I looked for the countries with the highest abstention percentages, and I drill down to have a detailed view on the issues for which the country casted most of his abstension votes. The following Countries are the top three

with the higher percentage of abstaintion votes.

```
V<- votes_per2 %>% group_by(country)%>% summarize(abst_perc=mean(vote=="abstain"))%>%
  arrange(desc(abst_perc))
```

```
# A tibble: 3 x 2
  country                     abst_perc
  <chr>                           <dbl>
1 Federal Republic of Germany     0.418
2 France                          0.334
3 South Sudan                     0.316
```

```
coun<- c("Federal Republic of Germany", "France", "South Sudan")
votes_per2 %>% filter (country %in% coun) %>% inner_join(un_roll_call_issues, by='rcid') %>%
  group_by(country, issue)%>% summarize(abst_perc=mean(vote=="abstain")) %>%

ggplot(aes(x=country, y=abst_perc, fill = issue)) + geom_bar(stat = 'identity',
    position = position_dodge()) + mynamestheme+ ggtitle('Germany, France, SouthSudan:
                                        abstention vote and issue')+ xlab("")+ ylab("")
```



It can be seen that the behavior of France and Germany with respect to the absention votes is the same (most abstentions concern "Colonialism", followed by "Economic development" and "Human rights", while the percentage for the "Palestinian conflict" is for both the lower). The behavior of South Sudan is however really different: "Palestinian conflict" is the issue with the highest percentage, quite differently from the other two States, while "Economic development" has a much lower percentage in comparison to "Human rights", which again is quite different from what has been commented before.

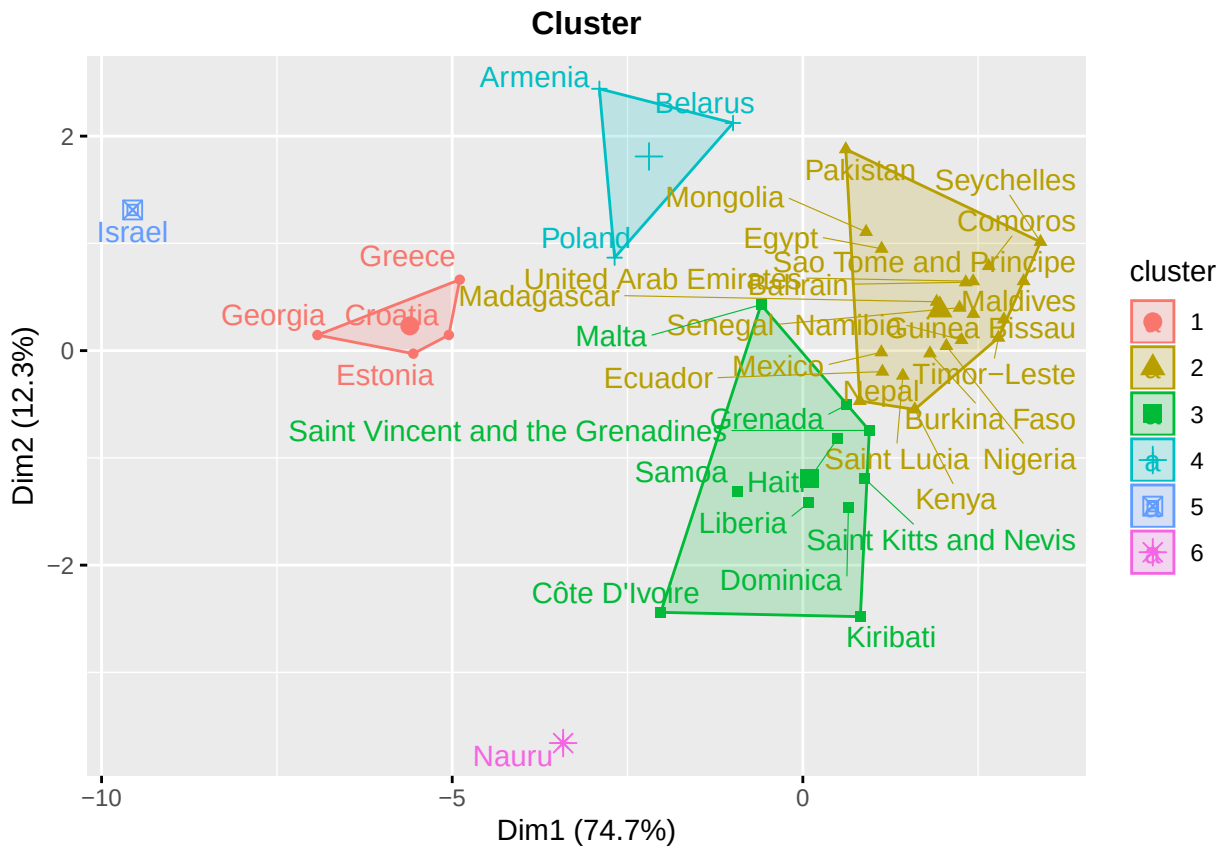
3 Cluster analysis

The task is to carry out a cluster analysis of our choice.
Since in the following sections it is required to do a cluster analysis for time series (yes% over the years) and another one which clusters records based on the issue, in this section the idea is to perform a cluster analysis based on different criterions than the ones which are required in the following. Note that **scaling** is not needed in any of the subsequent analysis, since all the values on which the distance measure is computed are proportions (hence, since they have a 0-1 range they have a uniform unit of measure and therefore distance is not affected by it).

The aim of the cluster analysis which follows is to **find groupings of Countries based on both the proportion of "yes" and "abstain votes" for the 6 issues discussed**. It means that Countries are considered as records to be clustered, and for each Country 12 variables are available (the proportion of yes-votes and abstain-votes for the 6 issues). Let's first create the dataset we need for this analysis and visualize the first 6 records and the first 2 columns:

```
                   Arms control and disarmament_percent_abstain
Afghanistan                                           0.09253066
Albania                                               0.11774194
Algeria                                               0.13245033
Andorra                                               0.17620137
Angola                                                0.08308605
Antigua and Barbuda                                   0.04585799
                   Arms control and disarmament_percent_yes
Afghanistan                                       0.8874025
Albania                                           0.6322581
Algeria                                           0.8642384
Andorra                                           0.5995423
Angola                                            0.9139466
Antigua and Barbuda                               0.9526627
```

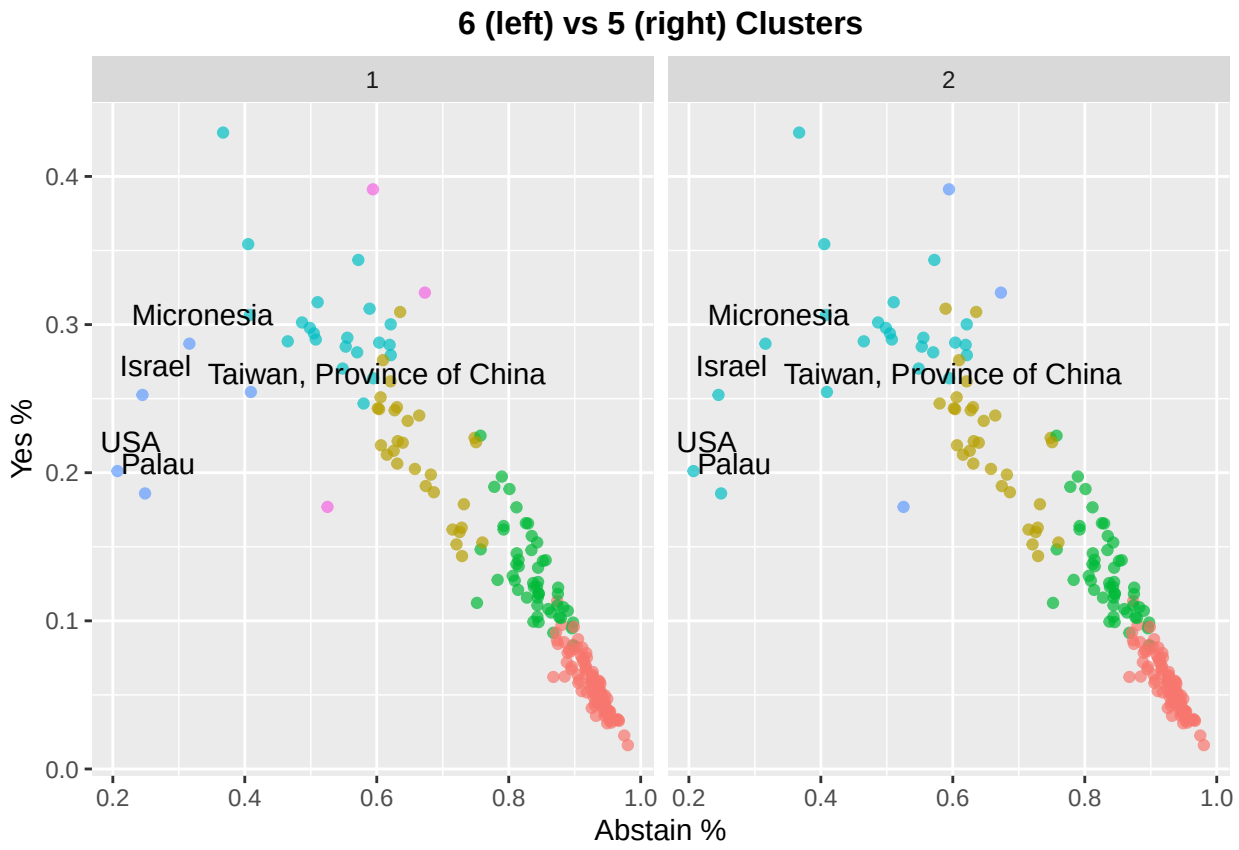
```
Dissimil <- dist(Dataset, method= "euclidean")

set.seed(5)
uu<- sample(c(1:199), 40)
cluster <- hkmeans(x= Dataset[uu, ] , k = 6)

fviz_cluster(object = cluster, pallete = "jco", repel = TRUE) + mynamestheme+ ggtitle("Cluster")+
 theme(plot.title = element_text(hjust = 0.5))
```



This results should be compared with those in the section **Clustering Countries based on the issue**, since the two tasks are quite similar. The difference is that here both the "yes" and the "abstain" proportions are taken into account (grouped by issue), therefore the number of attributes are 12 rather then 6 (for each issue two proportions are reported here).
Only a subset of Countries is displayed for graphic reasons.
"Naurau" and "Israel" appear to be really different from all the other Countries with respect to the "yes" and "abstain" choices. They will look like outliers also later on, when only the yes-vote is considered, thus their outlier feature do not only concearn their "abstain" voting behaviour. It is difficult to make exact comparisons among clusters since only a random subset of Countries are displayed in the two cases, but it can be seen that, broadly speaking, the same Countries are clustered together in the two cases, which means that Countries with similar "yes"-vote pattern also have a similar "abstain"-vote pattern, otherwise they would no longer be clustered together when the new attributes (concerning the "abstain" vote) are added.

Furthermore, another possibility for a cluster analysis is to consider only two variables for each Country: the percentage of "yes"-votes and that of "abstain"-votes. These are obtained by averaging "yes" and "abstain" for all issues and all years, for any given Country. In this way the variables on which the cluster is based are only two quantitative variables, or the two percentages mentioned above.
The graph below shows the clusters obtained (the `x` and `y` axes show the two variables used). The countries labeled are those that change clusters according to the `k` (number of clusters) used: if the clusters are 6, those Countries cluster together and form a separate cluster; if instead the number of clusters is limited to 5 then these Countries are agglomerated to the "closest" cluster.
The three points that cluster together in both cases (pink points on the left plot) are "Narau", "South Sudan" and Tonga. It can also be seen in the previous plot (where `x` and `y` were the first two PCs) that both "Israel" and "Narau" were quite outliers; however in the previous plot we could not see all of the Countries (like USA, Tonga etc) because only random subset of points had been selected.

```
      Nauru South Sudan       Tonga
        122         162         177
```


6 (left) vs 5 (right) Clusters
1
2
Micronesia
Israel
Taiwan, Province of China
USA
Palau
Micronesia
Israel
Taiwan, Province of China
USA
Palau
Yes %
Abstain %
0.0
0.1
0.2
0.3
0.4
0.2
0.4
0.6
0.8
1.0

4 Time series clustering

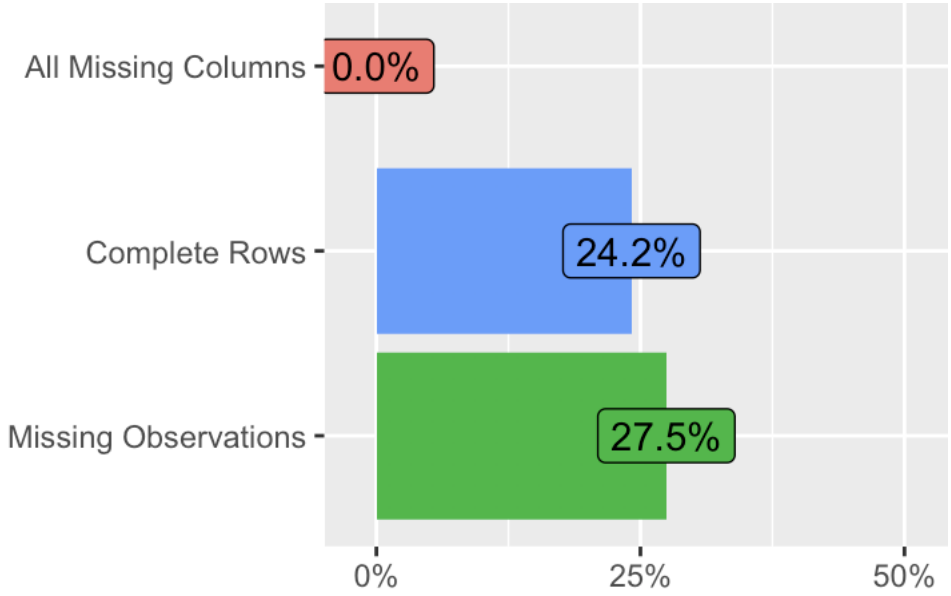
In what follows a clustering based on time series is performed. Clustering based on time series consists of considering the yes-no-abstain vote over time as a record representing a Country (the years from 1946 to 2015 are the 70 "attributes" considered, and the percentage of yes votes in the corresponding year is the value), and Countries which have a similar pattern over time are considered similar and likely to belong to the same cluster. The first step is to work out an appropriate distance/similarity metric, which is a key choice of huge impact on the results of clustering, and the second step is to use existing clustering techniques, such as k-means and hierarchical clustering, to find clustering structures.
The first thing to do is to prepare the data matrix as mentioned above, and then use this data matrix to compute a distance matrix based on some criterion.

```
Timeseries<- withGroup1%>% select(year, percent_yes)%>% spread(key=year, value=percent_yes)
Timeseries[1:3, 14:18]
```

```
# A tibble: 3 x 5
  `1958` `1959` `1960` `1961` `1962`
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1  0.548  0.589  0.685  0.547  0.636
2  0.548  0.679  0.630  0.491  0.581
3 NA     NA     NA     NA      0.535
```

If we take a look at the data we see that many values are NAs, because the voting percentages of many countries are not available (for various reasons) especially in the years from 1046 to 1970.

By running the command `plot_intro(Timeseries)` from the `DataExplorer` library it is possible to have a better look at the new dataset and at the missing values. It can be seen that there is not any completely missing column, but there are only 24.2% complete rows (i.e Countries for which the entire voting record from 1946 to 2015 is available) and a total of 27.5% missing observations (i.e Countries for which the vote percentage for some years is not available).

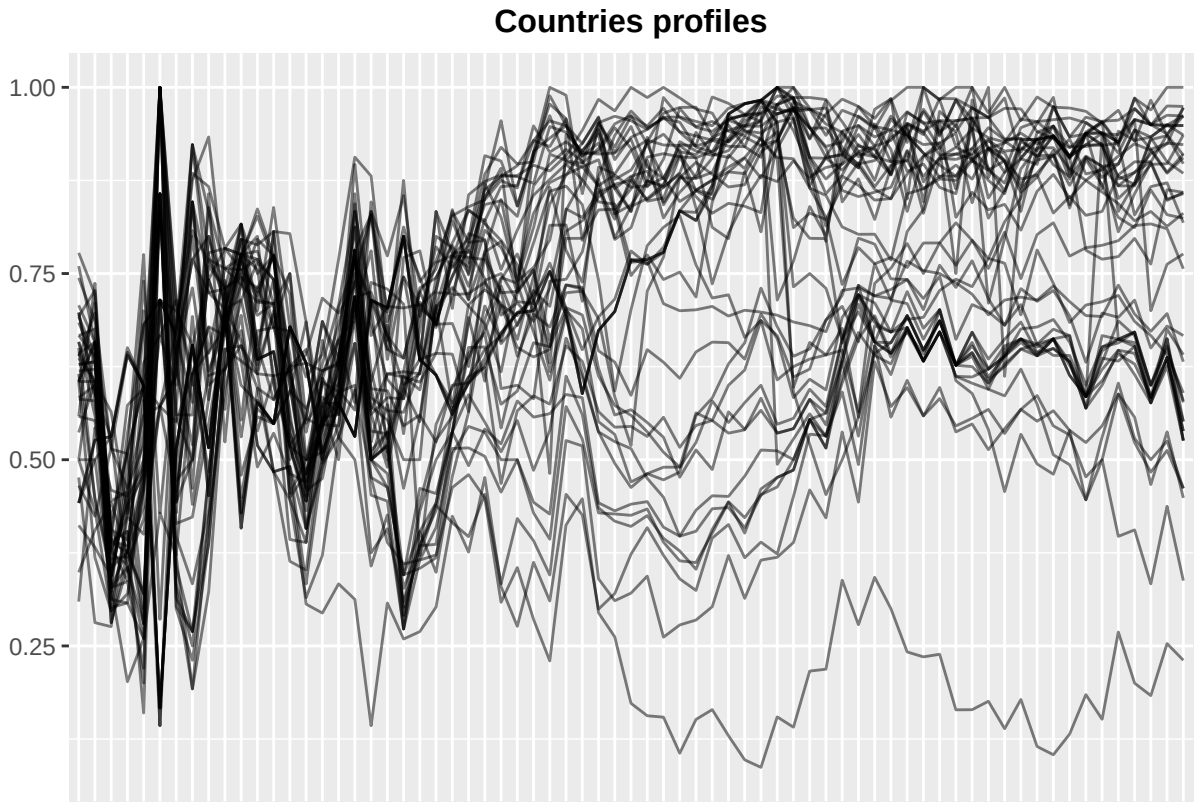


4.1 Dynamic Time Warping

Dynamic Time Warping (DTW) (`dwt` package) is a method to measure distance among time series which finds optimal alignment between two time series. This means that time series which have the same trend, with only a few years lag, will be considered very similar (for example `serie2` obtained as `lag(serie1, 2)` will be considered

identical since they have the same increments/decrements over time with only a two period lag, hence they could be perfectly overlapping if one of the two would be shifted parallel to the x axis). It would be different to consider a Euclidean distance (difference in the voting percentages of each year squared and added up for all the years), because `serie 1` and `serie2` will not result in a null distance in this case (because they do not have the same percentage for the exact same year, which is the only case in which the Euclidean distance would be null).
Given the difficulty of working with missing values (the `dist()` command returns an error) I will start by removing all the countries that have at least a missing percentage. The remaining Countries are 43, and a cluster analyisis will be performed using only those units.
A **k-means** clustering algorithm (non-hierarchical clustering) will first be used. To get an idea of how many clusters to use we plot the time profiles of the countries and see if it is possible to identify some groupings.

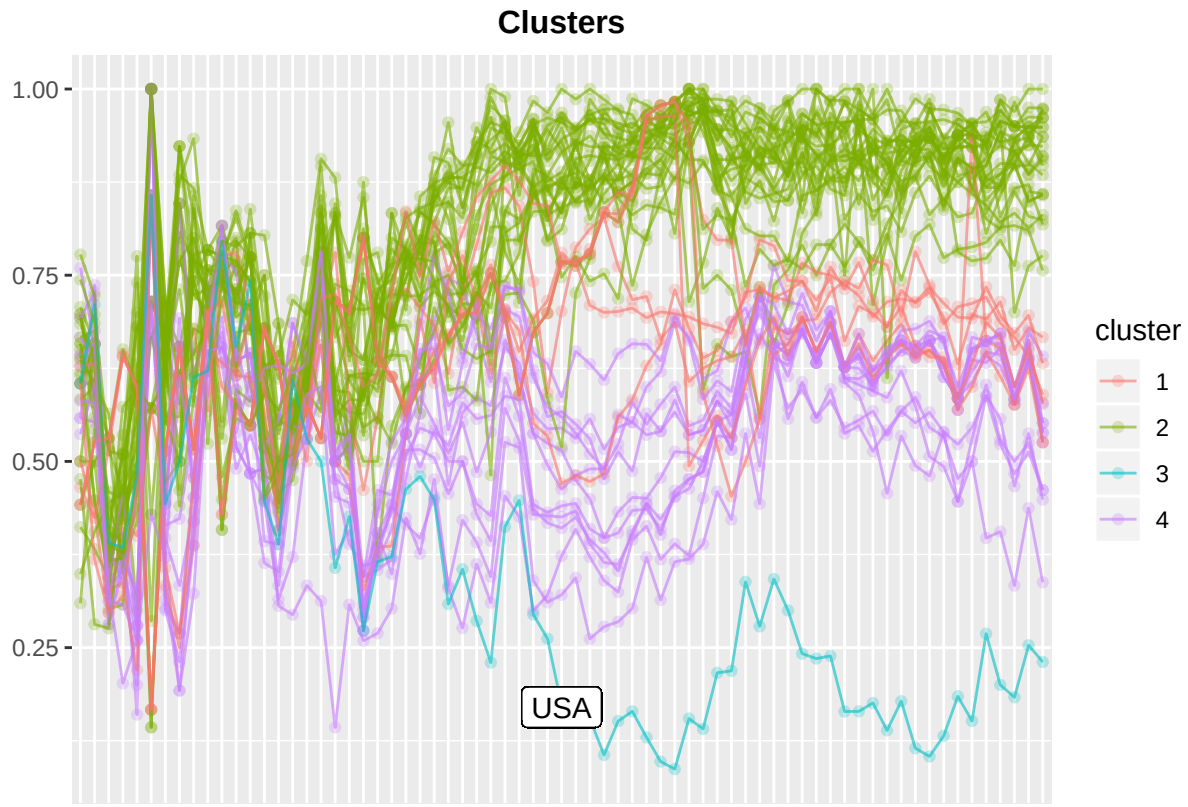
```
library(dtw)
Timeseries<- Timeseries %>% remove_rownames %>% column_to_rownames(var="country")
Timeseries2<- na.omit(Timeseries)

nomi<-row.names(Timeseries2)
DataPlot<- Timeseries2%>% gather(key=key, value=value)%>% mutate(Country=rep(nomi, 69))
ggplot(aes(x=key, y=value), data=DataPlot)+ geom_line(aes(group=Country), alpha=0.5)+
  theme(plot.title = element_text(hjust = 0.5), axis.ticks.x=element_blank(),
        axis.text.x=element_blank())+ xlab("")+ ylab("")+
  mynamestheme+ ggtitle("Countries profiles")
```



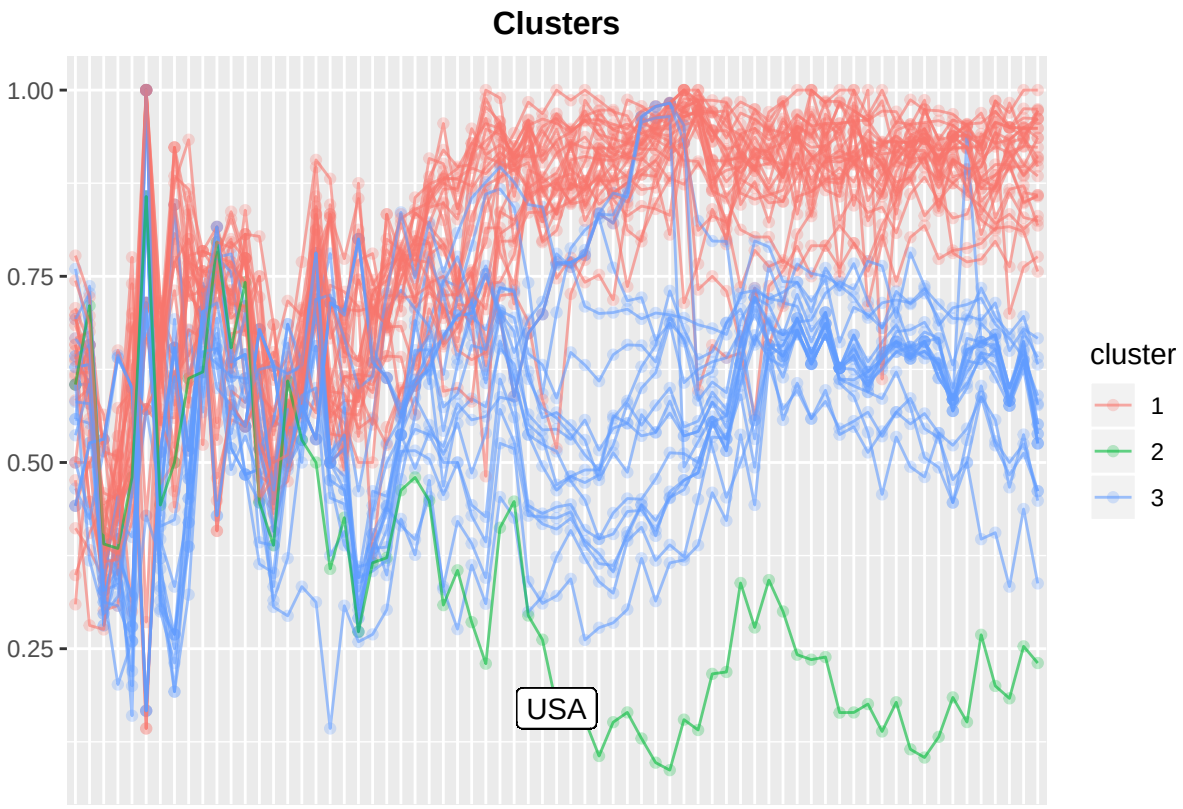
Initially a number of clusters `k` equal to 4 is consider.

```
distMatrix <- dist(Timeseries2, method= "DTW")
cls <- kmeans(x = distMatrix, centers = 4)
DataPlot$cluster <- rep(as.character(cls$cluster), 69)
```

```
DataPlot%>%  ggplot(aes(x = key, y = value)) +
  geom_point(aes(colour = cluster), alpha=0.25)  +
  geom_line(aes(group = Country, color=cluster), alpha=0.6)+
  theme(plot.title = element_text(hjust = 0.5), axis.ticks.x=element_blank(),
        axis.text.x=element_blank())+ xlab("")+ ylab("") +
  mynamestheme+ ggtitle("Clusters") + geom_label(label="USA", x=35, y=0.17)
```

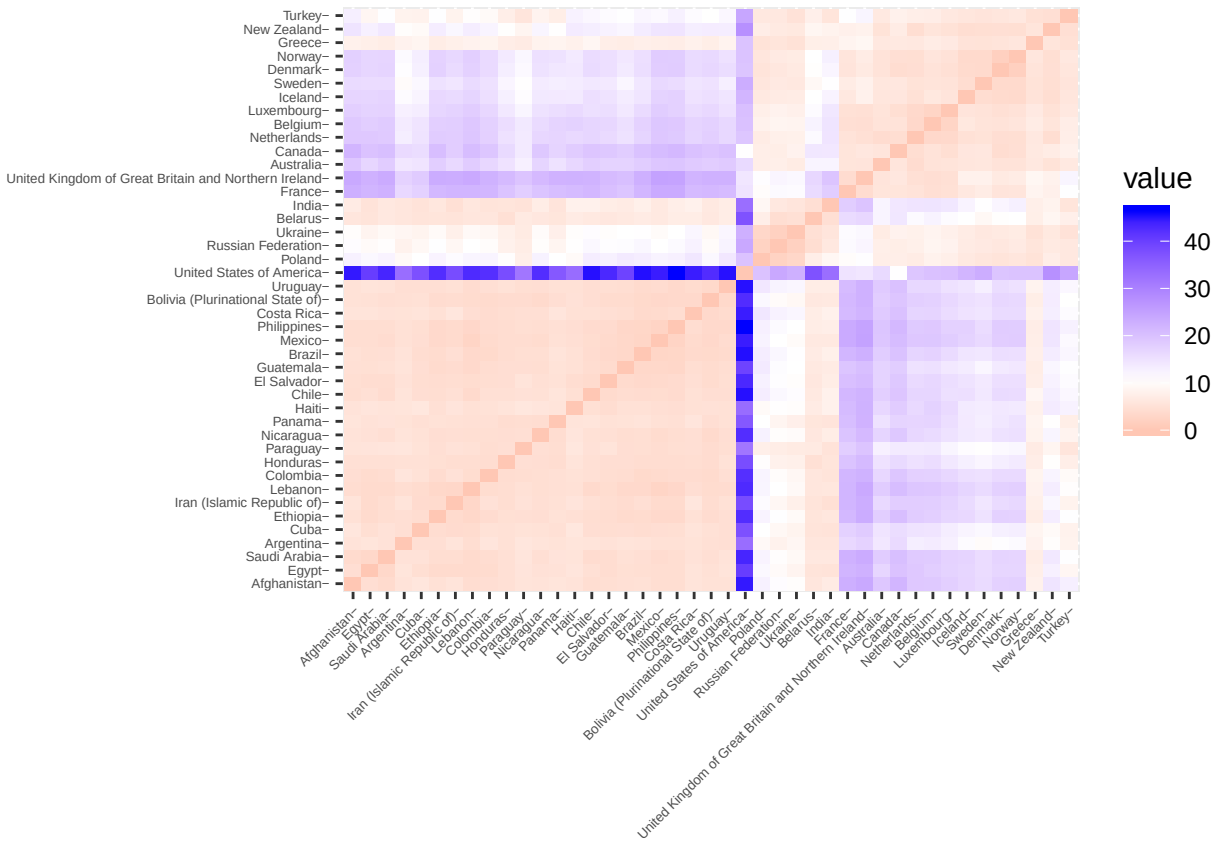


K-means with `k=4` return 4 clusters with dimension 1 (lightblue), 11 (purple), 6 (orange) and 25 (green). If instead the number of clusters was set to 3, the USA would still form a cluster (and therefore his outlier position is also identified in this case), but there would no longer be the possibility of differentiating the purple and orange series of the first plot. Below are the results with `k=3`



The heatmap corresponding to the distance matrix calculated with the DTW distance is displayed, in order to have a different view of what already emerged from the previous line graphs. The darker the color (bluish shades) the more distant are the two Countries. The dark blue color which corresponds to the USA shows how this Country completely stand out from the rest, as we had already noticed.

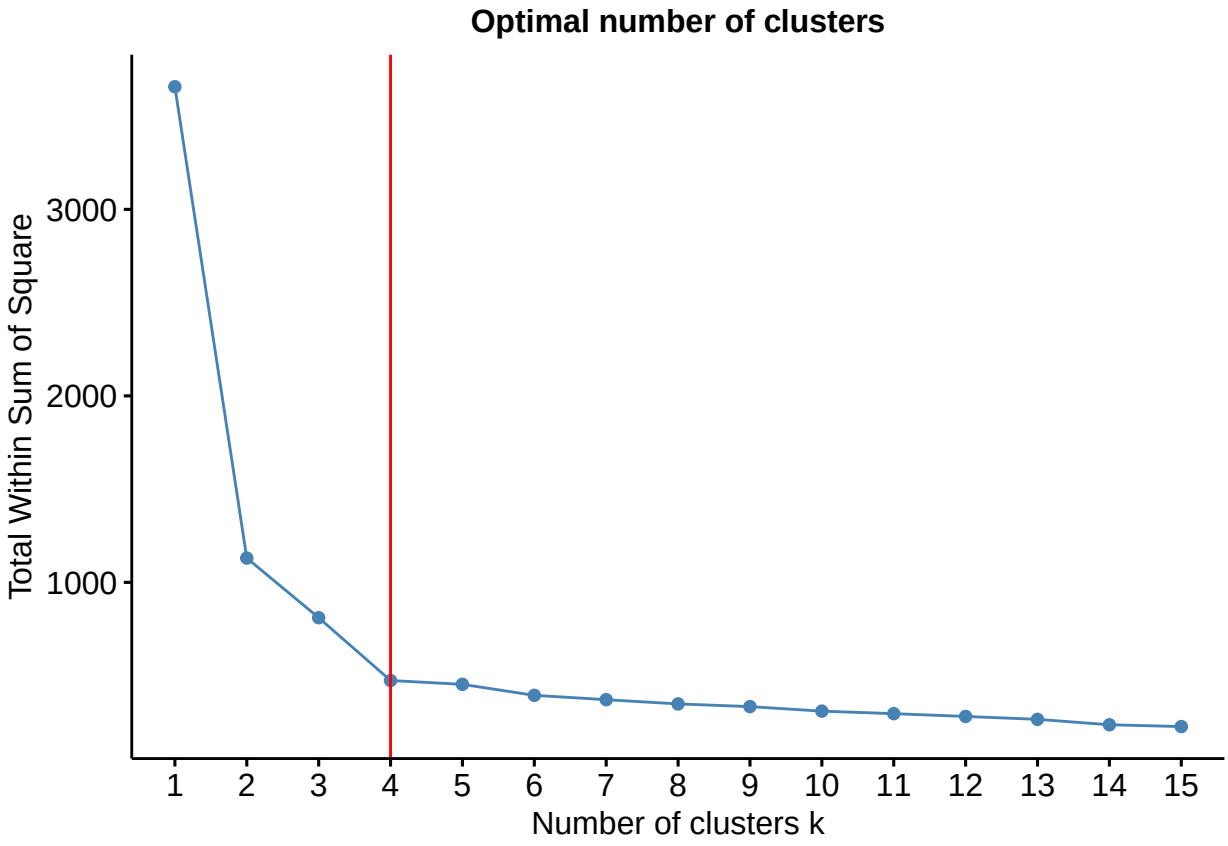
```
library(factoextra)

fviz_dist(distMatrix, lab_size = 5)
```



Now an hierarchical cluster analysis is performed, and then the results are compared with respect to the *kmeans* used previously. `hclust` is a `R` command which allows to perform hierarchical clustering. The arguments which have

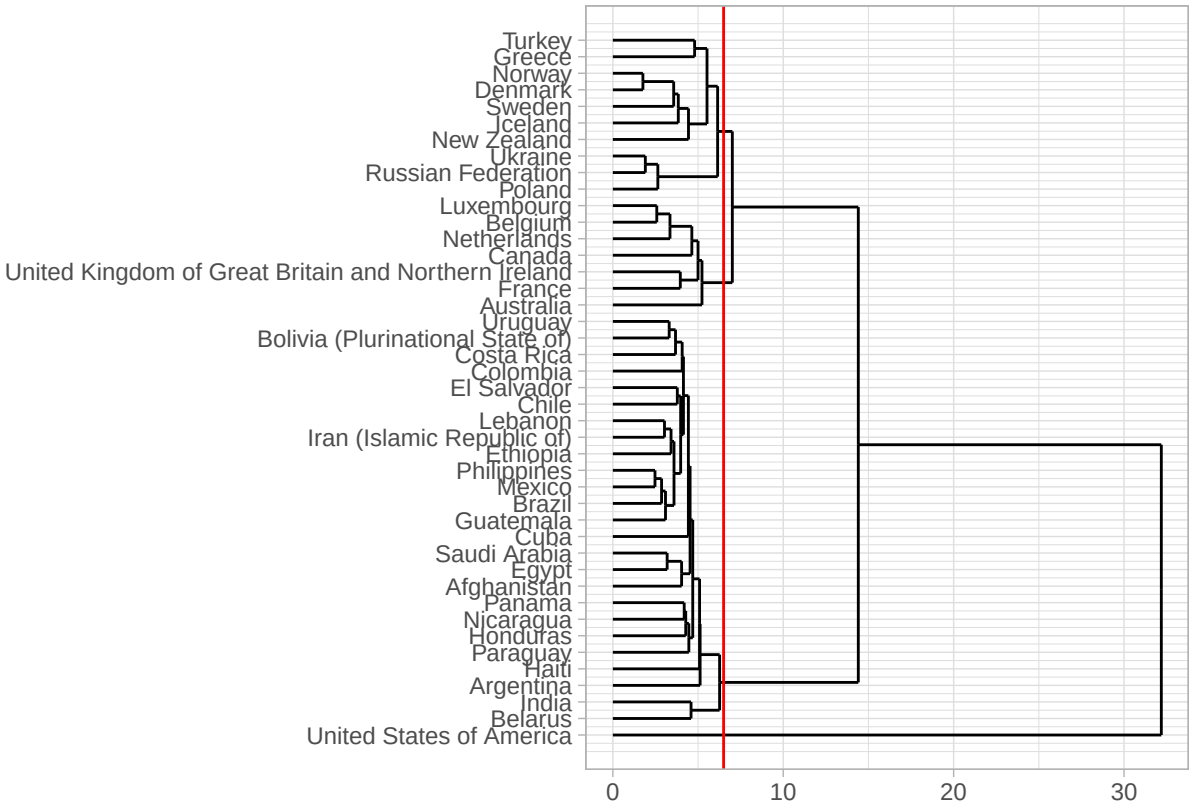
to be passed to the functions are `d`, which is a dissimilarity structure as produced by `dist` (in this case it is the distance Matrix produced using the DTW measure), and an agglomeration method (which distance has to be used to merge groups).
The first plot shows the optiman number of clusters by measuring the decrease in the total within sum of square variance: switching from `k=3` to `k=4` halves the total within sum of square variance, while increasing the number to `k=5` only causes a very small decrease in this measure, which makes `k=4` preferable.

```
hc <- hclust(distMatrix, method="average")

fviz_nbclust(x = Timeseries2, FUNcluster = kmeans, method = "wss", k.max = 15,
             diss = distMatrix, nstart = 50) + mynamestheme +
    theme(plot.title = element_text(hjust = 0.5))+ geom_vline(xintercept=4, col="red")
```



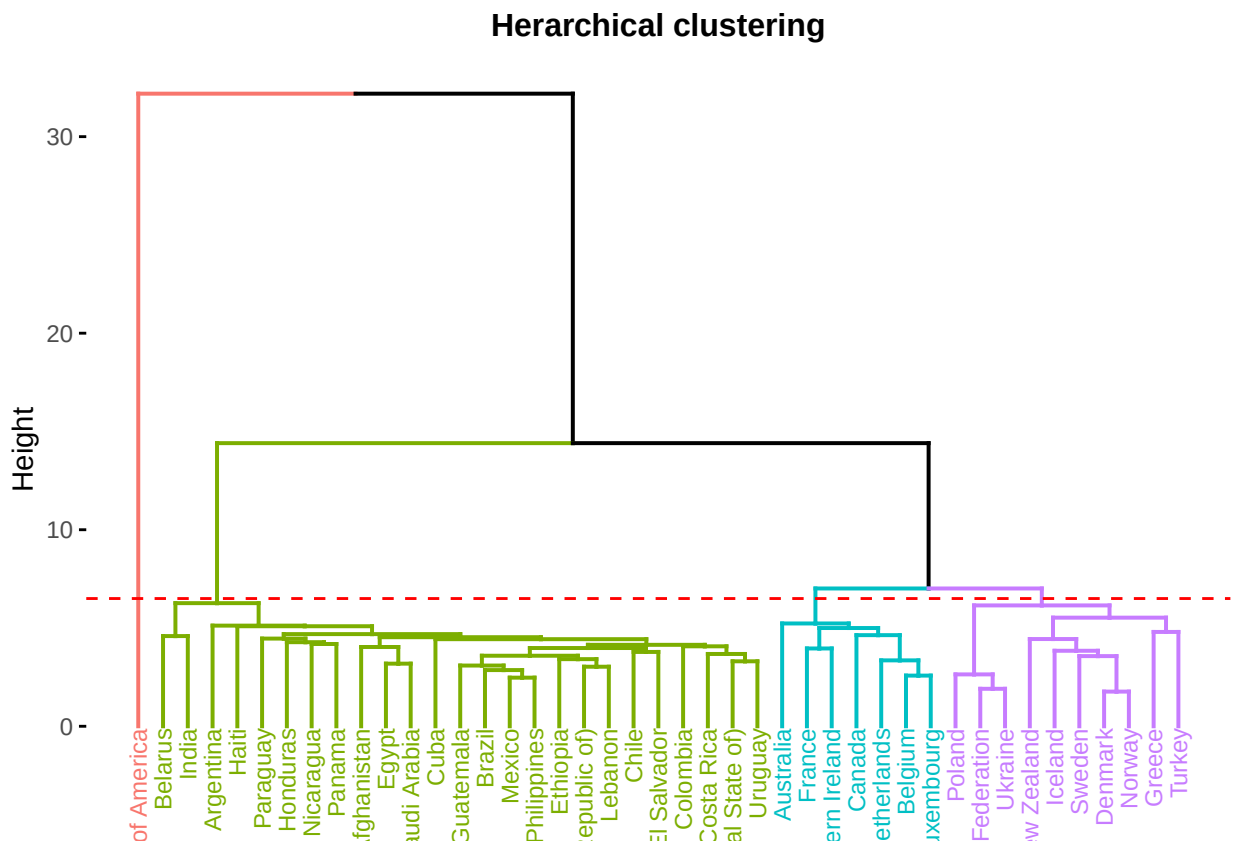
The following dendogram shows the distance at which fusion occurs: the red line has been added corresponding to a distance of 6.5. Stopping the fusion at this distance results in 4 clusters, one of which is the singleton USA (which is the last Country to be merged, at a much higher distance).

```
library(ggdendro)
ggdendrogram(hc, rotate = TRUE, theme_dendro = FALSE) + theme_light() + xlab("")+
  ylab("")+ geom_hline(yintercept=6.5, col="red")
```



The dendogram below is basically the same as the one above (the dashed red line has been added at the same level as before, that is 6.5), except that it shows the resulting 4 clusters by colouring the Countries accordingly.

```
fviz_dend(x = hc, k = 4, cex = 0.6) +
  geom_hline(yintercept = 6.5, linetype = "dashed", col="red") + mynamestheme+
  labs(title = "Herarchical clustering") + theme(plot.title = element_text(hjust = 0.5))
```



The clusters obtained using `k=4` are basically the same as the ones resulting from the *kmeans* algorithm: USA, a cluster (green) made up of 25 Countries, a cluster with 7 Countries (it was 6 before) and a last one with 10 Countries (instead of 10).

5 Cluster analysis based on the issue

The last task is to carry out a cluster analysis based on the *issue*. There are two ways to perform this analysis:

1. each issue can be considered as a record, whose variables are the yes-no-abstain vote for each years (that is, for each year the proportion of, say, yes votes is computed by averaging over all the Countries). By doing this, a total of just 6 units (i.e number of issues) will be considered, and for each one many variables will be available;
2. each Country is a record (as we have done so far) and the variables are the 6 issues: for each one, the proportion of yes votes will be considered as the value on which the clustering is based. The proportion is a global proportion for each Country averaged over the years (otherwise the problem would be again a time series clustering, as the alternative above).

The problem is quite similar in the two above mentioned approaches: the big difference is that in the first case the *issues* are clustered, while in the second one the clustering is performed on the Countries based on their behaviour concearning the issues.
Let's try to cluster `issues` as in 1.. Here we do not have any problem of NAs since the percentages are computed using only the available data, which means that some proportions are computed using a larger or smaller number of years, but since there is not any Country with completely missing values (no voting behaviour available for any year) we have a proportion for each Country and for each issue.
First, the data matrix is arranged in the way which is needed to perform the analysis, and the first 5 columns are displayed to show how the new data matrix looks like.

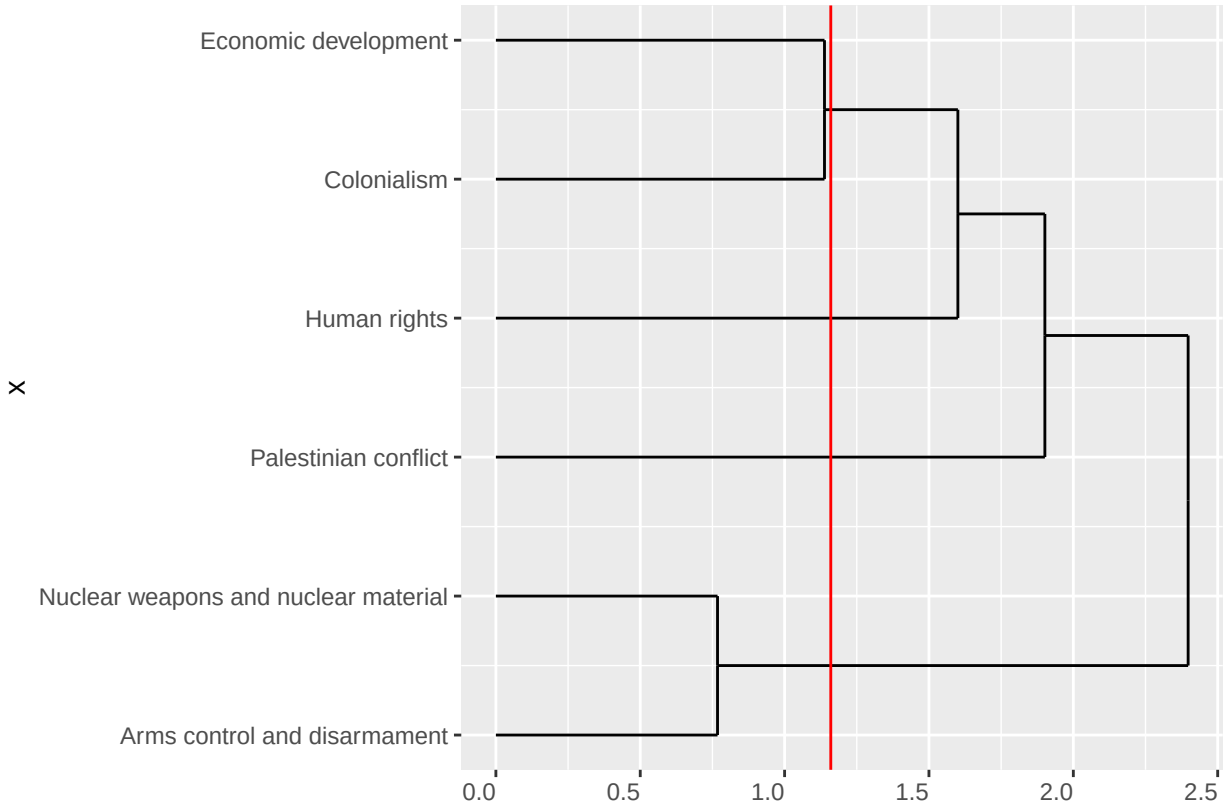
5.1 Clustering issues based on Countries' "Yes"-proportions

```
ClusterIssue<- votes_per2 %>% inner_join(un_roll_call_issues, by = "rcid") %>%
 group_by(issue, country) %>%
  summarize(percent_yes = mean(vote == "yes"))

ClusterIssue<- ClusterIssue%>% spread(key=country, value=percent_yes)
ClusterIssue<- ClusterIssue%>% remove_rownames()%>% column_to_rownames(var="issue")
ClusterIssue[, 1:5]
```

```
                                  Afghanistan   Albania   Algeria
Arms control and disarmament        0.8874025 0.6322581 0.8642384
Colonialism                         0.9086860 0.8410042 0.9660574
Economic development                0.9159091 0.7875817 0.9537037
Human rights                        0.8694690 0.6721088 0.8636364
Nuclear weapons and nuclear material  0.9127989 0.5750487 0.8940028
Palestinian conflict                0.9426559 0.8229989 0.9764475
                                    Andorra    Angola
Arms control and disarmament        0.5995423 0.9139466
Colonialism                         0.8032129 0.9788584
Economic development                0.7169811 0.9848485
Human rights                        0.5245536 0.8393113
Nuclear weapons and nuclear material 0.4726688 0.9257885
Palestinian conflict                0.7857143 0.9636364
```

Sunsequently, a 6x6 distance matrix (`DistIssue`) is obtained, and an hierarchical clustering with the euclidean distance is performed. By default, `hclust()` uses the `complete()` method to evaluate which clusters to join.

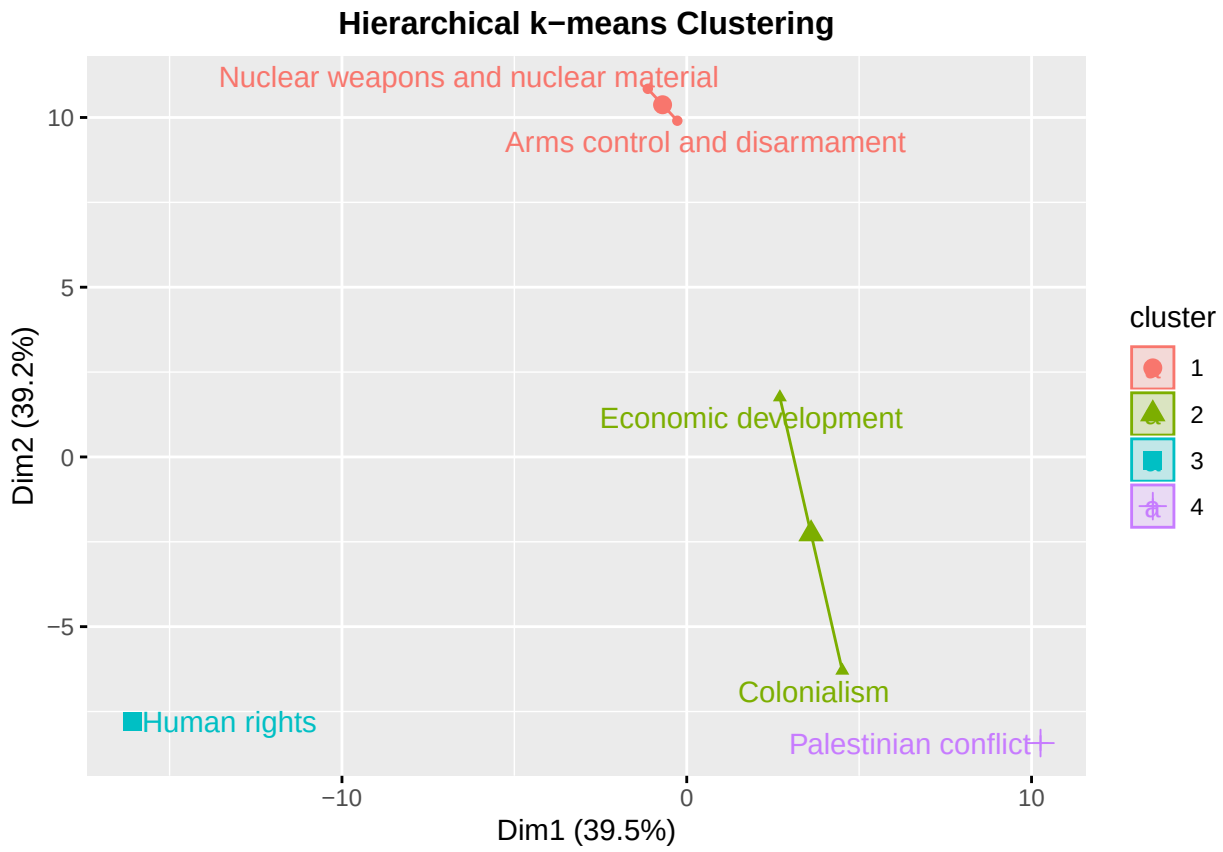
```
DistIssue <- dist(ClusterIssue, method= "euclidean")
hcIssue <- hclust(DistIssue)
ggdendrogram(hcIssue, rotate = TRUE, theme_dendro = FALSE) + ylab("")+
  geom_hline(yintercept=1.16, col="red")
```



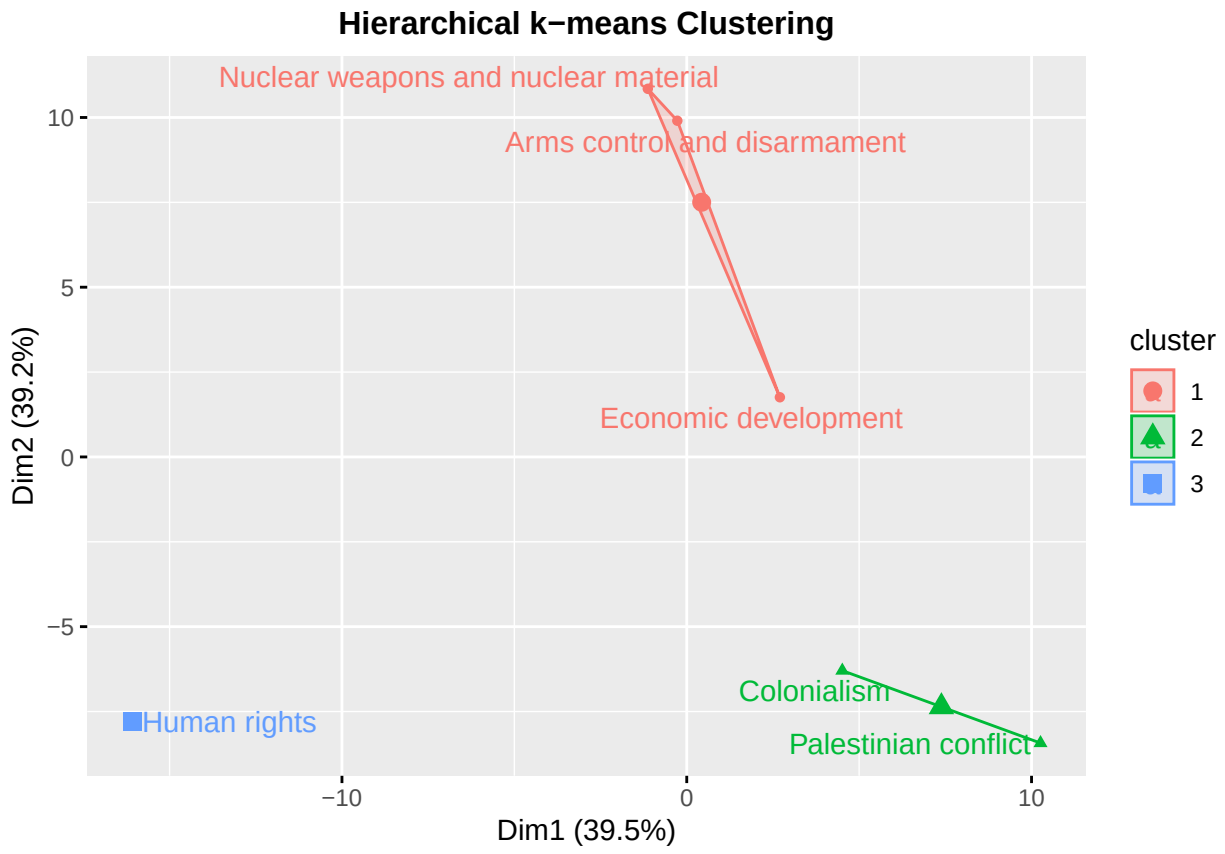
It can be seen that, stopping when the fusion distance is 1.16 (red vertical line), the resulting clusters are, in order of similarity, {Nuclear weapons and nuclear material, Arms control and disarmament} and {Economic development, Colonialism}, while the two singletons are {Human rights} and {Palestinian conflict}. The group {Nuclear weapons and nuclear material, Arms control and disarmament} is a stand-alone group, since it joins to the group made up of all the other (4) issues at a much higher level. The Palestinian conflict is the issue which is the more dissimilar to the others.

Finding 4 clusters with a non hierarchical method (again the kmeans) results is the following plot. The algorithm used is a variant of the k-means clustering solution, which is very sensitive to the initial random selection of cluster centers. The `hkmeans()` function provides a solution using an hybrid approach by combining the hierarchical clustering and the k-means methods. More to the point, the centers of the `k` clusters resulting from a hierarchical clustering are used as starting points for the *kmeans* algorithm. The issues are plotted against the first Principal Component (`x` axis) and the second Principal Component (`y` axis). It results in the same clusters as above.

```
clsIssue <- hkmeans(x = ClusterIssue, k = 4)
fviz_cluster(object = clsIssue, pallete = "jco", repel = TRUE) + mynamestheme+
 labs(title = "Hierarchical k-means Clustering") +
  theme(plot.title = element_text(hjust = 0.5))
```



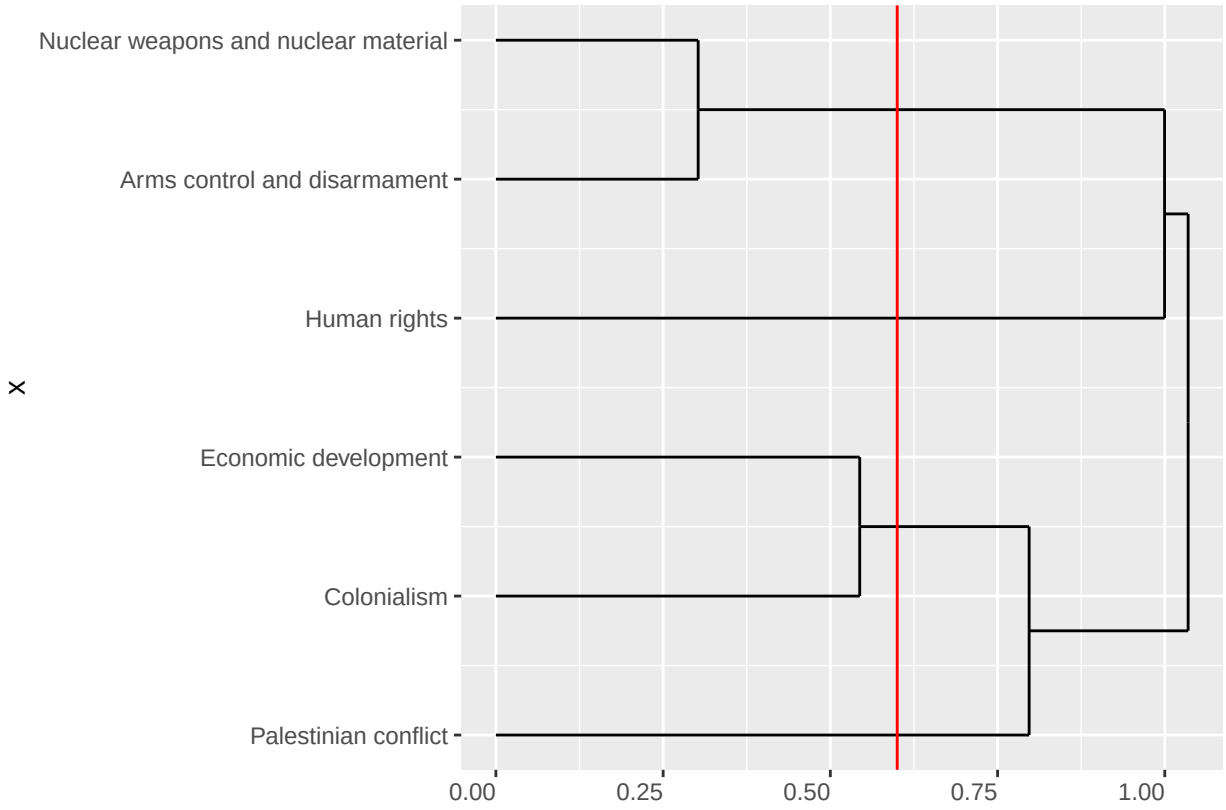
If the number of clusters is forced to be `k=3`, however, the result is a little bit different from what one could expect by looking at the previous dendogram. The resulting clusters are reported below. Note that what we get is more coherent with the clustering criterion used in the next section, where `Human rights` is the last issue to be joined and where Palestinian conflict and the group made up of Economic development and Colonialism are not so far apart (see **Clustering issues based on yearly "Yes"-proportions**).



5.2 Clustering issues based on yearly "Yes"-proportions

Alternatively, the grouping con be based on `year` and `issue`: the country voting behaviour is averaged for each year, and there is no distinction among Countries (below is a glimpse of the new dataframe). The variables are the years (years with NA for the proportion for some issue are removed).

```
                                     1948       1949      1957
Arms control and disarmament       0.1259080 0.10697674 0.5527638
Colonialism                        0.6516691 0.51272204 0.5682782
Economic development               0.5300000 0.25576923 0.8037975
Human rights                       0.3201970 0.60989011 0.6568627
Nuclear weapons and nuclear material 0.1235955 0.09411765 0.6317992
Palestinian conflict               0.3805601 0.55778894 0.7390511
                                     1961
Arms control and disarmament       0.5766667
Colonialism                        0.6045570
Economic development               0.5922747
Human rights                       0.7987680
Nuclear weapons and nuclear material 0.6140156
Palestinian conflict               0.4895561
```



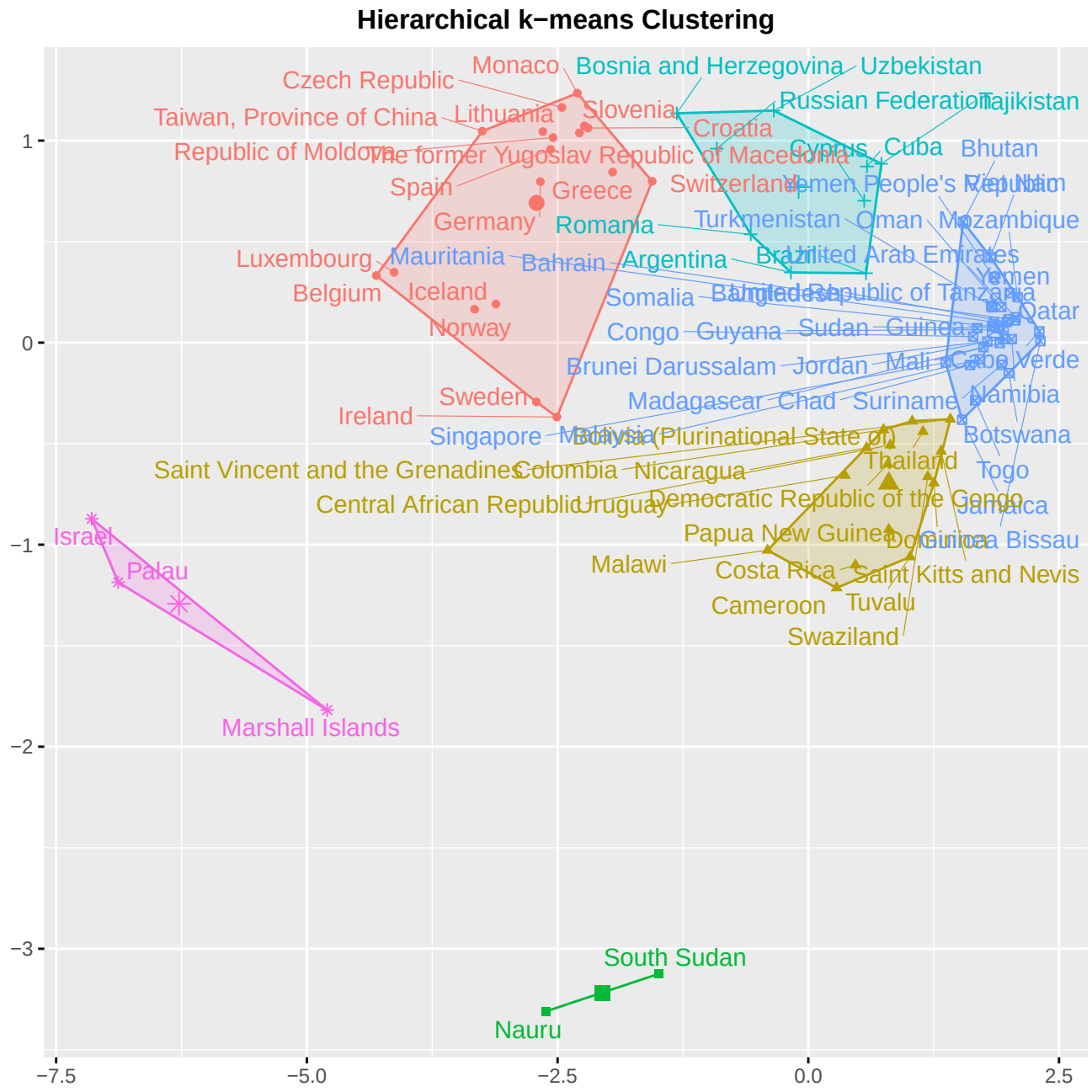
Also this second way of doing cluster analysis (i.e. using data in a very different form) gives result which are quite similar to the other way of clustering the issues. "Palestinian conflict" and "Human rights" are also in this case quite dissimilar to the other issues, while "Nuclear weapons and nuclear material" and "Arms control and disarmament" are again the closest, together with the couple "Colonialism" and "Economic development".

5.3 Clustering Countries based on the issue

As discussed at the beginning of the section **Cluster analysis based on the issue**, there is another way of carrying out the task "*perform a cluster analysis on the issue*" by clustering Countries according to their voting behaviour concerning the 6 issues. The new data matrix on which a distance matrix will be computed is the following:

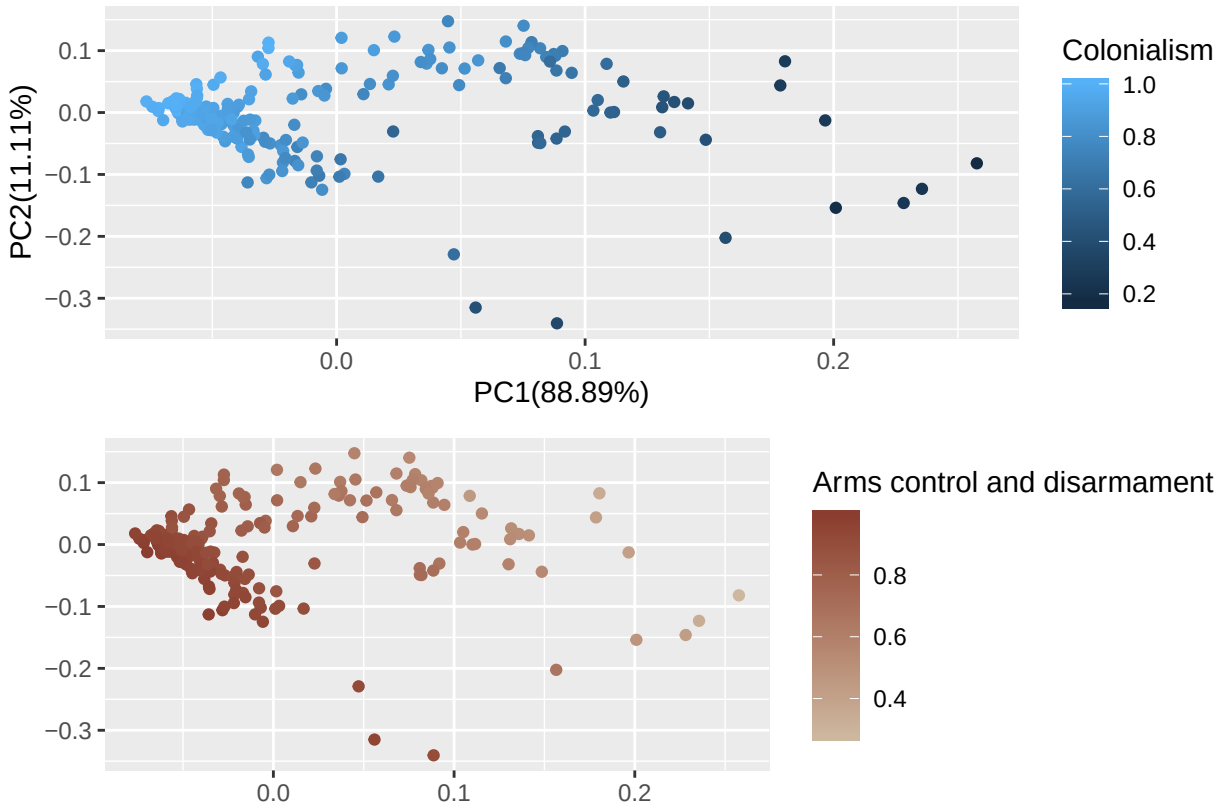
```
            Arms control and disarmament Colonialism Economic development
Afghanistan                    0.8874025   0.9086860            0.9159091
Albania                        0.6322581   0.8410042            0.7875817
Algeria                        0.8642384   0.9660574            0.9537037
Andorra                        0.5995423   0.8032129            0.7169811
Angola                         0.9139466   0.9788584            0.9848485
            Human rights
Afghanistan    0.8694690
Albania        0.6721088
Algeria        0.8636364
Andorra        0.5245536
Angola         0.8393113
```

Again, in this case an hierarchical k-means clustering will be performed. For graphic reasons, a subset of 80 countries is randomly selected to display clusters. A number of `k=6` clusters has been chosen, since it is the number of Macro-regions plus one, to allow for outliers to be detected in case there were any. The clusters identified do not exactly reflect the macro regions (although in each cluster there is a prevalence of a specific geographical area, for example cluster 1 gathers many Countries from Northern and East Europe). We also see that the first PC explains quite a good proportion of the total variability (84.8% of the total variance of the subsetted dataset).


Hierarchical k−means Clustering
Monaco
Bosnia and Herzegovina
Uzbekistan
Czech Republic
Russian Federation
Tajikistan
Taiwan, Province of China
Lithuania
Slovenia
Croatia
Republic of Moldova
The former Yugoslav Republic of Macedonia
Cyprus
Cuba
Bhutan
Spain
Greece
Switzerland
Yemen
Germany
Romania
Turkmenistan
Oman
Mozambique
Luxembourg
Mauritania
Bahrain
Argentina
Brazil
United Arab Emirates
Belgium
Iceland
Somalia
Bangladesh
United Republic of Tanzania
Norway
Qatar
Congo
Guyana
Sudan
Guinea
Brunei Darussalam
Jordan
Mali
Cabo Verde
Sweden
Madagascar
Chad
Suriname
Namibia
Ireland
Singapore
Botswana
Thailand
Saint Vincent and the Grenadines
Colombia
Nicaragua
Togo
Central African Republic
Uruguay
Democratic Republic of the Congo
Jamaica
Israel
Papua New Guinea
Dominica
Guinea Bissau
Palau
Malawi
Costa Rica
Saint Kitts and Nevis
Cameroon
Tuvalu
Swaziland
Marshall Islands
South Sudan
Nauru
1
0
−1
−2
−3
−7.5
−5.0
−2.5
0.0
2.5

6 Principal Component Analysis and visualization

To perform a principal component analysis (PCA) the last dataset will be used, where the rows are Countries and the variables are the 6 issues (and the value in the position (i,j) is the proportion of yes-votes for the ith-Country on the j-th issue).
A PCA is performed with the `prcomp()` command, and the eigenvalues and eigenvectors are obtained. Those are used to understand respectively the percentage of total variance explained by each component and the most influential variables in each component (by looking at the highest absolute value in the eigenvalues 1, 2, ..., which represents the vector of the weights of each variable). By running the command `eigen(cov(ClusterIssue2))$vectors[i,]` we can see the i-th autovector, while `eigen(cov(ClusterIssue2))$values` returns the list of eigenvalues. As for the first eigenvector, for example, the highest weights (absolute value) are associated to Colonialism and Palestinial conflict. In the following graphs the 200 countries are plotted against the first and second PCs. The color was given based on the voting proportion for different issues. It can be seen that Colonialism is relevant for the PC1 (since higher proportions are on the left and low proportion on the right, thus the PC1 helps discriminating Countries on this issue), while Arms control and disarmament is relevant for the PC2, since we can see bands of y values for which the colours of Countries have the same shade of brown.



A **biplot** is now represented to allow for an easier interpretation of the result of the PCA. Arrows for each variable point in the direction of increasing values of that variable. All the arrows point to the direction of increased values for PC1, which means that the higher the value of PC1, the higher the values for those variables. For graphic reasons Countries are represented as points instead of as text labels, which does not allow to understand the relation of a Country with the variables.

```
p<-princomp(d)

fviz_pca_biplot(p, geom=c("point"))+ theme_gray()+ mynamestheme+
  theme(plot.title = element_text(hjust = 0.5))+ xlab("")+ ylab("")
```

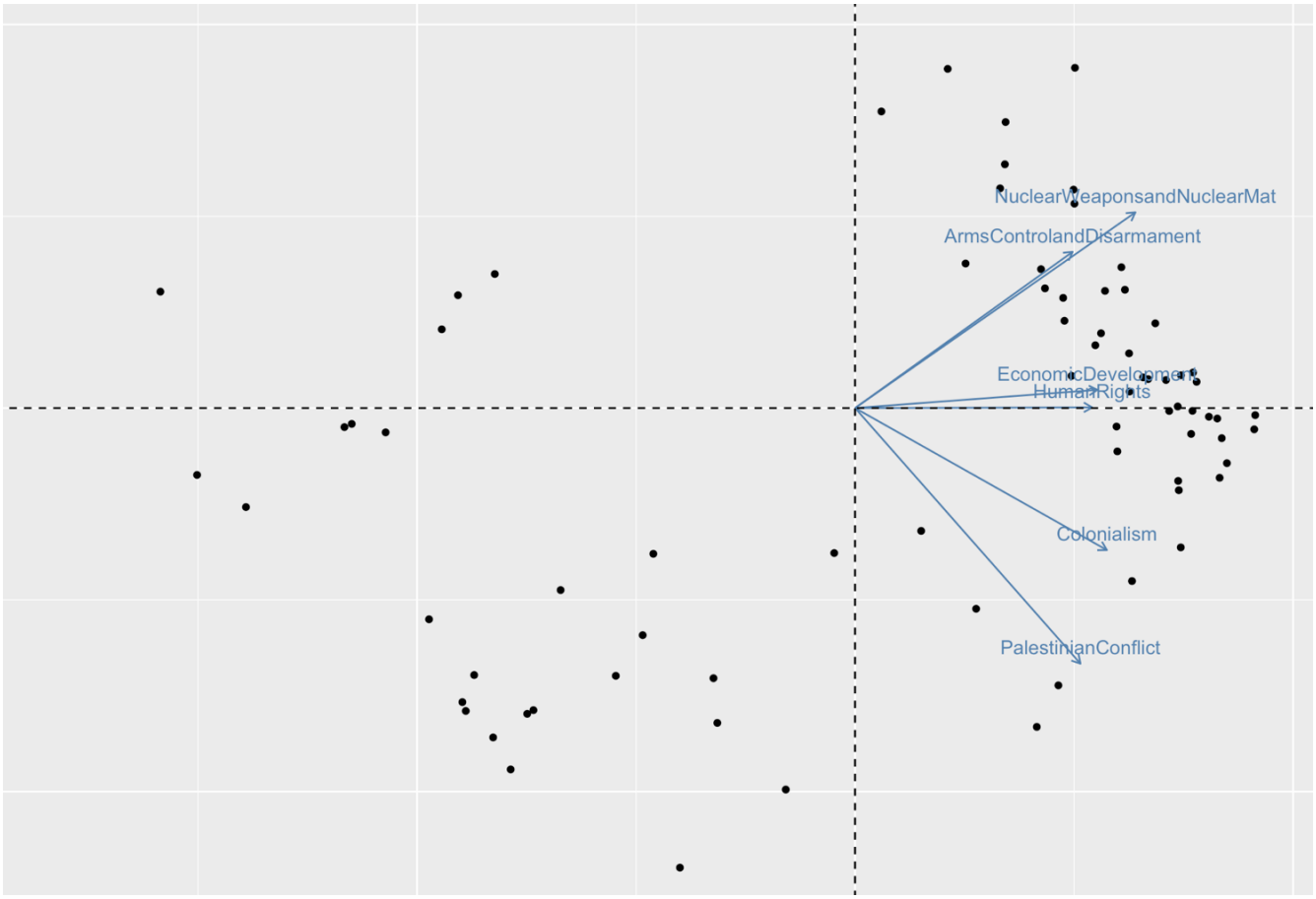


Figure 1: PCA and biplot