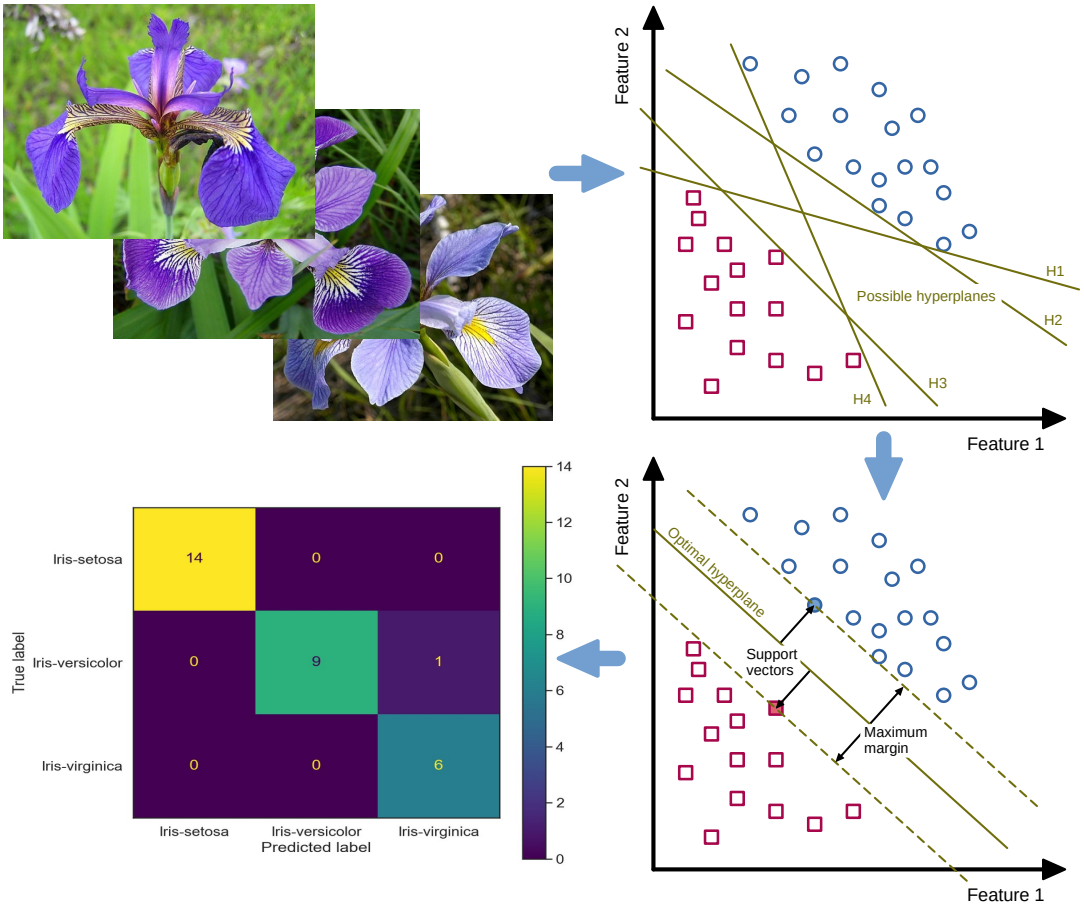
Application of the processed survey data in the analytical hierarchy process (AHP)

Björn Kasper (kasper.bjoern@bgetem.de)[1] and Henriette John (h.john@ioer.de)[2]

[1] *Berufsgenossenschaft Energie Textil Elektro Medienerzeugnisse*
[2] *Leibniz Institute of Ecological Urban and Regional Development*

December 8, 2022; version 0.1 (pre-release)



This is a placeholder for the abstract that needs to be added later.

Contents

1 Introduction

Why we use a Jupyter notebook to to publish the R program examples:

Jupyter is a new **open source** alternative to the proprietary numerical software Mathematica from **Wolfram Research** that is well on the way to become a **standard for exchanging research results** (Somers 2018; Romer 2018).

Originally Jupyter was intended as an IDE for the programming languages **Julia** and **Python**. Besides that it is also possible to install other interpreter kernels, such as the **IRkernel** for R. This can be interesting if the IDE **RStudio Desktop** is not available on the target platform used. For example, it is very difficult to install RStudio on the ARM-based embedded computer **Raspberry Pi** due to many technical dependencies. In contrast, using the R kernel in JupyterLab on the Raspberry Pi works very well and performant.

2 Global settings and dependencies

2.1 Install missing packages if not present yet

Attention: For some R packages several dependencies have to be installed first with `apt install <package name>`.

Dependencies for package `ahpsurvey`:
- R package `randomNames` (it depends on R 4.0, refer to https://cran.r-project.org/web/packages/randomNames/index.html)

Drawback for **Raspbian buster**: the dependency `randomNames` is not available for R v3.5.2 as it depends on R (4.0). Upgrading R in Raspbian following the instruction on https://cran.rstudio.com/bin/linux/debian/#debian-buster-stable does not work so far ...

[1]:
```
# List of R packages that are used in this script
list.of.packages <- c("data.table", "ggplot2", "tidyr", "dplyr", "magrittr",␣
 ↪"ahpsurvey")

# Query the already installed packages and save the missing ones in a new list
missing.packages <- list.of.packages[!(list.of.packages %in% installed.
 ↪packages()[,"Package"])]

# Install missing packages
if(length(missing.packages)) {
    install.packages(missing.packages)
} else {
    print("All required packages are installed.")
}
```

```
[1] "All required packages are installed."
```

2.2 Load package `data.table`

The package `data.table` is used to read and manipulate tables (`data.table` inherits from `data.frame`). Install and load it:

[2]:
```
library(data.table)
```

2.3 Load package `ggplot2`

The package `ggplot2` is used to plot diagrams. Install and load it:

[3]:
```
library(ggplot2)
```

2.4 Load package `tidyr`

The package `tidyr` is used to **reshape** the dataframes and provides functions like `gather()` or `spread()`. Some examples for the application can be found here: Reshaping Your Data with tidyrReshaping Your Data with tidyr.

Install and load it:

[4]:
```
library(tidyr)
```

2.5 Load package `dplyr`

The package `dplyr` is necessary to manipulate dataframes using functions like `select()`, `mutate()` and `left_join()`. Install and load it:

Hint: Setting the parameter `warn.conflicts=FALSE` when calling the `library()` function silences annoying messages about masked functions.

[5]:
```
library(dplyr, warn.conflicts=FALSE)
```

2.6 Use pipes for better coding

HINT: The pipe functionality is already available by loading the library `tidyr` - so you don't have to load it explicitly.

What pipes like `%>%` are and how to use them is described here: https://statistik-dresden.de/archives/15679.

Before using pipes in R, you have to install and load the package `magrittr`:

[6]:
```
library(magrittr, warn.conflicts=FALSE)
```

2.7 Load package `ahpsurvey`

The package `ahpsurvey` contains all the necessary mathematical and statistical methods to run the analytical hierarchy process (AHP).

[7]:
```
library(ahpsurvey)
```

3 Functions for processing AHP

3.1 Set globally used input and output folders

[8]:
```
str_input_path = "./output_data_manipulated"
str_output_path = "./output_data_AHP"
```

3.2 Function to read in the processed survey data from CSV files to dataframes

Define a function for reading in a CSV file to a date frame.

[9]:
```
func_readCSVdata_to_dataframe <- function(str_CSVfilename) {

  df_CSVdata <- fread(
    file = str_CSVfilename, encoding = "UTF-8",
    header = TRUE, sep = "\t", quote = "\""
    )

  return(df_CSVdata)
}
```

3.3 Function for generating a data frame with *eigentrue values* (weights)

[10]:
```
func_genEigentrue_to_dataframe <- function(df_surveyData, vec_attributes) {
  list_mat_judgement <- df_surveyData %>%
    ahp.mat(vec_attributes, negconvert = TRUE)

  df_eigentrue <- ahp.indpref(list_mat_judgement, vec_attributes, method = "eigen")

  return(df_eigentrue)
}
```

3.4 Function for generating an array with consistency ratios

[11]:
```
func_genCR_to_arr <- function(df_surveyData, vec_attributes) {
  arr_cr <- df_surveyData %>%
    ahp.mat(vec_attributes, negconvert = TRUE) %>%
    ahp.cr(vec_attributes, ri=0.58)

  return(arr_cr)
}
```

3.5 Function for generating a data frame with consistency ratios

[12]:
```
func_genCR_to_dataframe <- function(df_surveyData, vec_attributes, arr_cr,␣
↪consistency_thres=0.1, str_CRlabel) {
  df_cr <- df_surveyData %>%
    ahp.mat(vec_attributes, negconvert = TRUE) %>%
    ahp.cr(vec_attributes, ri=0.58) %>%
    data.frame() %>%
    mutate(rowid = 1:length(arr_cr), arr_cr.dum = as.factor(ifelse(arr_cr <=␣
↪consistency_thres, 1, 0)))

  # rename column with consistency ratios
  colnames(df_cr)[1] <- str_CRlabel

  return(df_cr)
}
```

3.6 Function for visualizing individual priorities and consistency ratios

[13]:
```
func_visuPriosCRs <- function(df_surveyData, df_cr, arr_cr, consistency_thres=0.1,␣
↪vec_attributes, df_eigentrue, vec_labels, str_image_filename, str_title) {
  df_cr_sel <- df_cr %>%
    select(arr_cr.dum, rowid)

  plot <- df_surveyData %>%
      ahp.mat(atts = vec_attributes, negconvert = TRUE) %>%
      ahp.indpref(vec_attributes, method = "eigen") %>%
      mutate(rowid = 1:nrow(df_eigentrue)) %>%
      left_join(df_cr_sel, by = 'rowid') %>%
      gather(all_of(vec_attributes), key = "var", value = "pref") %>%
      ggplot(aes(x = var, y = pref)) +
      geom_violin(alpha = 0.6, width = 0.8, color = "transparent", fill = "gray") +
      geom_jitter(alpha = 0.6, height = 0, width = 0.1, aes(color = arr_cr.dum)) +
```

```
      geom_boxplot(alpha = 0, width = 0.3, color = "#808080") +
      scale_x_discrete("Attribute", label = vec_labels) +
      scale_y_continuous("Weight (dominant eigenvalue)",
                         labels = scales::percent,
                         breaks = c(seq(0,0.7,0.1))) +
      guides(color=guide_legend(title=NULL))+
      scale_color_discrete(breaks = c(0,1),
                           labels = c(paste("CR >", consistency_thres),
                                      paste("CR <", consistency_thres))) +
      labs(NULL, caption = paste("n =", nrow(df_surveyData), ",", "Mean CR =",
                                 round(mean(arr_cr),3))) +
      theme_minimal() +
      ggtitle(str_title)

  # save generated ggplot graphic to a PNG image file
  ggsave(filename = str_image_filename, width = 7, height = 7, dpi = 300)

  print(plot)
}
```

3.7 Function for generating geometric mean values from individual judgement matrices

[14]:
```
func_aggpref_gmean <- function(df_surveyData, vec_attributes, arr_cr,␣
↪consistency_thres=0.1, str_CRlabel) {
  df_cr <- df_surveyData %>%
    ahp.mat(vec_attributes, negconvert = TRUE) %>%
    ahp.cr(vec_attributes, ri=0.58) %>%
    data.frame() %>%
    mutate(rowid = 1:length(arr_cr), arr_cr.dum = as.factor(ifelse(arr_cr <=␣
↪consistency_thres, 1, 0)))

  # rename column with consistency ratios
  colnames(df_cr)[1] <- str_CRlabel

  # combine data frame 'df_cr' with raw survey data ('df_surveyData')
  df_cr_wRaw <- cbind(df_cr, df_surveyData)

  # remove rows, where 'arr_cr.dum' == 0 (inconsistent data)
  df_cr_wRaw_cons <- df_cr_wRaw[df_cr_wRaw$arr_cr.dum != 0, ]

  # get individual judgement matrices from last 3 columns
  list_mat_judgement <- df_cr_wRaw_cons[tail(names(df_cr_wRaw_cons), 3)] %>%
    ahp.mat(vec_atts, negconvert = TRUE)

  # get geometric mean values from judgement matrices
  list_gmean_l <- ahp.aggpref(list_mat_judgement, vec_atts, method = "eigen",␣
↪aggmethod = "geometric")

  return(list_gmean_l)
}
```

3.8 Function for normalizing the geometric mean values

[15]:
```
func_norm_gmean <- function(list_gmeans) {
  # normalization so that the sum of the geometric mean values is 1 (corresponds to␣
 ↪100%)
  df_gmean_l <- data.frame(list_gmeans)
  # rename column with geometric mean values (raw)
  colnames(df_gmean_l)[1] <- "gmean.raw"

  gmean_sum <- 0
  for ( val in list_gmeans ) {
    gmean_sum <- gmean_sum + val
  }
  df_gmean_l["Sum", 1] <- gmean_sum

  for (idx in 1:length(list_gmeans)) {
    gmean_norm <- list_gmeans[[idx]] / gmean_sum
    df_gmean_l[idx, "gmean.norm"] <- gmean_norm
  }

  gmean_sum_norm <- 0
  # iterate over all rows except the last, because this is the sum itself
  for ( row in 1:(nrow(df_gmean_l)-1) ) {
    gmean_sum_norm <- gmean_sum_norm + df_gmean_l[row, 2]
  }
  df_gmean_l["Sum", 2] <- gmean_sum_norm

  return(df_gmean_l)
}
```

3.9 Function to process the ahpsurvey and create violin plots

[31]:
```
func_process_ahpsurvey <- function(df_csvInputFiles, str_participants_group,␣
 ↪str_input_path, str_output_path, df_attributes_labels) {
    row_start = 1
    row_end = 3

    df_outputTable <- data.table()

    for ( file_idx in 1:nrow(df_csvInputFiles) ) {
        # create dataframe from current input CSV file
        str_filename <- paste(str_input_path, df_csvInputFiles[file_idx,␣
 ↪filenames], sep="/")
        df_processed_survey_data <- func_readCSVdata_to_dataframe(str_filename)

        # create vectors for attributes and labels from a subset of data frame␣
 ↪'df_attributes_labels_XXX'
        vec_atts <- df_attributes_labels[c(row_start:row_end), attr]
        vec_labels <- df_attributes_labels[c(row_start:row_end), labels]

        # shift row interval for next iteration
        row_start = row_start + 3
        row_end = row_end + 3

        # create title string for violin plots
```

```
        str_violin_title = "Violins display priorities and consistency ratios for␣
↪attributes"
        str_violin_title <- paste(str_violin_title, vec_atts[1], sep = " ")
        # extend title string by looping through attributes, starting with 2nd␣
↪element
        for ( idx in 2:length(vec_atts) ) {
            str_violin_title <- paste(str_violin_title, vec_atts[idx], sep = ", ")
        }
        # print(str_violin_title)

        # generate data frame with eigentrue values (weights)
        df_eigentrue_weights <-␣
↪func_genEigentrue_to_dataframe(df_processed_survey_data, vec_atts)

        # generate an array with consistency ratios
        arr_CRs <- func_genCR_to_arr(df_processed_survey_data, vec_atts)

        # generate an extended data frame with consistency ratios
        consistency_thres = 0.1
        str_CRlabel <- paste("CR", df_csvInputFiles[file_idx, keys], sep="_")
        df_CRs <- func_genCR_to_dataframe(df_processed_survey_data, vec_atts,␣
↪arr_CRs, consistency_thres, str_CRlabel)

        str_image_filename <- paste("ahp_violin", str_participants_group,␣
↪df_csvInputFiles[file_idx, keys], sep="_")
        str_image_filename <- paste(str_image_filename, ".png", sep="")
        str_image_filename <- paste(str_output_path, str_image_filename, sep="/")
        func_visuPriosCRs(df_processed_survey_data, df_CRs, arr_CRs,␣
↪consistency_thres, vec_atts, df_eigentrue_weights, vec_labels,␣
↪str_image_filename, str_violin_title)

        # combine data frames of eigentrue values (weights) with consistency ratios
        df_outputTable <- cbind(df_outputTable, df_eigentrue_weights)
        # add only specific columns of 'df_CRs' (omit column 'row_id')
        df_outputTable <- cbind(df_outputTable, df_CRs[c(1, 3)])
    }

    # extend file name by path
    str_CSVfilename_output <- paste("rdata", str_participants_group,␣
↪"eigentrue_CRs", sep="_")
    str_CSVfilename_output <- paste(str_CSVfilename_output, ".csv", sep="")
    str_CSVfilename_output <- paste(str_output_path, str_CSVfilename_output, sep="/
↪")

    # write data frame 'df_outputTable' to CSV file for further statistical analysis
    write.table(df_outputTable, file = str_CSVfilename_output,
                fileEncoding = "UTF-8", row.names = FALSE,
                col.names = TRUE, sep = "\t", quote = TRUE)
}
```

4 Create data frames (tables) handling the file names of processed survey data

4.1 File table for all participants

[60]:
```
df_csvInputFiles_all <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_all_crit_AHP_edible_Cities_2022-03-18_09-53.csv",
                "rdata_all_env_AHP_edible_Cities_2022-03-18_09-53.csv",
                "rdata_all_soc_AHP_edible_Cities_2022-03-18_09-53.csv",
                "rdata_all_eco_AHP_edible_Cities_2022-03-18_09-53.csv"),
  descriptions = c("criteria (main criteria)",
                   "environmental sub-criteria",
                   "social sub-criteria",
                   "economic sub-criteria")
)
```

4.2 File table for city administrations

[61]:
```
df_csvInputFiles_CA <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_CA_crit_AHP_edible_Cities_2022-03-18_10-28.csv",
                "rdata_CA_env_AHP_edible_Cities_2022-03-18_10-28.csv",
                "rdata_CA_soc_AHP_edible_Cities_2022-03-18_10-28.csv",
                "rdata_CA_eco_AHP_edible_Cities_2022-03-18_10-28.csv"),
  descriptions = c("criteria (main criteria)",
                   "environmental sub-criteria",
                   "social sub-criteria",
                   "economic sub-criteria")
)
```

4.3 File table for non-governmental organizations

[62]:
```
df_csvInputFiles_NGO <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_NGO_crit_AHP_edible_Cities_2022-03-18_10-40.csv",
                "rdata_NGO_env_AHP_edible_Cities_2022-03-18_10-40.csv",
                "rdata_NGO_soc_AHP_edible_Cities_2022-03-18_10-40.csv",
                "rdata_NGO_eco_AHP_edible_Cities_2022-03-18_10-40.csv"),
  descriptions = c("criteria (main criteria)",
                   "environmental sub-criteria",
                   "social sub-criteria",
                   "economic sub-criteria")
)
```

4.4 File table for practitioners and experts

[63]:
```
df_csvInputFiles_PE <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_PE_crit_AHP_edible_Cities_2022-03-18_10-41.csv",
```

```
                "rdata_PE_env_AHP_edible_Cities_2022-03-18_10-41.csv",
                "rdata_PE_soc_AHP_edible_Cities_2022-03-18_10-41.csv",
                "rdata_PE_eco_AHP_edible_Cities_2022-03-18_10-41.csv"),
  descriptions = c("criteria (main criteria)",
                   "environmental sub-criteria",
                   "social sub-criteria",
                   "economic sub-criteria")
)
```

5 Exploit datasets of own survey with package ahpsurvey for each group of participants

[109]:
```
df_attributes_labels <- data.table(
  # idx = 1:12,
  attr = c("Envi", "Soci", "Econ",
           "Clim", "BDiv", "CiEc",
           "KEdu", "Comm", "Part",
           "Qual", "LVCs", "Affo"),
  labels = c("Environment", "Society", "Economy",
             "Microclimate and Hydrology", "Biodiversity", "Circular economy",
             "Knowledge and Education", "Community Building", "Participation",
             "Food Quality and Safety", "Local Value Chains", "Food Affordability")
)

print.data.frame(df_attributes_labels, right=FALSE)
```

```
   attr labels
1  Envi Environment
2  Soci Society
3  Econ Economy
4  Clim Microclimate and Hydrology
5  BDiv Biodiversity
6  CiEc Circular economy
7  KEdu Knowledge and Education
8  Comm Community Building
9  Part Participation
10 Qual Food Quality and Safety
11 LVCs Local Value Chains
12 Affo Food Affordability
```

5.1 All participants

[72]:
```
func_process_ahpsurvey(df_csvInputFiles_all, "all", str_input_path,␣
↪str_output_path, df_attributes_labels)
```

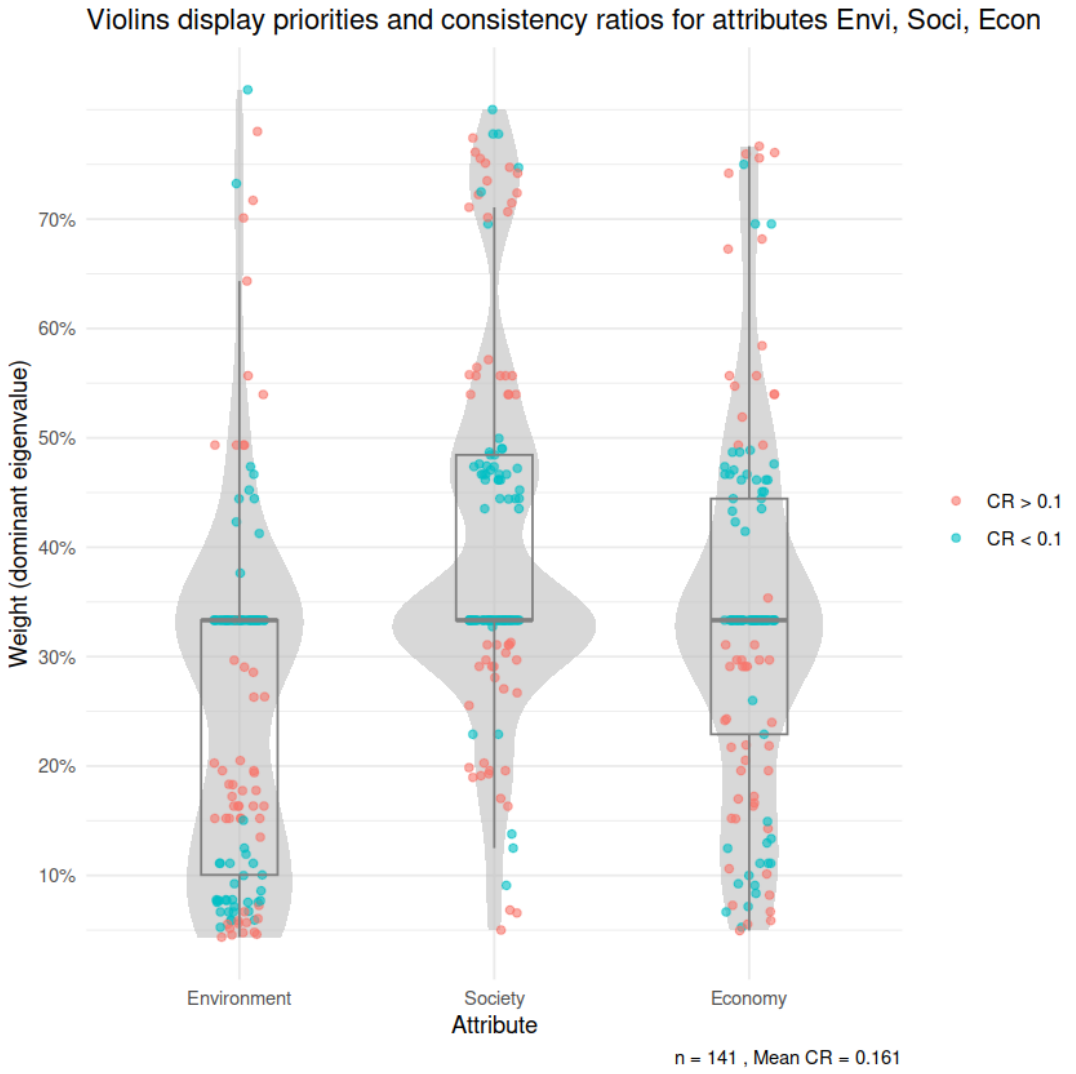


Figure 1: Series of violin plots that display the priorities and consistency ratios of all participants

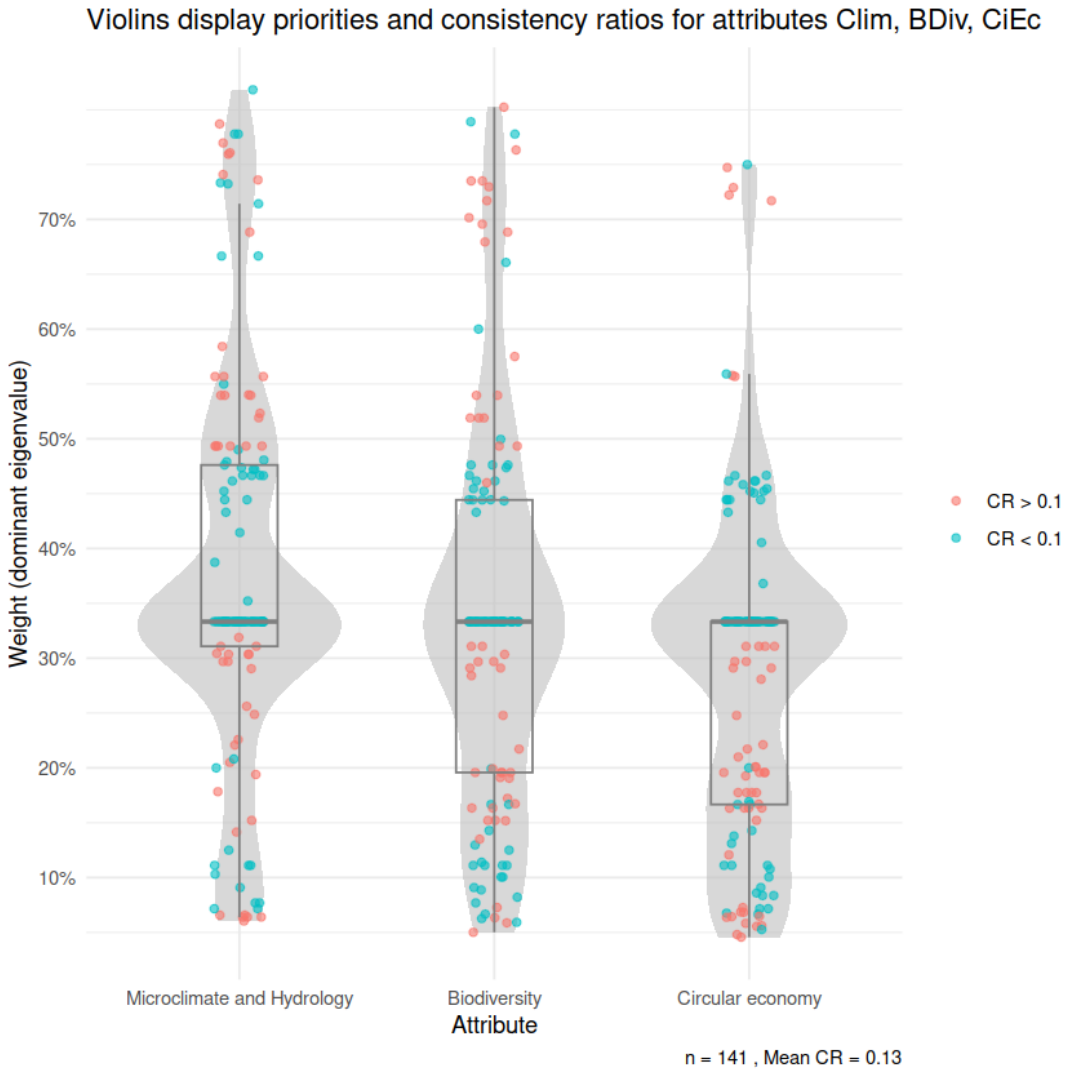


Figure 2: Series of violin plots that display the priorities and consistency ratios of all participants

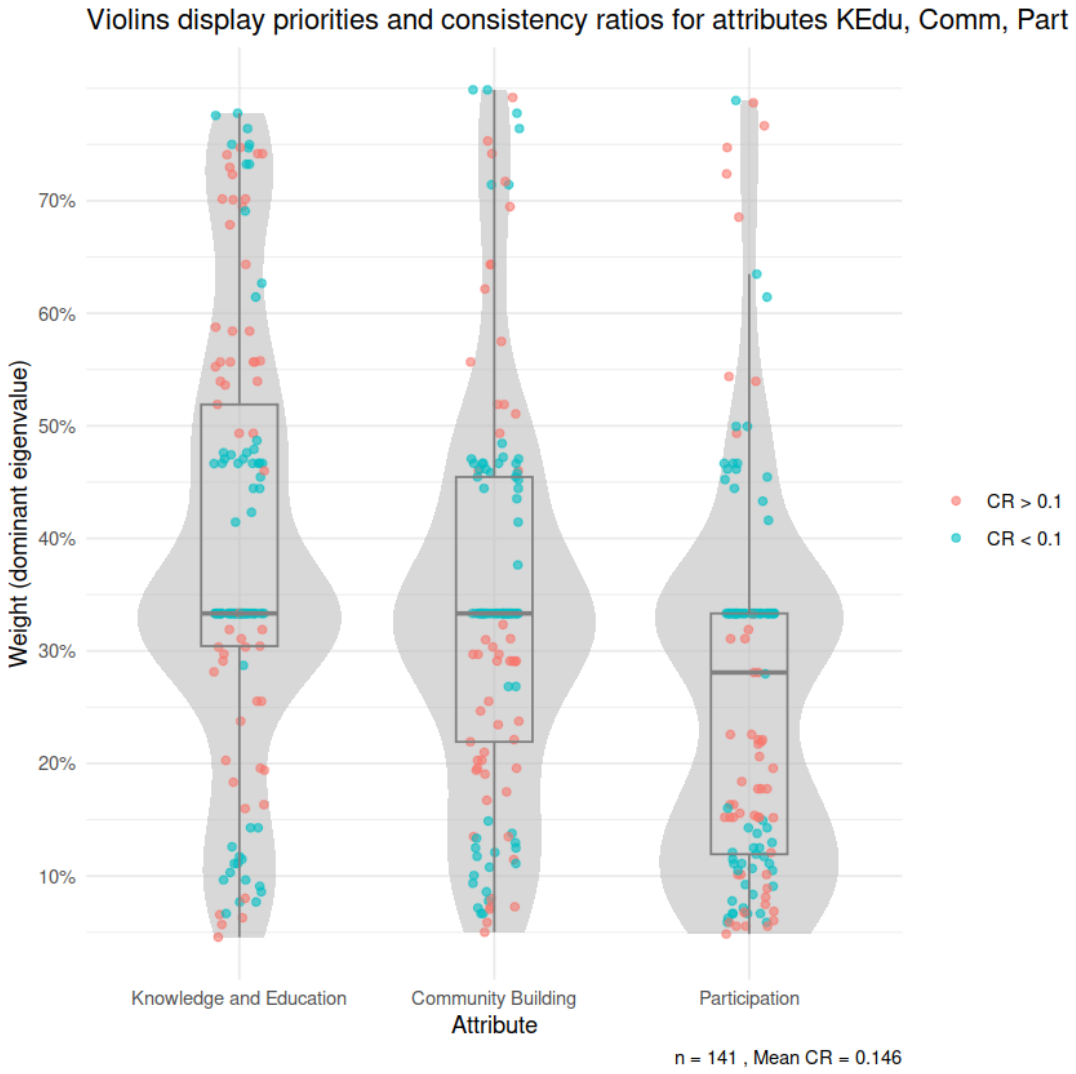


Figure 3: Series of violin plots that display the priorities and consistency ratios of all participants

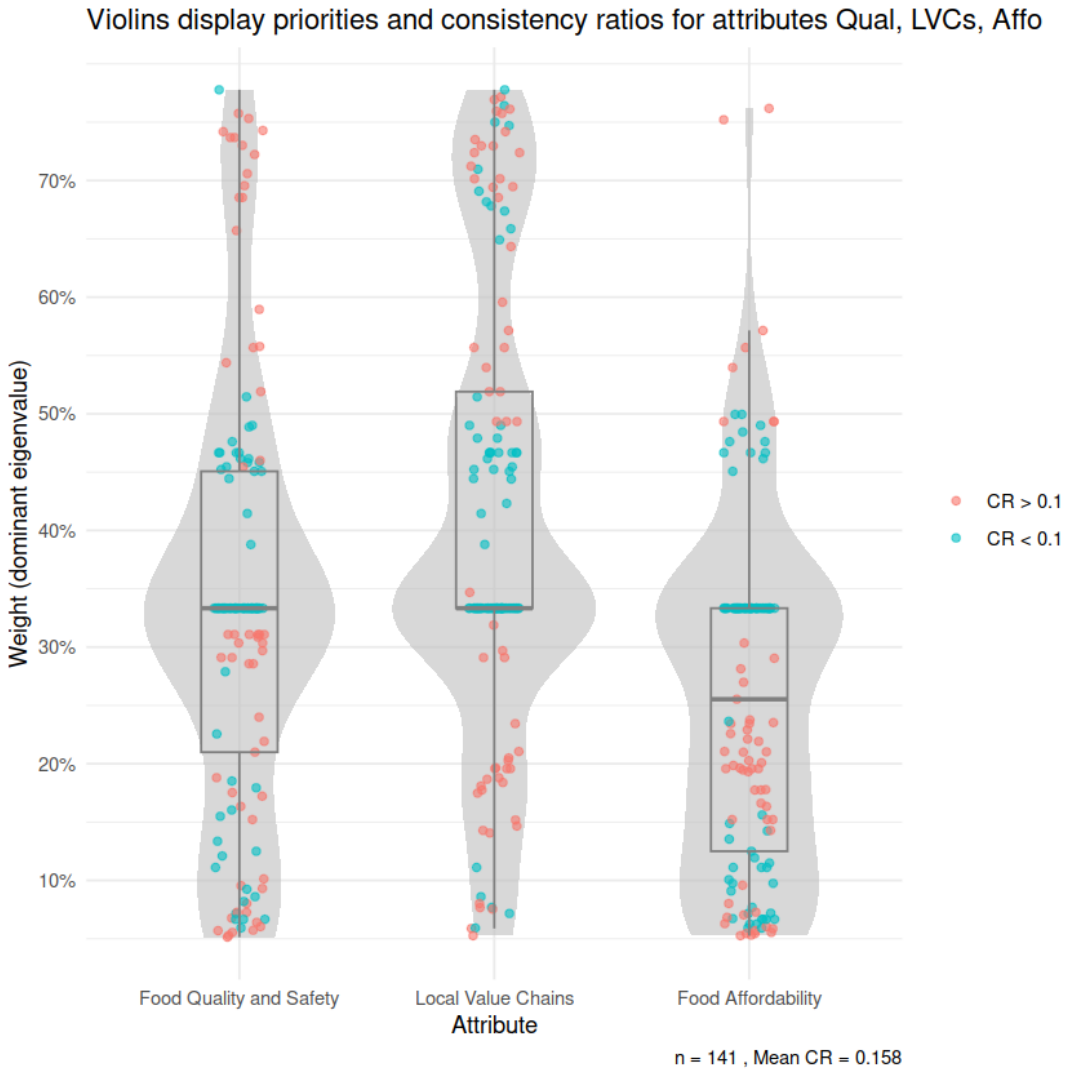


Figure 4: Series of violin plots that display the priorities and consistency ratios of all participants

5.2 Participants of city administrations

[73]:
```
func_process_ahpsurvey(df_csvInputFiles_CA, "CA", str_input_path, str_output_path,␣
↪df_attributes_labels)
```

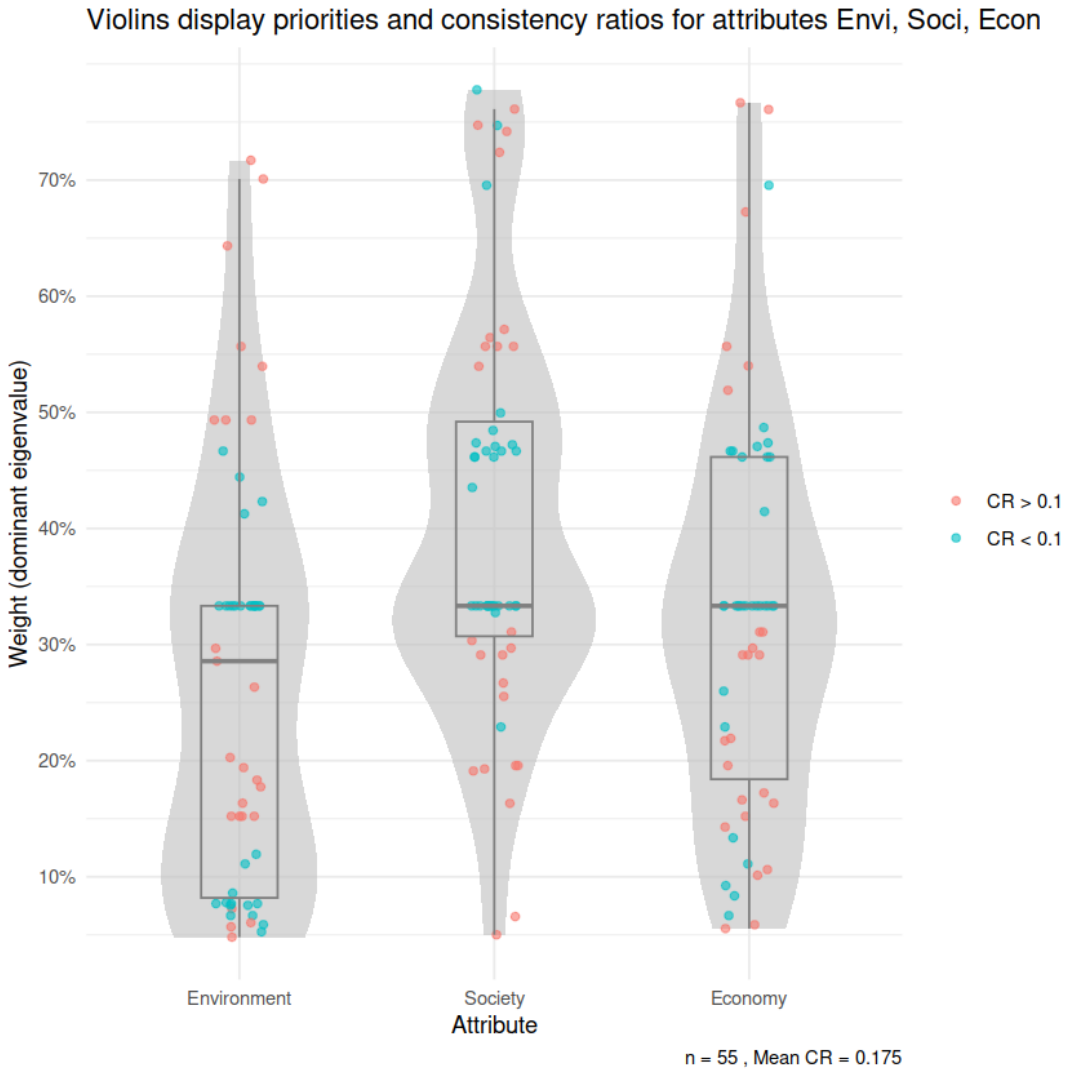


Figure 5: Series of violin plots that display the priorities and consistency ratios of city administrations

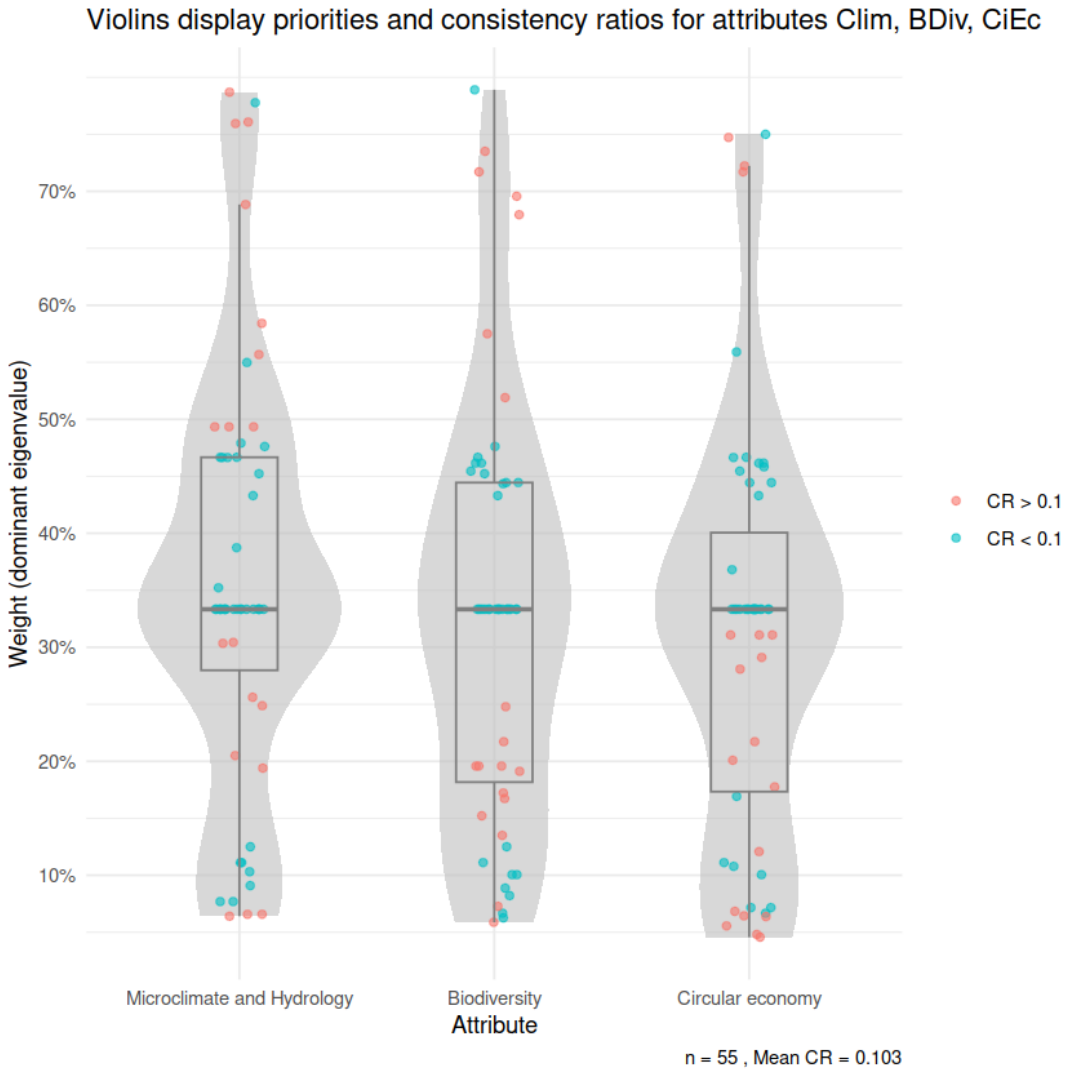


Figure 6: Series of violin plots that display the priorities and consistency ratios of city administrations

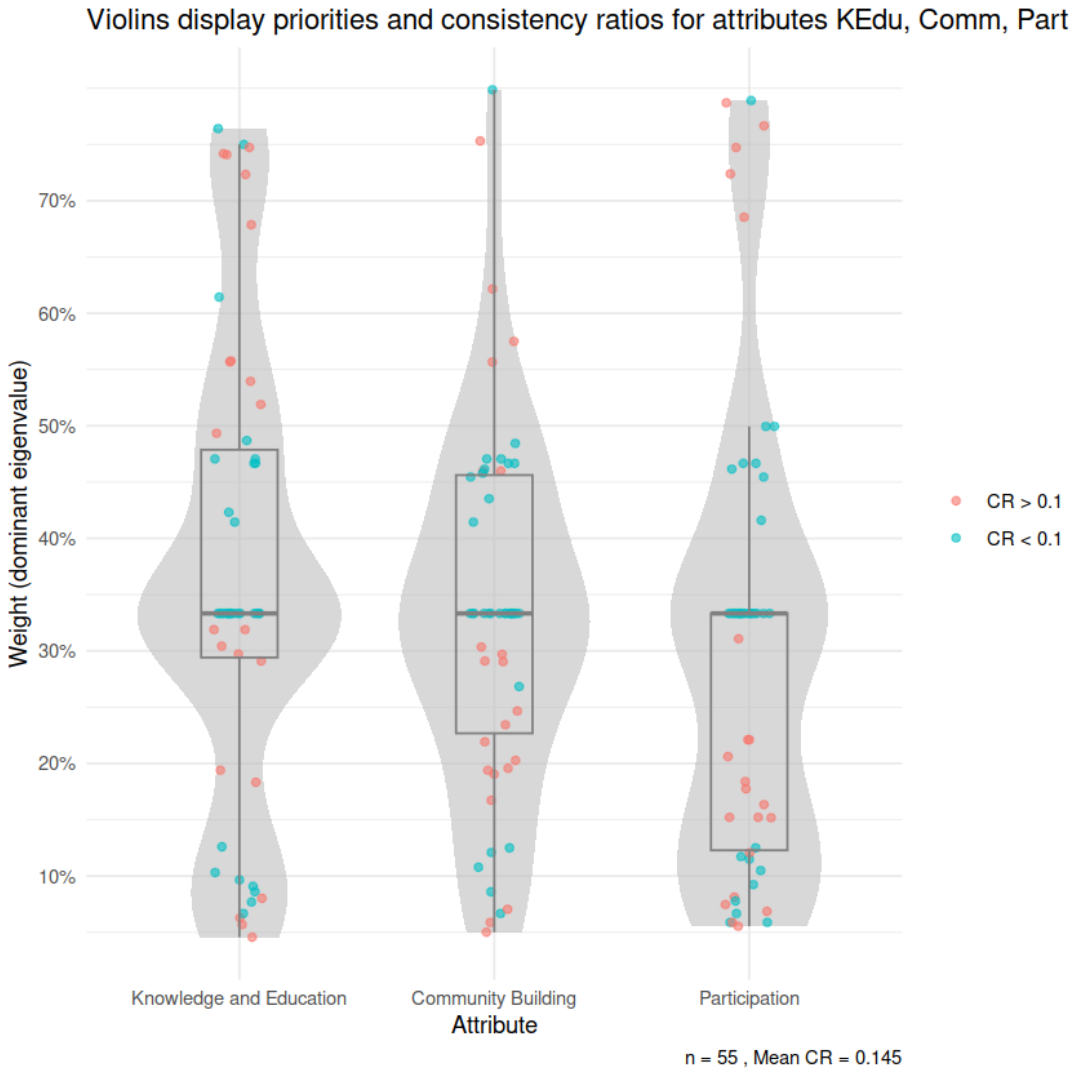


Figure 7: Series of violin plots that display the priorities and consistency ratios of city administrations

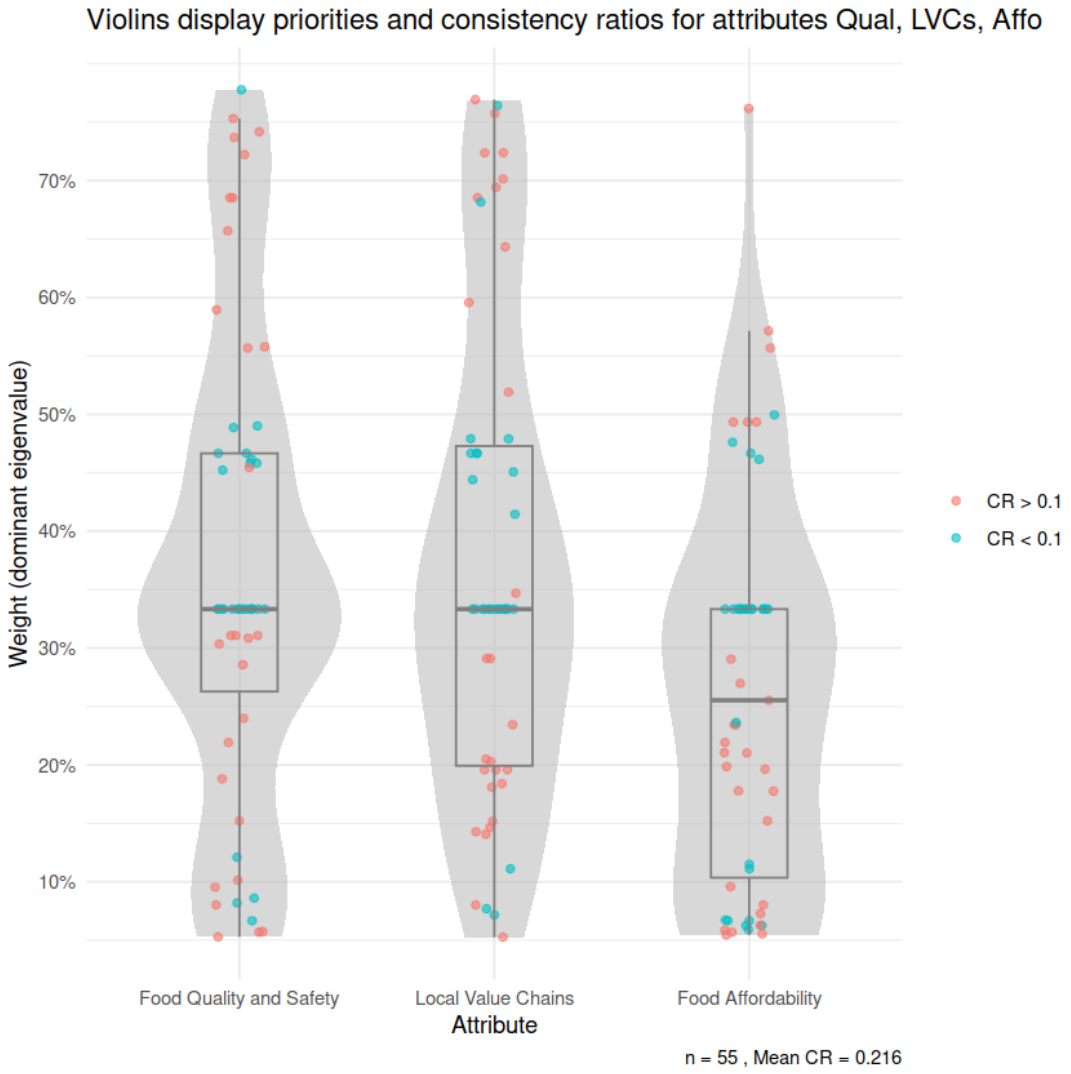


Figure 8: Series of violin plots that display the priorities and consistency ratios of city administrations

5.3 Participants of non-governmental organizations

[74]:
```
func_process_ahpsurvey(df_csvInputFiles_NGO, "NGO", str_input_path,␣
↪str_output_path, df_attributes_labels)
```

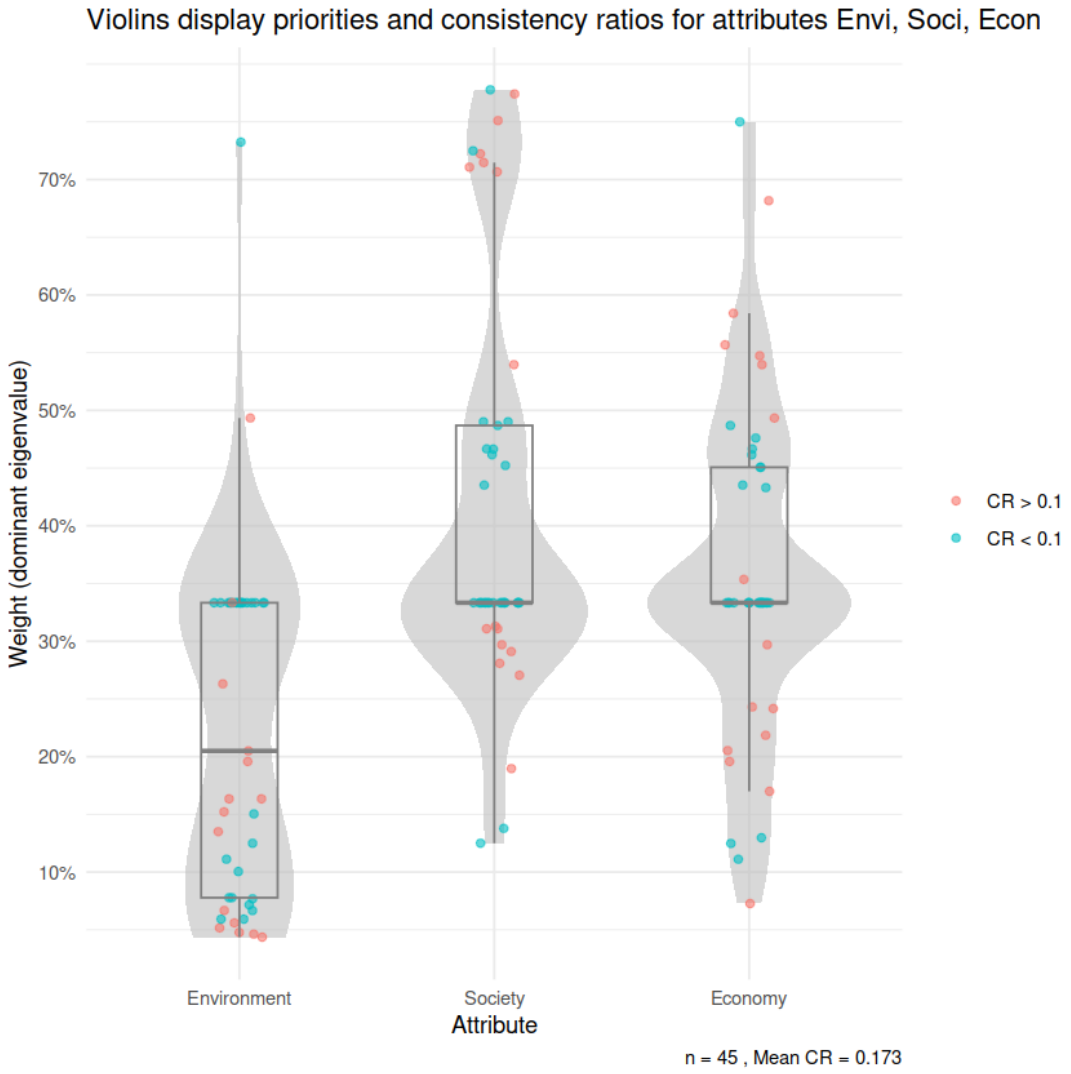


Figure 9: Series of violin plots that display the priorities and consistency ratios of non-governmental organizations

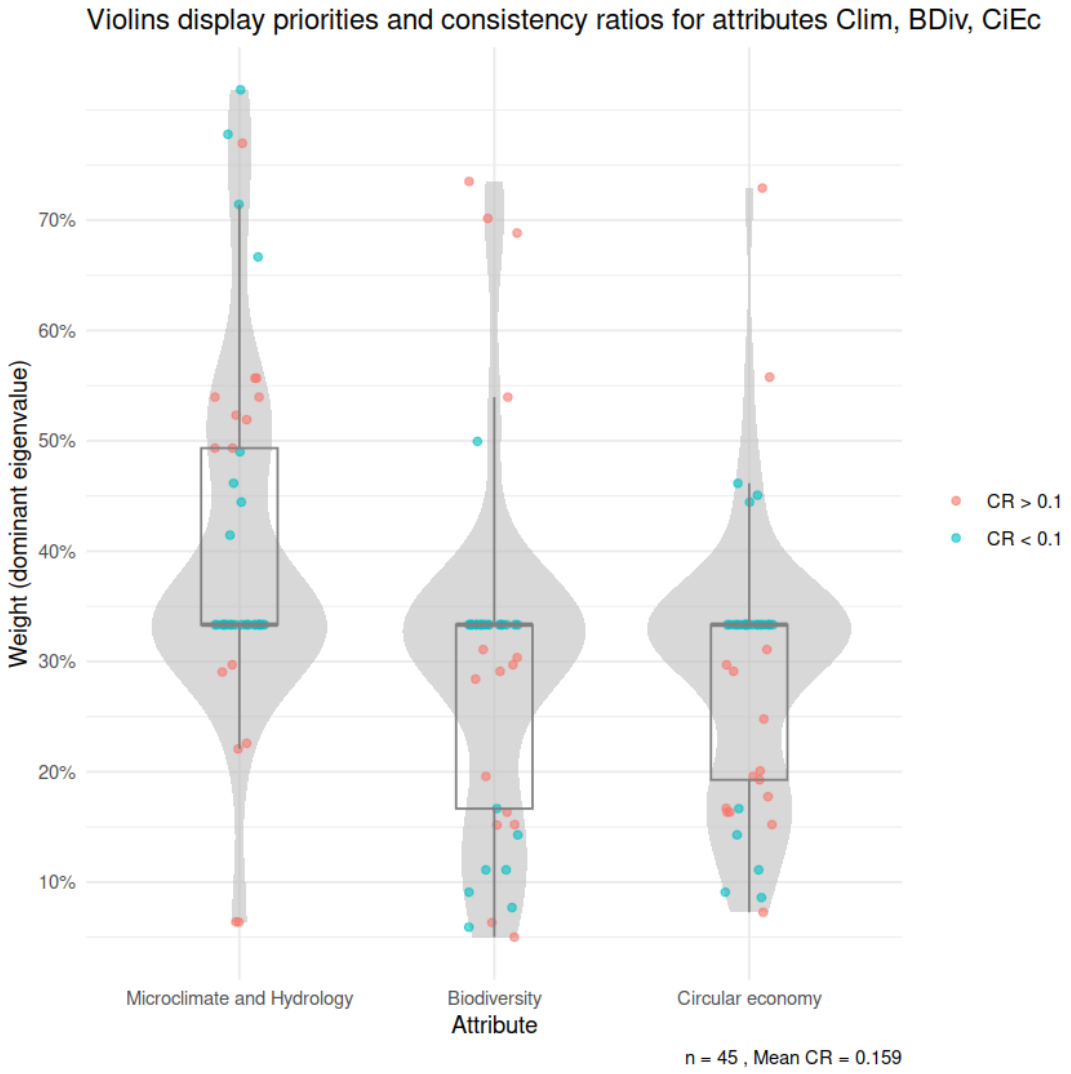


Figure 10: Series of violin plots that display the priorities and consistency ratios of non-governmental organizations

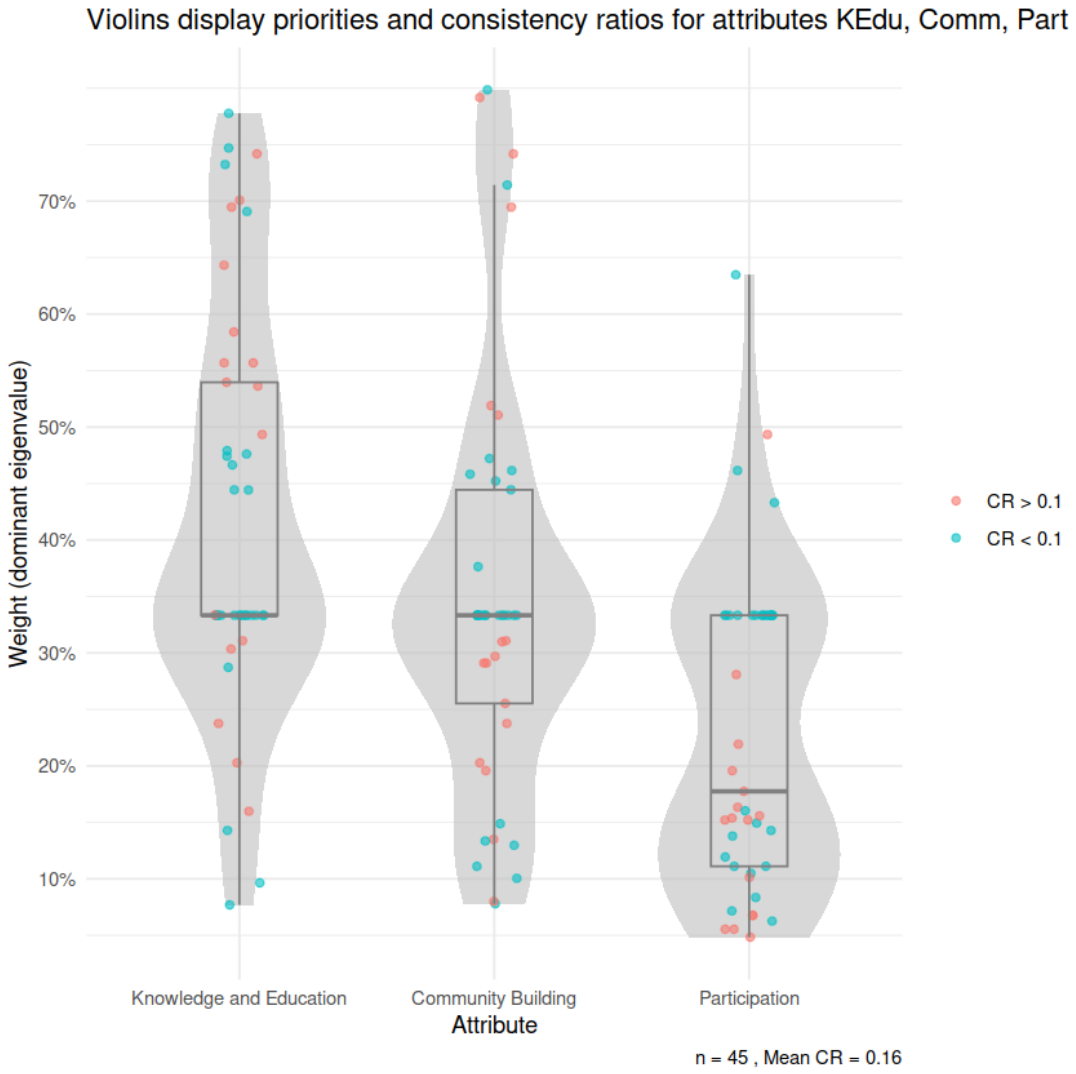


Figure 11: Series of violin plots that display the priorities and consistency ratios of non-governmental organizations

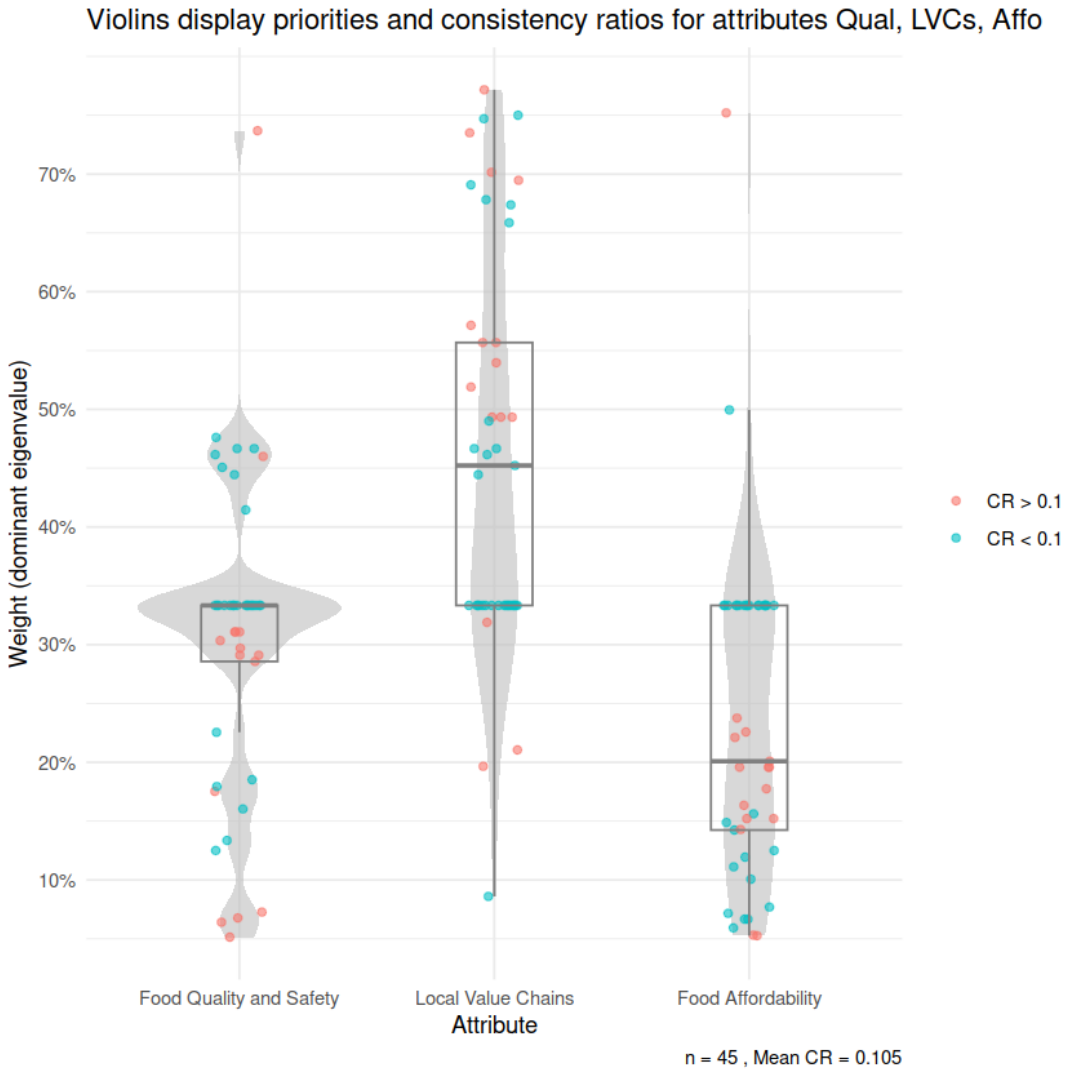


Figure 12: Series of violin plots that display the priorities and consistency ratios of non-governmental organizations

5.4 Participants of practitioners and experts

[75]:
```
func_process_ahpsurvey(df_csvInputFiles_PE, "PE", str_input_path, str_output_path,␣
↪df_attributes_labels)
```

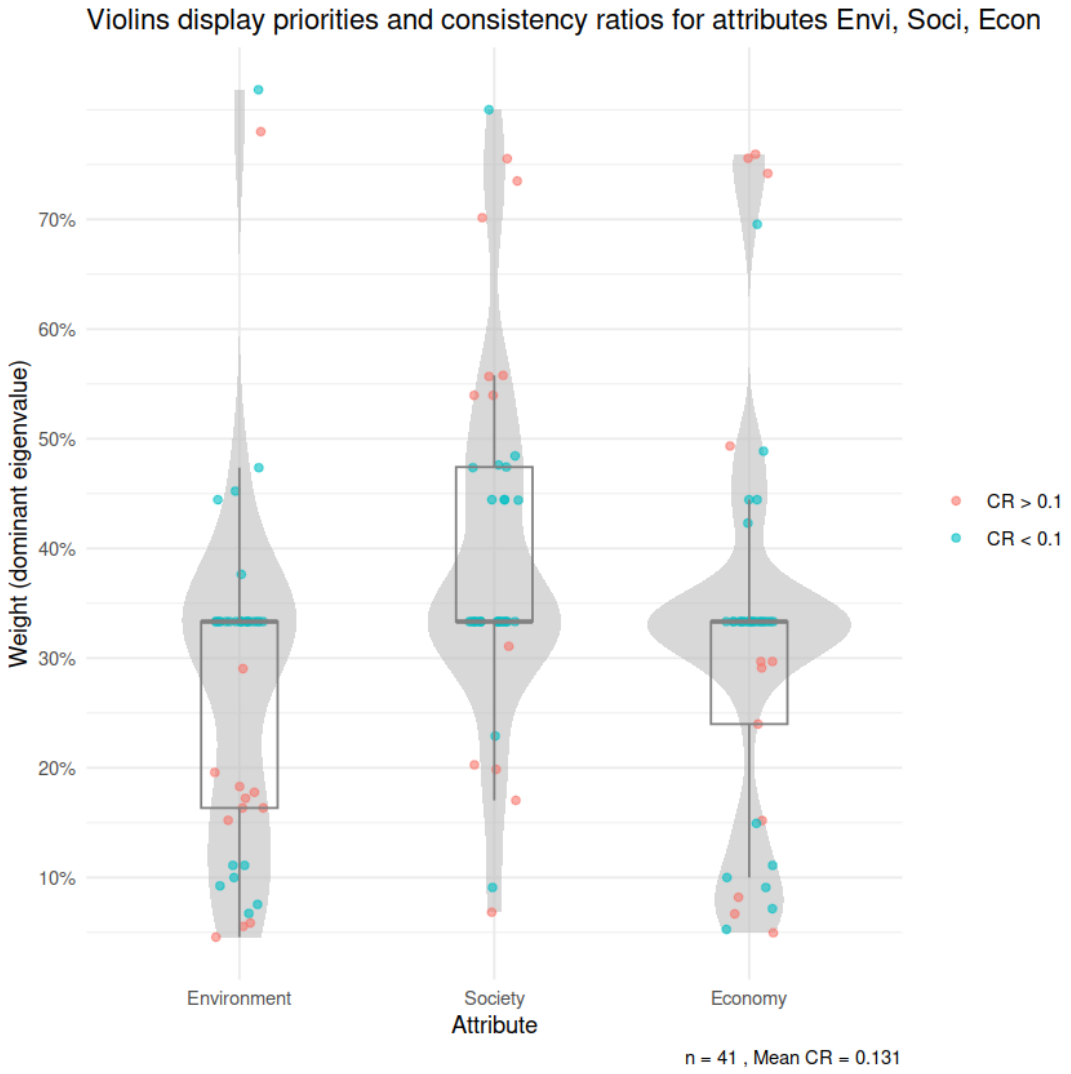


Figure 13: Series of violin plots that display the priorities and consistency ratios of practitioners and experts

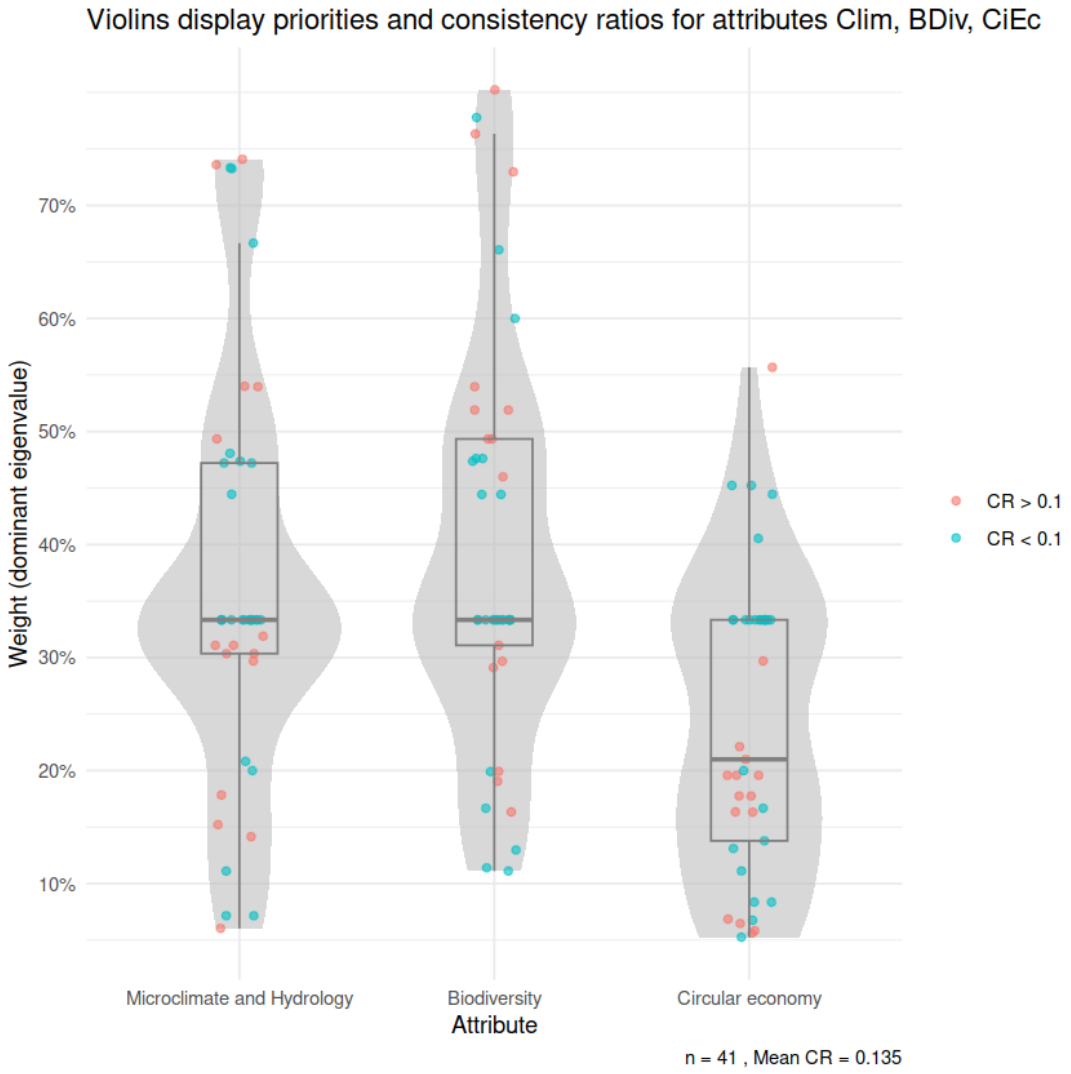


Figure 14: Series of violin plots that display the priorities and consistency ratios of practitioners and experts

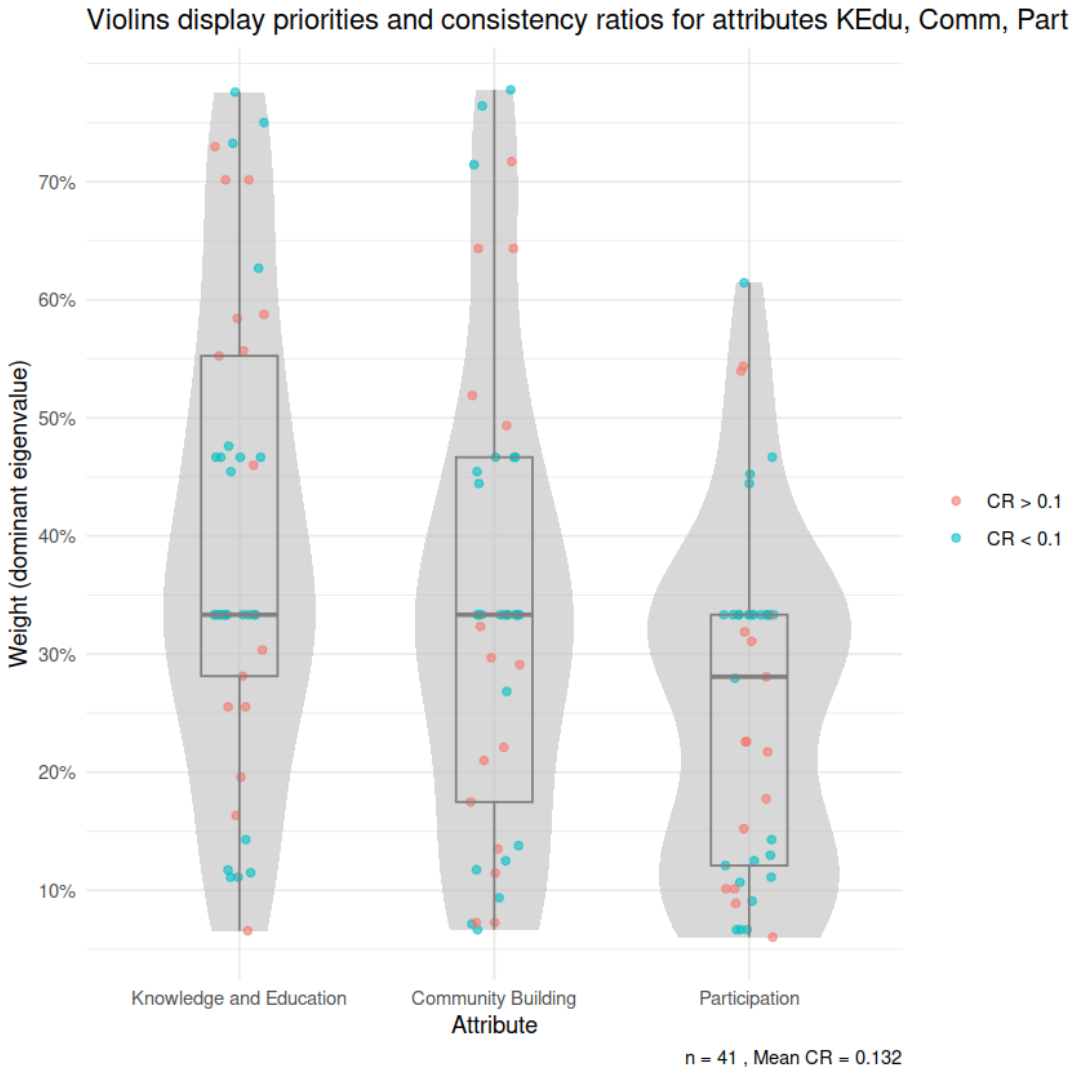


Figure 15: Series of violin plots that display the priorities and consistency ratios of practitioners and experts

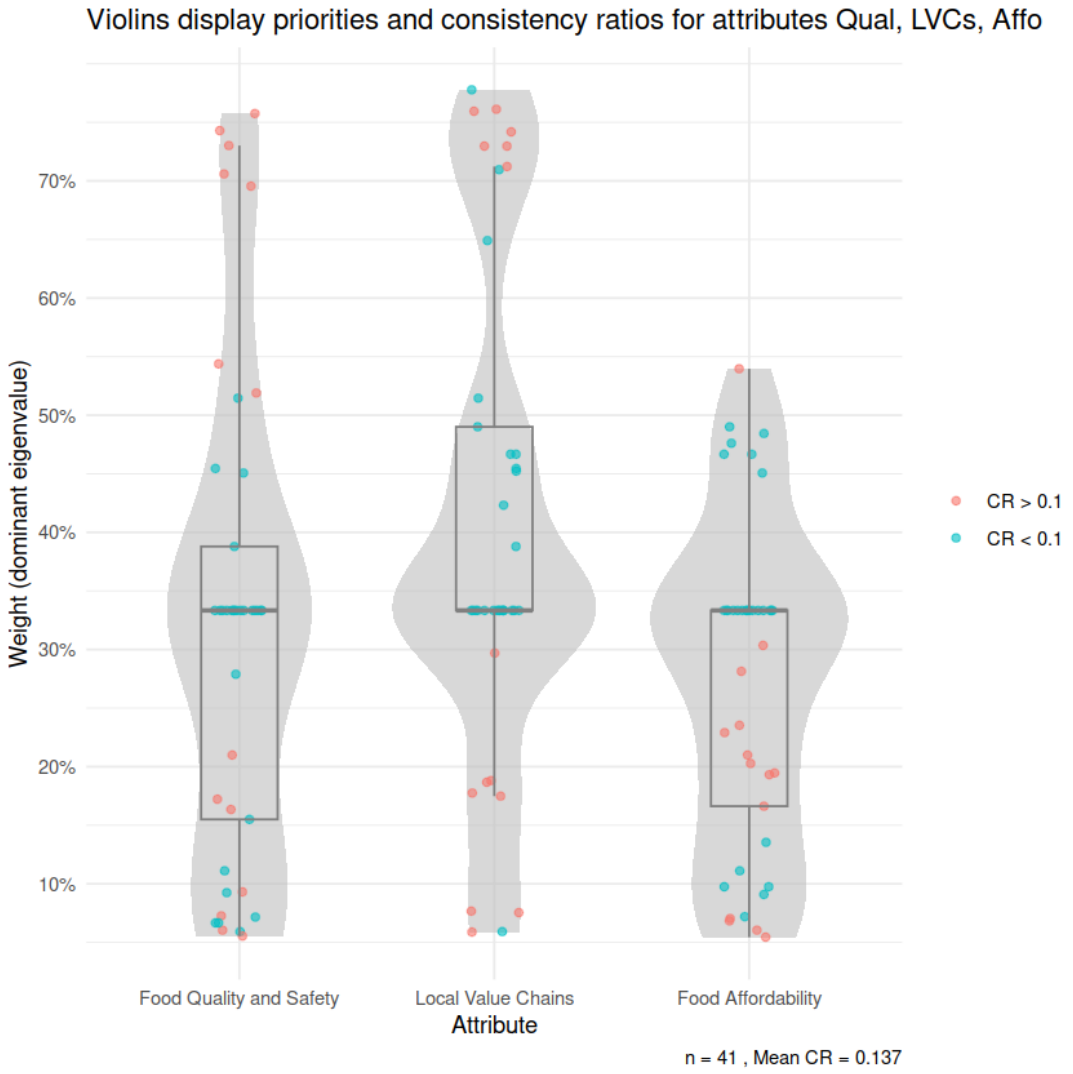


Figure 16: Series of violin plots that display the priorities and consistency ratios of practitioners and experts

5.5 Calculation of aggregated preference weights for consistent datasets

[123]:
```
df_csvInputFiles <- df_csvInputFiles_all
row_start = 1
row_end = 3

for ( file_idx in 1:nrow(df_csvInputFiles) ) {
    # create dataframe from current input CSV file
    str_filename <- paste(str_input_path, df_csvInputFiles[file_idx, filenames],␣
 ↪sep="/")
    str_file_description <- df_csvInputFiles[file_idx, descriptions]

    df_processed_survey_data <- func_readCSVdata_to_dataframe(str_filename)

    # create vectors for attributes and labels from a subset of data frame␣
 ↪'df_attributes_labels_XXX'
    vec_atts <- df_attributes_labels[c(row_start:row_end), attr]
    vec_labels <- df_attributes_labels[c(row_start:row_end), labels]
```

```
    # shift row interval for next iteration
    row_start = row_start + 3
    row_end = row_end + 3

    # generate an array with consistency ratios
    arr_CRs <- func_genCR_to_arr(df_processed_survey_data, vec_atts)

    str_CRlabel <- paste("CR", df_csvInputFiles[file_idx, keys], sep="_")

    list_gmean <- func_aggpref_gmean(df_processed_survey_data, vec_atts, arr_CRs,␣
 ↪consistency_thres=0.1, str_CRlabel)
    df_gmean <- func_norm_gmean(list_gmean)

    cat(sprintf("Aggregated preference weights for %s:\n", str_file_description))
    print.data.frame(df_gmean, right=TRUE)
    cat("\n")
}
```

```
Aggregated preference weights for criteria (main criteria):
     gmean.raw gmean.norm
Envi 0.3739039  0.4179807
Soci 0.3023657  0.3380094
Econ 0.2182787  0.2440099
Sum  0.8945482  1.0000000

Aggregated preference weights for environmental sub-criteria:
     gmean.raw gmean.norm
Clim 0.2741300  0.3089871
BDiv 0.3291718  0.3710277
CiEc 0.2838874  0.3199852
Sum  0.8871892  1.0000000

Aggregated preference weights for social sub-criteria:
     gmean.raw gmean.norm
KEdu 0.3018698  0.3512801
Comm 0.3212634  0.3738480
Part 0.2362090  0.2748719
Sum  0.8593422  1.0000000

Aggregated preference weights for economic sub-criteria:
     gmean.raw gmean.norm
Qual 0.2329901  0.2625724
LVCs 0.3666029  0.4131498
Affo 0.2877435  0.3242777
Sum  0.8873365  1.0000000
```

6 Summary and outlook

7 References

Online references

Romer, Paul (Apr. 13, 2018). *Jupyter, Mathematica, and the Future of the Research Paper*. English. URL: https://paulromer.net/jupyter-mathematica-and-the-future-of-the-research-paper/ (visited on 09/08/2022) (cit. on p. 2).

Somers, James (Apr. 5, 2018). *The Scientific Paper Is Obsolete*. English. The Atlantic. URL: https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/ (visited on 09/08/2022) (cit. on p. 2).