

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

Björn Kasper*

Henriette John†

April 10, 2022

Contents

1	Global settings and dependencies	2
1.1	Load package <code>data.table</code>	2
1.2	Load <code>ggplot2</code>	2
1.3	Load <code>tidyr</code> for <i>gather()</i> function	2
1.4	Load <code>dplyr</code> for data manipulation	2
1.5	Load <code>magick</code> for image manipulation	2
1.6	Use pipes for better coding	3
1.7	Load package <code>ahpsurvey</code>	3
2	Functions for processing AHP	3
2.1	Set globally used input and output folders	3
2.2	Function for reading in processed survey data from CSV files to data frames	3
2.3	Function for generating a data frame with <i>eigentrue values</i> (weights)	3
2.4	Function for generating an array with consistency ratios	4
2.5	Function for generating a data frame with consistency ratios	4
2.6	Function for visualising individual priorities and consistency ratios	4
3	Create data frames (tables) handling the file names of processed survey data	5
3.1	File table for all participants	5
3.2	File table for city administrations	5
3.3	File table for non-governmental organisations	5
3.4	File table for practitioners and experts	6
4	Exploit datasets of own survey with package <code>ahpsurvey</code> for each group of participants	6
4.1	All participants	6
4.2	Participants of city administrations	7
4.3	Participants of non-governmental organisations	8
4.4	Participants of practitioners and experts	9
4.5	Playground for inserting images	11

*BG ETEM, kasper.bjoern@bgetem.de

†IÖR, h.john@ioer.de

1 Global settings and dependencies

1.1 Load package `data.table`

The package `data.table` is used for reading and manipulating tables (`data.table` inherits from `data.frame`). Install and load it:

```
# install.packages("data.table")
library(data.table)
```

1.2 Load `ggplot2`

The package `ggplot2` is used for plotting diagrams. Install and load it:

```
# install.packages("ggplot2")
library(ggplot2)
```

1.3 Load `tidyr` for *gather()* function

```
# install.packages("tidyr")
library(tidyr)
```

1.4 Load `dplyr` for data manipulation

Load necessary library `dplyr` for data manipulation with functions like `select()`, `mutate()` and `left_join()`.

```
# install.packages("dplyr")
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'
## Die folgenden Objekte sind maskiert von 'package:data.table':
##
##      between, first, last
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##      filter, lag
## Die folgenden Objekte sind maskiert von 'package:base':
##
##      intersect, setdiff, setequal, union
```

1.5 Load `magick` for image manipulation

The library `magick` is the R API to *ImageMagick*.

```
# install.packages('magick')
library(magick)
```

```
## Linking to ImageMagick 6.9.10.23
## Enabled features: fontconfig, freetype, fftw, lcms, pango, webp, x11
## Disabled features: cairo, ghostscript, heic, raw, rsvg
## Using 4 threads
```

1.6 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`:

```
# install.packages("magrittr")
library(magrittr)

##
## Attache Paket: 'magrittr'
## Das folgende Objekt ist maskiert 'package:tidyr':
##
##      extract
```

1.7 Load package `ahpsurvey`

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

```
# install.packages("ahpsurvey")
library(ahpsurvey)
```

2 Functions for processing AHP

2.1 Set globally used input and output folders

```
str_input_path = "./output_data_manipulated"
str_output_path = "./output_data_AHP"
```

2.2 Function for reading in processed survey data from CSV files to data frames

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

```
func_readCSVdata_to_dataframe <- function(str_CSVfilename) {

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

  return(df_CSVdata)
}
```

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

```
func_genEigentrue_to_dataframe <- function(df_surveyData, vec_attributes) {
  my_cr_eigen <- df_surveyData %>%
    ahp.mat(vec_attributes, negconvert = TRUE)

  df_eigentrue <- ahp.indpref(my_cr_eigen, vec_attributes, method = "eigen")
}
```

```

    return(df_eigentrue)
}

```

2.4 Function for generating an array with consistency ratios

```

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)
}

```

2.5 Function for generating a data frame with consistency ratios

```

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)
}

```

2.6 Function for visualising individual priorities and consistency ratios

```

func_visuPriosCRs <- function(df_surveyData, df_cr, arr_cr, consistency_thres=0.1, vec_attributes, df_eigentrue) {
  df_cr_sel <- df_cr %>%
    select(arr_cr.dum, rowid)

  df_surveyData %>%
    ahp.mat(attrs = 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("Violins displaying priorities and consistency ratios")

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

}

```

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

3.1 File table for all participants

```

df_csvInputFiles_all <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_all_env_AHP_essbare_Stadt_2022-03-18_09-53.csv",
                "rdata_all_soc_AHP_essbare_Stadt_2022-03-18_09-53.csv",
                "rdata_all_eco_AHP_essbare_Stadt_2022-03-18_09-53.csv",
                "rdata_all_crit_AHP_essbare_Stadt_2022-03-18_09-53.csv"),
  descriptions = c("environmental sub-criteria",
                  "social sub-criteria",
                  "economic sub-criteria",
                  "criteria (main criteria)")
)

```

3.2 File table for city administrations

```

df_csvInputFiles_CA <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_CA_env_AHP_essbare_Stadt_2022-03-18_10-28.csv",
                "rdata_CA_soc_AHP_essbare_Stadt_2022-03-18_10-28.csv",
                "rdata_CA_eco_AHP_essbare_Stadt_2022-03-18_10-28.csv",
                "rdata_CA_crit_AHP_essbare_Stadt_2022-03-18_10-28.csv"),
  descriptions = c("environmental sub-criteria",
                  "social sub-criteria",
                  "economic sub-criteria",
                  "criteria (main criteria)")
)

```

3.3 File table for non-governmental organisations

```

df_csvInputFiles_NGO <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_NGO_env_AHP_essbare_Stadt_2022-03-18_10-40.csv",
                "rdata_NGO_soc_AHP_essbare_Stadt_2022-03-18_10-40.csv",

```

```

        "rdata_NGO_eco_AHP_essbare_Stadt_2022-03-18_10-40.csv",
        "rdata_NGO_crit_AHP_essbare_Stadt_2022-03-18_10-40.csv"),
descriptions = c("environmental sub-criteria",
                 "social sub-criteria",
                 "economic sub-criteria",
                 "criteria (main criteria)")
)

```

3.4 File table for practitioners and experts

```

df_csvInputFiles_PE <- data.table(
  file_idx = 1:4,
  keys = c("env", "soc", "eco", "crit"),
  filenames = c("rdata_PE_env_AHP_essbare_Stadt_2022-03-18_10-41.csv",
               "rdata_PE_soc_AHP_essbare_Stadt_2022-03-18_10-41.csv",
               "rdata_PE_eco_AHP_essbare_Stadt_2022-03-18_10-41.csv",
               "rdata_PE_crit_AHP_essbare_Stadt_2022-03-18_10-41.csv"),
  descriptions = c("environmental sub-criteria",
                  "social sub-criteria",
                  "economic sub-criteria",
                  "criteria (main criteria)")
)

```

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

```

df_attributes_labels_all <- data.table(
  idx = 1:12,
  attr = c("Klima", "BioV", "KlW",
           "Wiss", "Gem", "Bet",
           "Quali", "WSK", "Bez",
           "Oeko", "Soz", "Wirt"),
  labels = c("Klima", "Biologische Vielfalt", "Kreislaufwirtschaft",
            "Wissensvermittlung", "Gemeinschaftsbildung", "Beteiligung",
            "Lebensmittelqualitaet", "lokale Wertschoepfungsketten", "Bezahlbarkeit",
            "Oekologisch", "Sozial", "Wirtschaftlich")
)

```

4.1 All participants

```

row_start = 1
row_end = 3

str_participants_group = "all"

df_outputTable <- data.table()

for ( file_idx in 1:nrow(df_csvInputFiles_all) ) {
  # create data frame from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles_all[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_all'
vec_atts <- df_attributes_labels_all[c(row_start:row_end), attr]
vec_labels <- df_attributes_labels_all[c(row_start:row_end), labels]
# shift row interval for next iteration
row_start = row_start + 3
row_end = row_end + 3

# generate data frame with eigenttrue values (weights)
df_eigenttrue_weights <- func_genEigenttrue_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 a extended data frame with consistency ratios
consistency_thres = 0.1
str_CRlabel <- paste("CR", df_csvInputFiles_all[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_PE[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_eigenttrue_weights)

# combine data frames of eigenttrue values (weights) with consistency ratios
df_outputTable <- cbind(df_outputTable, df_eigenttrue_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, "eigenttrue_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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)

```

4.2 Participants of city administrations

```

row_start = 1
row_end = 3

str_participants_group = "CA"

df_outputTable <- data.table()

for ( file_idx in 1:nrow(df_csvInputFiles_CA) ) {
  # create data frame from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles_CA[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_all'
vec_atts <- df_attributes_labels_all[c(row_start:row_end), attr]
vec_labels <- df_attributes_labels_all[c(row_start:row_end), labels]
# shift row interval for next iteration
row_start = row_start + 3
row_end = row_end + 3

# generate data frame with eigenttrue values (weights)
df_eigenttrue_weights <- func_genEigenttrue_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 a extended data frame with consistency ratios
consistency_thres = 0.1
str_CRlabel <- paste("CR", df_csvInputFiles_CA[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_PE[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_eigenttrue_weights)

# combine data frames of eigenttrue values (weights) with consistency ratios
df_outputTable <- cbind(df_outputTable, df_eigenttrue_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, "eigenttrue_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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)

```

4.3 Participants of non-governmental organisations

```

row_start = 1
row_end = 3

str_participants_group = "NGO"

df_outputTable <- data.table()

for ( file_idx in 1:nrow(df_csvInputFiles_NGO) ) {
  # create data frame from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles_NGO[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_all'
vec_atts <- df_attributes_labels_all[c(row_start:row_end), attr]
vec_labels <- df_attributes_labels_all[c(row_start:row_end), labels]
# shift row interval for next iteration
row_start = row_start + 3
row_end = row_end + 3

# generate data frame with eigenttrue values (weights)
df_eigenttrue_weights <- func_genEigenttrue_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 a extended data frame with consistency ratios
consistency_thres = 0.1
str_CRlabel <- paste("CR", df_csvInputFiles_NGO[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_PE[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_eigenttrue_weights)

# combine data frames of eigenttrue values (weights) with consistency ratios
df_outputTable <- cbind(df_outputTable, df_eigenttrue_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, "eigenttrue_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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)

```

4.4 Participants of practitioners and experts

```

row_start = 1
row_end = 3

str_participants_group = "PE"

df_outputTable <- data.table()

for ( file_idx in 1:nrow(df_csvInputFiles_PE) ) {
  # create data frame from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles_PE[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_all'
vec_atts <- df_attributes_labels_all[c(row_start:row_end), attr]
vec_labels <- df_attributes_labels_all[c(row_start:row_end), labels]
# shift row interval for next iteration
row_start = row_start + 3
row_end = row_end + 3

# generate data frame with eigentru values (weights)
df_eigentru_weights <- func_genEigentru_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 a extended data frame with consistency ratios
consistency_thres = 0.1
str_CRlabel <- paste("CR", df_csvInputFiles_PE[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_PE[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_eigentru_weights, str_image_filename)

img <- image_graph(width = 800, height = 800, res = 24)
img <- image_read(str_image_filename)
print(img)

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

## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>   <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      2100    2100 sRGB        TRUE     189485 118x118

## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>   <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      2100    2100 sRGB        TRUE     194715 118x118

## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>   <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      2100    2100 sRGB        TRUE     188161 118x118

## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>   <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      2100    2100 sRGB        TRUE     176500 118x118

# extend file name by path
str_CSVfilename_output <- paste("rdata", str_participants_group, "eigentru_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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)

```

4.5 Playground for inserting images

```

# # install.packages('magick')
# library(magick)
#
# # img <- image_read(str_image_filename)
# # image_info(img)
# # img <- image_scale(img, "600")
# img <- image_graph(width = 800, height = 800, res = 24)
# img <- image_read(str_image_filename)
#
# print(img) # or print(img)
# # image_display(img)

# ggplot2::qplot(factor(cyl), data = mtcars, fill = factor(gear))
#
# # Produce graphic
# fig <- image_graph(width = 800, height = 600, res = 96)
# ggplot2::qplot(factor(cyl), data = mtcars, fill = factor(gear))
# invisible(dev.off())
#
# print(fig)

# library(ggplot2)
#
# ggdraw() +
#   draw_image(str_image_filename)

```