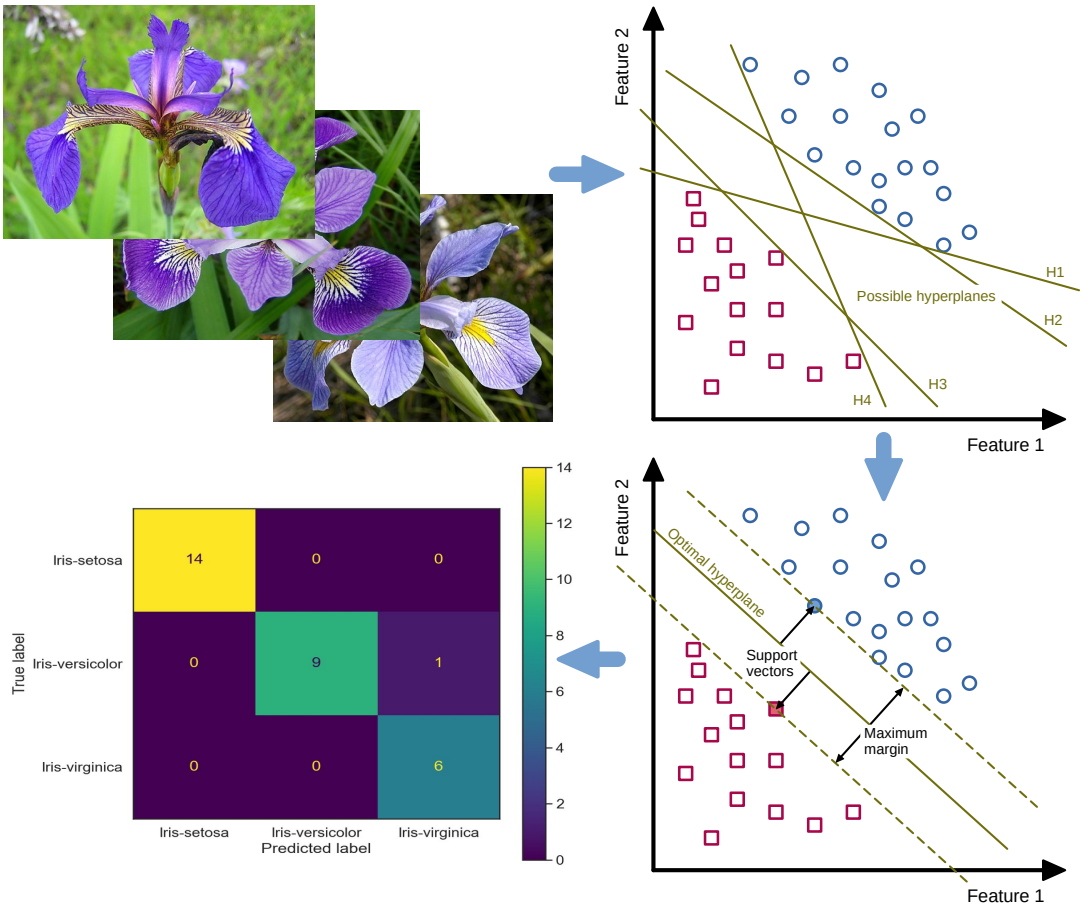
Application of the processed survey data in the analytical hierarchy process (AHP)

Dipl.-Ing. Björn Kasper (kasper.bjoern@bgetem.de)

BG ETEM

November 12, 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 becoming 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 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:

[1]:
```
# install.packages("data.table")
library(data.table)
```

2.2 Load `ggplot2`

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

[2]:
```
# install.packages("ggplot2")
library(ggplot2)
```

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

[3]:
```
# install.packages("tidyr")
library(tidyr)
```

2.4 Load `dplyr` for data manipulation

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

[5]:
```
# install.packages("dplyr")
library(dplyr)
```

2.5 Load `magick` for image manipulation

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

[7]:
```
# install.packages('magick')
library(magick)
```

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`:

[9]:
```
# install.packages("magrittr")
library(magrittr)
```

2.7 Load package `ahpsurvey`

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

[10]:
```
# install.packages("ahpsurvey")
library(ahpsurvey)
```

3 Functions for processing AHP

3.1 Set globally used input and output folders

[11]:
```
str_input_path = "./output_data_manipulated"
str_output_path = "./output_data_AHP"
```

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

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

[12]:
```
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)

[13]:
```
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

[14]:
```
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

[15]:
```
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

[16]:
```
func_visuPriosCRs <- function(df_surveyData, df_cr, arr_cr, consistency_thres=0.1,␣
 ↪vec_attributes, df_eigentrue, vec_labels, str_image_filename) {
  df_cr_sel <- df_cr %>%
    select(arr_cr.dum, rowid)

  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("Violins displaying priorities and consistency ratios")

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

3.7 Function for generating geometric mean values from individual judgement matrices

[17]:
```
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

[18]:
```
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)
}
```

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

4.1 File table for all participants

[19]:
```
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)")
)
```

4.2 File table for city administrations

[20]:
```
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)")
)
```

4.3 File table for non-governmental organizations

[21]:
```
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)")
)
```

4.4 File table for practitioners and experts

[22]:
```
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)")
)
```

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

[23]:
```
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")
)
```

5.1 All participants

[24]:
```
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 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 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_eigentrue_weights, vec_labels, str_image_filename)

  # 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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)
```

[25]:
```
list_gmean <- func_aggpref_gmean(df_processed_survey_data, vec_atts, arr_CRs,␣
 ↪consistency_thres, str_CRlabel)

df_gmean <- func_norm_gmean(list_gmean)

df_gmean
```

A data.frame: 4 × 2

	gmean.raw <dbl>	gmean.norm <dbl>
Oeko	0.3739039	0.4179807
Soz	0.3023657	0.3380094
Wirt	0.2182787	0.2440099
Sum	0.8945482	1.0000000

5.2 Participants of city administrations

[26]:
```
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 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 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_eigentrue_weights, vec_labels, str_image_filename)

  # 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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)
```

5.3 Participants of non-governmental organizations

[27]:
```
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 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 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_eigentrue_weights, vec_labels, str_image_filename)

  # 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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)
```

5.4 Participants of practitioners and experts

[28]:
```
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 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 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_eigentrue_weights, vec_labels, str_image_filename)

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

  # 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
write.table(df_outputTable, file = str_CSVfilename_output,
            fileEncoding = "UTF-8", row.names = FALSE,
            col.names = TRUE, sep = "\t", quote = TRUE)
```

5.5 Playground for inserting images

[29]:
```
# install.packages('magick')
library(magick)

# img <- image_read(str_image_filename)
# image_info(img)
# img <- image_scale(img, "600")
img <- image_graph(width = 400, height = 400, res = 12)
img <- image_read(str_image_filename)

# print(img)
image_display(img)
```

```
Error in magick_image_display(image, animate): R: unable to open X server `' @ erro
  ↪animate.c/AnimateImages/301
Traceback:

1. image_display(img)
2. magick_image_display(image, animate)
```

[30]:
```
# # 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)
```

[31]:
```
# library(ggplot2)
# # install.packages("png")
# library(png)
#
# img1 <- readPNG(str_image_filename, native = TRUE, info = FALSE)
# print(img1)
```

[32]:
```
# #install.packages("rsvg")
# library(rsvg)
# library(magick)
# tiger <- image_read_svg('http://jeroen.github.io/images/tiger.svg', width = 350)
# print(tiger)
```

[ ]:
```

```

6 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).