Preparing raw CSV input data from survey for analytical hierarchy process (AHP)

Björn Kasper* Henriette John†

April 10, 2022

Contents

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 Create data frame (table) handling the file names of input CSV data

```
df_csvInputFiles <- data.table(
  file_idx = 1:4,
  keys = c("all", "CA", "NGO", "PE"),
  filenames = c("rdata_all_AHP_essbare_Stadt_2022-03-18_09-53.csv",
                "rdata_CA_AHP_essbare_Stadt_2022-03-18_10-28.csv",
                "rdata_NGO_AHP_essbare_Stadt_2022-03-18_10-40.csv",
                "rdata_PE_AHP_essbare_Stadt_2022-03-18_10-41.csv"),
```

*BG ETEM, kasper.bjoern@bgetem.de
†IOER, h.john@ioer.de

```
  descriptions = c("all target groups together",
                   "from city administrations",
                   "from non-governmental organisations",
                   "practitioners and experts")
)

df_csvInputFiles
```

```
##    file_idx keys                                          filenames
## 1:        1  all rdata_all_AHP_essbare_Stadt_2022-03-18_09-53.csv
## 2:        2   CA  rdata_CA_AHP_essbare_Stadt_2022-03-18_10-28.csv
## 3:        3  NGO rdata_NGO_AHP_essbare_Stadt_2022-03-18_10-40.csv
## 4:        4   PE  rdata_PE_AHP_essbare_Stadt_2022-03-18_10-41.csv
##                           descriptions
## 1:          all target groups together
## 2:           from city administrations
## 3: from non-governmental organisations
## 4:           practitioners and experts
```

1.3 Set globally used input an doutput folders

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

2 Functions for manipulation of raw CSV input data of survey

2.1 Function for reading in survey data from CSV files to data frame objects

Define a function for reading in a CSV file to 4 different date frames by selecting different columns.

```
func_readCSVdata_to_dataframes <- function(str_CSVfilename) {

  df_mySurvey_1 <- fread(
    file = str_CSVfilename, encoding = "UTF-8",
    header = TRUE, sep = "\t", quote = "\"",
    # dec = ".", row.names = "CASE",
    select = c("CASE", "AU01", "AU02", "AU03",
               "RU01_01", "RU02_01", "RU03_01", "RU04_01", "RU05_01", "RU06_01")
    )

  df_mySurvey_2 <- fread(
    file = str_CSVfilename, encoding = "UTF-8",
    header = TRUE, sep = "\t", quote = "\"",
    # dec = ".", row.names = "CASE",
    select = c("CASE", "AS01", "AS02", "AS03",
               "RS01_01", "RS02_01", "RS03_01", "RS04_01", "RS05_01", "RS06_01")
    )

  df_mySurvey_3 <- fread(
    file = str_CSVfilename, encoding = "UTF-8",
    header = TRUE, sep = "\t", quote = "\"",
    # dec = ".", row.names = "CASE",
    select = c("CASE", "AW01", "AW02", "AW03",
```

```
                "RW01_01", "RW02_01", "RW03_01", "RW04_01", "RW05_01", "RW06_01")
    )

  df_mySurvey_4 <- fread(
    file = str_CSVfilename, encoding = "UTF-8",
    header = TRUE, sep = "\t", quote = "\"",
    # dec = ".", row.var = "CASE",
    select = c("CASE", "AK01", "AK02", "AK03",
                "RK01_01", "RK02_01", "RK03_01", "RK04_01", "RK05_01", "RK06_01")
    )

  output <- list(df_mySurvey_1, df_mySurvey_2, df_mySurvey_3, df_mySurvey_4)

  return(output)
}
```

2.2 Function for manipulation of the read in data and store in new data frame

```
func_scrambleData <- function(df_inputData, vec_colnames_search_1, vec_colnames_search_2, vec_colnames_
  # Generate new data frame ...
  df_outputData <- data.frame(matrix(ncol = 3, nrow = 0))
  # ... and name the columns
  colnames(df_outputData) <- vec_colnames_out

  # Generate 1. column
  for ( row_idx in 1:nrow(df_inputData) ) {
    # filter column names by vector element
    if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[1], with=FALSE] == 1) {
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_2
      int_tmp_val <- int_tmp_val * -1 - 1

      df_outputData[row_idx, vec_colnames_out[1]] <- int_tmp_val
    }
    else if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[1], with=FALSE] ==
      df_outputData[row_idx, vec_colnames_out[1]] <- 1
    }
    else if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[1], with=FALSE] ==
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_2
      int_tmp_val <- int_tmp_val + 1

      df_outputData[row_idx, vec_colnames_out[1]] <- int_tmp_val
    }
  }

  # Generate 2. column
  for ( row_idx in 1:nrow(df_inputData) ) {
    # filter column names by vector element
    if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[2], with=FALSE] == 1) {
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_2
      int_tmp_val <- int_tmp_val * -1 - 1

      df_outputData[row_idx, vec_colnames_out[2]] <- int_tmp_val
    }
```

```
    else if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[2], with=FALSE] ==
      df_outputData[row_idx, vec_colnames_out[2]] <- 1
    }
    else if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[2], with=FALSE] ==
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_2
      int_tmp_val <- int_tmp_val + 1

      df_outputData[row_idx, vec_colnames_out[2]] <- int_tmp_val
    }
  }

  # Generate 3. column
  for ( row_idx in 1:nrow(df_inputData) ) {
    # filter column names by vector element
    if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[3], with=FALSE] == 1) {
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_2
      int_tmp_val <- int_tmp_val * -1 - 1

      df_outputData[row_idx, vec_colnames_out[3]] <- int_tmp_val
    }
    else if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[3], with=FALSE] ==
      df_outputData[row_idx, vec_colnames_out[3]] <- 1
    }
    else if (df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_1[3], with=FALSE] ==
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in% vec_colnames_search_2
      int_tmp_val <- int_tmp_val + 1

      df_outputData[row_idx, vec_colnames_out[3]] <- int_tmp_val
    }
  }

  # return scrambled data frame
  return(df_outputData)
}
```

Function for writing resulting data frame to CSV file:

```
func_writeDataframe_to_CSVfile <- function(str_path, str_CSVfilename, df_dataframe, str_filenameExtensi
  # Split file name on second underscore, found here:
  # https://stackoverflow.com/questions/32398427/r-split-a-character-string-on-the-second-underscore/323
  list_str_split <- strsplit(sub('(^[^_]+_[^_]+)_(.*)$', '\\1 \\2', str_CSVfilename), ' ')

  # extend the file name prefix and glue together with old suffix
  str_CSVfilename_extended <- paste(list_str_split[[1]][1], str_filenameExtension, list_str_split[[1]][

  # extend file name by path
  str_CSVfilename_extended <- paste(str_path, str_CSVfilename_extended, sep="/")

  write.table(df_dataframe, file = str_CSVfilename_extended,
              fileEncoding = "UTF-8", row.names = FALSE,
              col.names = TRUE, sep = "\t", quote = TRUE)
}
```

3 Manipulate the data and store in new data frames for each criteria

3.1 Environmental sub-criteria

Walk over all input CSV files, manipulate the data, and write the results to output CSV files:

```
vec_colnames_search_1 <- c('AU01', 'AU02', 'AU03')
vec_colnames_search_2 <- c('RU01_01', 'RU02_01', 'RU03_01', 'RU04_01', 'RU05_01', 'RU06_01')
vec_colnames_out <- c('Klima_BioV', 'Klima_KlW', 'BioV_KlW')

for ( row_idx in 1:nrow(df_csvInputFiles) ) {
  # create list of data frames from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles[row_idx, filenames], sep="/")
  list_dataframes <- func_readCSVdata_to_dataframes(str_filename)

  # scramble the data frames
  df_scrambledData <- func_scrambleData(list_dataframes[[1]], vec_colnames_search_1, vec_colnames_searc

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx, filenames], df_scrambledDat
}
```

3.2 Social sub-criteria

Walk over all input CSV files, manipulate the data, and write the results to output CSV files:

```
vec_colnames_search_1 <- c('AS01', 'AS02', 'AS03')
vec_colnames_search_2 <- c('RS01_01', 'RS02_01', 'RS03_01', 'RS04_01', 'RS05_01', 'RS06_01')
vec_colnames_out <- c('Wiss_Gem', 'Wiss_Bet', 'Gem_Bet')

for ( row_idx in 1:nrow(df_csvInputFiles) ) {
  # create list of data frames from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles[row_idx, filenames], sep="/")
  list_dataframes <- func_readCSVdata_to_dataframes(str_filename)

  # scramble the data frames
  df_scrambledData <- func_scrambleData(list_dataframes[[2]], vec_colnames_search_1, vec_colnames_searc

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx, filenames], df_scrambledDat
}
```

3.3 Economic sub-criteria

Walk over all input CSV files, manipulate the data, and write the results to output CSV files:

```
vec_colnames_search_1 <- c('AW01', 'AW02', 'AW03')
vec_colnames_search_2 <- c('RW01_01', 'RW02_01', 'RW03_01', 'RW04_01', 'RW05_01', 'RW06_01')
vec_colnames_out <- c('Quali_WSK', 'Quali_Bez', 'WSK_Bez')

for ( row_idx in 1:nrow(df_csvInputFiles) ) {
  # create list of data frames from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles[row_idx, filenames], sep="/")
  list_dataframes <- func_readCSVdata_to_dataframes(str_filename)
```

```
  # scramble the data frames
  df_scrambledData <- func_scrambleData(list_dataframes[[3]], vec_colnames_search_1, vec_colnames_searc

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx, filenames], df_scrambledDat
}
```

3.4 Criteria (main criteria)

Walk over all input CSV files, manipulate the data, and write the results to output CSV files:

```
vec_colnames_search_1 <- c('AK01', 'AK02', 'AK03')
vec_colnames_search_2 <- c('RK01_01', 'RK02_01', 'RK03_01', 'RK04_01', 'RK05_01', 'RK06_01')
vec_colnames_out <- c('Oeko_Soz', 'Oeko_Wirt', 'Soz_Wirt')

for ( row_idx in 1:nrow(df_csvInputFiles) ) {
  # create list of data frames from current input CSV file
  str_filename <- paste(str_input_path, df_csvInputFiles[row_idx, filenames], sep="/")
  list_dataframes <- func_readCSVdata_to_dataframes(str_filename)

  # scramble the data frames
  df_scrambledData <- func_scrambleData(list_dataframes[[4]], vec_colnames_search_1, vec_colnames_searc

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx, filenames], df_scrambledDat
}
```