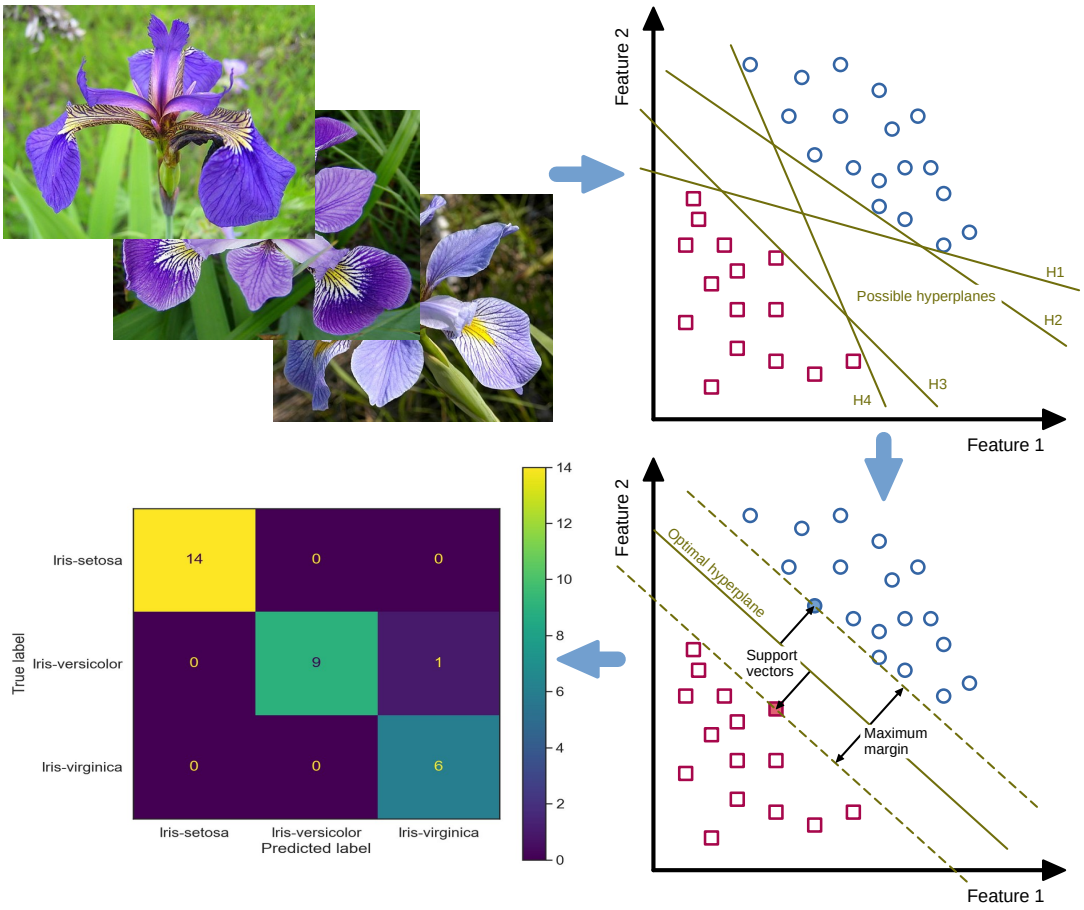
Preparing raw CSV input data from survey for analytical hierarchy process (AHP)

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:

[16]:
```
# install.packages("data.table")
require(data.table)
library(data.table)
```

2.2 Set globally used input and output folders

[2]:
```
str_input_path = "./input_data_from_survey"
str_output_path = "./output_data_manipulated"
```

2.3 Create data frame (table) handling the file names of input CSV data (raw data from survey)

[19]:
```
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"),
  descriptions = c("all target groups together",
                   "from city administrations",
                   "from non-governmental organisations",
                   "practitioners and experts")
)

#print(df_csvInputFiles)
# See first few rows
head(df_csvInputFiles)
```

A data.table: 4 × 4

file_idx <int>	keys <chr>	filenames <chr>	descriptions <chr>
1	all	rdata_all_AHP_essbare_Stadt_2022-03-18_09-53.csv	all target groups t
2	CA	rdata_CA_AHP_essbare_Stadt_2022-03-18_10-28.csv	from city administ
3	NGO	rdata_NGO_AHP_essbare_Stadt_2022-03-18_10-40.csv	from non-governm
4	PE	rdata_PE_AHP_essbare_Stadt_2022-03-18_10-41.csv	practitioners and e

3 Functions for manipulation of raw CSV input data of survey

3.1 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.

[4]:
```
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)
}
```

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

[5]:
```
func_scrambleData <- function(df_inputData, vec_colnames_search_1,␣
↪vec_colnames_search_2, vec_colnames_out) {
  # 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[1], with=FALSE])
      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] == -1) {
      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] == 2) {
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in%␣
↪vec_colnames_search_2[2], with=FALSE])
      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[3], with=FALSE])
```

```
      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] == -1) {
      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] == 2) {
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in%␣
↪vec_colnames_search_2[4], with=FALSE])
      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[5], with=FALSE])
      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] == -1) {
      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] == 2) {
      int_tmp_val <- as.integer(df_inputData[row_idx, colnames(df_inputData) %in%␣
↪vec_colnames_search_2[6], with=FALSE])
      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)
}
```

3.3 Function for writing resulting data frame to CSV file

[6]:
```
func_writeDataframe_to_CSVfile <- function(str_path, str_CSVfilename, df_dataframe,␣
↪str_filenameExtension) {
  # Split file name on second underscore, found here:
  # https://stackoverflow.com/questions/32398427/
↪r-split-a-character-string-on-the-second-underscore/32398489#32398489
```

```
  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]][2], sep="_")

  # 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)
}
```

4 Manipulate the data and store in new CSV files for each criteria

4.1 Environmental sub-criteria

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

[7]:
```
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_search_2, vec_colnames_out)

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx,␣
 ↪filenames], df_scrambledData, "env")
}
```

4.2 Social sub-criteria

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

[8]:
```
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_search_2, vec_colnames_out)

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx,␣
 ↪filenames], df_scrambledData, "soc")
}
```

4.3 Economic sub-criteria

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

[9]:
```
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_search_2, vec_colnames_out)

  # write scrambled data frames to output CSV file
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx,␣
 ↪filenames], df_scrambledData, "eco")
}
```

4.4 Criteria (main criteria)

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

[10]:
```
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_search_2, vec_colnames_out)

  # write scrambled data frames to output CSV file
```

```
  func_writeDataframe_to_CSVfile(str_output_path, df_csvInputFiles[row_idx,␣
 ↪filenames], df_scrambledData, "crit")
}
```

[ ]:

5 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).