rLSM_Script accompanying the article Introducing rLSM:An Integrated Metric Assessing Language Style Matching in Dyadic Interaction (Mueller-Frommeyer, Frommeyer, & Kauffeld, 2019)

Code written by: L. C. Mueller-Frommeyer (Technische Universitaet Braunschweig)

Date last modified: 19 August 2019

Before we start…

In order to make the contents of my paper (Mueller-Frommeyer, Frommeyer, & Kauffeld, 2019) accessible to as large a community as possible, I created an R script for the preparation and calculation of rLSM values. This script should be able to process LIWC outputs of all languages. If you encounter any problems, please feel free to contact me at any time.

Since I haven't been working with R for too long and can still remember my first rather bumpy attempts in R, I decided to use R Markdown. Above all, R Markdown offers the possibility to illustrate the functions of the script as well as necessary explanations to use it correctly. In order to use RMarkdown you have to install and use RStudio. If everything goes according to plan - and a few test runs with nice colleagues were already successful - the script should now be usable by people who have never or only briefly worked with R before.

To be able to use RMarkdown you should execute the following commands (by clicking on the green triangle in the right corner of the grey box) - it is possible that some of the packages are already installed automatically, then you don't need to execute the corresponding commands. To not execute them, simply put a "#" at the beginning of the line of code.

Please, cite the paper if you use this script for the calculation of your rLSM scores. Again, if you encounter any difficulties in processing the script or want to collaborate on a project, please feel free to contact me any time.

One more thing: A special thanks goes out to Timo Kortsch who took my first baby-steps in R with me. His knowledge and patience have helped me so much to overcome my fear of coding and get this script started.

Have fun calculating rLSM!

M.Sc. Lena Mueller-Frommeyer

Technische Universitaet Braunschweig Institue of Psychology Department for Industrial/Organizational and Social Psychology Spielmannstr. 19 & Fallersleber-Tor-Wall 23 l.mueller-frommeyer@tu-bs.de D-38106 Braunschweig

```
# clear the working environment
rm(list=ls())

#install and load necessary packages - if you not installed the following
packages yet, please remove the # to make it work
#install.packages("rmarkdown")
#install.packages("stringi")
#install.packages("dplyr")
library(rmarkdown)
library(stringi)
library(dplyr)
library(tidyr)
```

Preparation of analysis - transcripts and LIWC analysis

To be able to use the tool, the LIWC outputs **must** have a specific format. Any deviation from this format can lead to an error in the assignment of your results. For information on the transcription of natural language, please refer to the method section of the paper.

Step 1 - Transcript editing

...in Word/Texteditor

rLSM was developed by us to analyze the dynamic adaption of linguistic styles over the course of a conversation in dyadic interaction. Therefore, the transcripts have to be structured accordingly. They should contain clearly separated successive statements of the two speakers in the dyad. Statements by speakers A and B should be separated by at least 1 paragraph so that LIWC analyzes the speakers independently of each other. Within each statement, they should not contain any paragraphs so that each statement is analyzed as one segment of text. Before running your LIWC analyses, please remove all speaker identifiers from the transcript (e.g. Speaker A, Speaker B).

An example for a correctly prepared Word document can be found in document Transcript1_Example_Word in the folder “Exemplary data files”.

Analysis with LIWC

Step 1:

First start LIWC (Version 2015) and select the dictionary in the language of your transcripts under Dictionary - e.g., for the German dictionary this is “Internal German Dictionary 2015” and for the English dictionary “Internal dictionary 2015”. You can download other

dictionaries from http://www.liwc.net/dictionaries/index.php/site/login if you have purchased LIWC.

Step 2:

Since rLSM is a measure for the analysis of language style, only language style categories should be selected for further LIWC analyses. In German these are *function* (Function Words), *Pronoun* (with its sub-categories), *article* (Articles), *prep* (Prepositions) *auxverb* (Auxiliary Verbs), *adverb* (Adverbs), *conj* (Conjunctions) and *negate* (Negations). I would advice to analyze all available function word categories but only use the general function word categories (function) for your analyses - unless specifically theorized otherweise. This is a slight deviation from the paper due to a change in categories in the new German dictionary.

In addition, you can select the general descriptive categories: *WC* (Total word count), and *Dic* (Dictionary words count). You can, of course, select more of the summary dimension - the script will calculate rLSM scores for all columns starting from column five. Those will not be very meaningful for the summary dimensions.

The meaning of the respective categories can be found in the LIWC manual, in Wolf et al., (2008) or Meier et al., 2018.

You can set the categories via *Options* –> *Categories*. The selection of further categories is not necessary for the rLSM calculation.

Step 3:

In addition, under *Options* –> *Segments* you can set the number of paragraphs to differentiate between the two speakers via *Define segments based on a number of carriage returns*. In your transcripts, the amount of paragraphs should distinguish between the speakers. If you have set only one paragraph between speaker changes, you should select *1* here, if you have selected more paragraphs to identify speaker changes (e.g., *2*), you adapt it accordingly.

Step 4:

For the final LIWC analysis, you can now select your finished and uniformly prepared transcripts via *File* –> *Analyze Text* (Word/Texteditor files).

ATTENTION The R script provided here only works if you analyze each of your transcripts individually - this means you can **NOT** use the function *Analyze Text in Folder* . I am still trying to figure the less time-consuming "Analyze text in folder" option out but for now it seems like I found a bug in some of the R packages I need to use. I will update this once it is adapted. Please note, that you also cannot use the *Analyze Excel/CSV file* option as it produces different column names and therefore errors in the output. If you have prepared your transcripts as excel/csv files, please contact me and I can most probably make it work.

You can now save each LIWC output under a unique name as a **.txt file** in one folder. After, setting this folder as your working directory (which you will do in just a few seconds), the R-Script automatically accesses all LIWC outputs in this folder.

You can find a correct example of a LIWC output in the "Exeplary data files" folder (LIWC2015 Results (Transcript1_Example_Word).

TIPP Check (maybe manually) if the number of talk-turns recognized by LIWC matches the actual amount of talk-turns to avoid errors for a random selection of transcripts (ca. 10%).

Once all of the above steps have been completed, your rLSM calculation should work correctly and you can proceed with the code below.

#rLSM calculation

The following code defines the function for calculating rLSM. You don't need to change anything here. The function automatically takes into account all categories in your LIWC output file. Just activate it by clicking the green play-button on the right.

```
#create rLSM function

calculate.rLSM <- function(x) {
#prepare conversation for rLSM calculation
   ncol <- length(x[1,])
  for(i in 5:ncol)
  {
    x[length(x[1,])+1] <- c(x[,i][-1],999)
    b <- sum(ncol-4,i)
    colnames(x)[b] <- paste(colnames(x[i]),"_b", sep = "")
  }

#treatment of missing values
  for(i in 1:nrow(x)) {
    for(j in 5:ncol) {

      x[i,c(j,j+ncol-4)] <- if (x[i,j] == 0 && x[i,j+ncol-4] == 0) {999
      } else {x[i,c(j, j+ncol-4)]}
    }
  }


  for(i in 1:nrow(x)) {
    for(j in 5:ncol) {

      if (x[i,j] == 0 && x[i,j+ncol-4] > 0) {
        x[i,c(j)]<-999
        x[i,c(j+ncol-4)]<-x[i,c(j+ncol-4)]
      } else {x[i,c(j,j+ncol-4)]}
    }
  }

  x[x == 999] <- NA

  #rLSM calculations
  for(i in 5:ncol)
```

```
  {
    x[length(x[1,])+1] <- 1 - (abs(x[,i] - x[,i+ncol-4]) / (x[,i] +
x[,i+ncol-4] + 0.0001))
    colnames(x)[length(x[1,])] <- paste(colnames(x[i]),"_rLSM", sep = "")

  }

  #create a column speciyfing the speaker
  x[,length(x[1,])+1] <-rep_len(c("B","A"), length.out = nrow(x))
  colnames(x)[length(x[1,])] <- "Speaker"

  #select only the relevant columns into a new data frame
    rLSM_temp = dplyr::select(x, Filename, Segment, WC, Dic, Speaker,
ends_with("_rLSM"))
}
```

The function *calculate.rLSM* is now automatically applied to all LIWC output files in the folder you created with the following steps.

You first have to enter the path to the folder where all your LIWC outputs are located. **ATTENTION** The separators must have the following format "/".

Now insert your folder between the **""** here.

```
path <- "C:/Users/Lena/Desktop/Test/LIWC Results"
```

If you now run the following code, your computer should be busy calculating rLSM for the next seconds to minutes.

```
setwd(path)
file.names <- dir(path)
rLSM = data.frame()

for (next_conv in  1:length(file.names)) {

  rLSM_dfs <- read.csv(file.names[next_conv], sep = "\t", dec=",", header =
TRUE)

  rLSM_temp <- calculate.rLSM(rLSM_dfs)

  rLSM = dplyr::bind_rows(rLSM, rLSM_temp)

}

rLSM = rLSM %>%
  dplyr::rename(function_word_rLSM = function._rLSM)
```

To save the results to a new file, please enter your output directory between the **""** in the following chunk of code. (You have to do this as the following code will use this table to proceed the mean calculations.) Behind the last **/** you have to specify the name of the file.

```
#write table with all rLSM values
write.table(rLSM, file = "C:/Users/Lena/Desktop/Test/rLSM.txt", sep = "\t",
dec = ",", col.names = NA)
```

In the next step, we will extract temporal rLSM per speaker. Please specify the correct directory and filename.

```
#read in the saved output file
rLSM_temporal <- read.table(file = 'C:/Users/Lena/Desktop/Test/rLSM.txt',
sep = "\t", dec = ",", header = TRUE)

#choose only the relevant columns
rLSM_temporal = rLSM_temporal %>%
  dplyr::select(-X)

#extract timeline for speaker A
sub_speakerA <- subset(rLSM_temporal, Speaker=="A")
  colnames(sub_speakerA) <- paste(colnames(sub_speakerA),"_A", sep = "")
  colnames(sub_speakerA)[1] <- paste("Filename")

sub_speakerA = sub_speakerA %>%
 group_by(Filename) %>%
  mutate(join = seq_along(along.with = Speaker_A)) %>%
ungroup()

#extract timeline for speaker B
sub_speakerB <- subset(rLSM_temporal, Speaker=="B") #hier noch B einsetzen,
wenn es läuft
   colnames(sub_speakerB) <- paste(colnames(sub_speakerB),"_B", sep = "")
   colnames(sub_speakerB)[1] <- paste("Filename")

sub_speakerB = sub_speakerB %>%
 group_by(Filename) %>%
  mutate(join = seq_along(along.with = Speaker_B)) %>%
ungroup()

#join both timelines into one data set

speaker_temporal = left_join(sub_speakerA, sub_speakerB, by = c("Filename",
"join"))
```

And again, we will save this into a new data file. Please, again specify the directory and name of the file.

```
#alles wieder in eine Datentabelle schreiben
write.table(speaker_temporal, file =
"C:/Users/lena/Desktop/Test/speaker_temporal.txt",sep = "\t", na = "NA", dec
=",", col.names = NA)
```

We will now calculate a mean rLSM score for each speaker as well as each conversation.

Before you can run this chunk of code, you have to manually enter all the column names for which you want to extract summary scores. For the first set of column names, you can access the column names by looking into the data file you saved from the rLSM (I think, it's easiest if you open it in Excel). For the second set of column names, you can access the column names by looking in the data file containing the temporal rLSM per speaker.You have to separate the individual column names by comma ",". Below I already selected all function word categories available in the English dictionary.

Additionally, you have to specify the directory and file name in lines 282 and 295. Use the first table you produced in line 283 and the second table containing the timelines for each individual speaker in line 298.

```
#calculate means for each conversation and each function word category

rLSM <- read.table(file = 'C:/Users/Lena/Desktop/Test/rLSM.txt',  sep = "\t",
dec = ",", header = TRUE)

Means_conv = rLSM %>%

  group_by(Filename) %>%

  #please enter the names of the columns you want to extract summary scores
for here
 dplyr::summarise_each(funs(mean(.,na.rm =TRUE)), function_word_rLSM,
pronoun_rLSM, ppron_rLSM, i_rLSM, we_rLSM, you_rLSM, shehe_rLSM, they_rLSM,
ipron_rLSM, article_rLSM, prep_rLSM,auxverb_rLSM, adverb_rLSM, conj_rLSM,
negate_rLSM) %>%

  ungroup()

#calculate means for speaker throughout the whole conversation and each
function word category

speaker_temporal <- read.table(file =
'C:/Users/Lena/Desktop/Test/speaker_temporal.txt',  sep = "\t", dec = ",",
header = TRUE)

Means_speaker = speaker_temporal %>%

  group_by(Filename) %>%

  #please enter the names of the columns you want to extract summary scores
here
  dplyr::summarise_each(funs(mean(.,na.rm =TRUE)), function_word_rLSM_A,
pronoun_rLSM_A, ppron_rLSM_A, i_rLSM_A, we_rLSM_A, you_rLSM_A, shehe_rLSM_A,
they_rLSM_A, ipron_rLSM_A, article_rLSM_A, prep_rLSM_A,auxverb_rLSM_A,
adverb_rLSM_A, conj_rLSM_A, negate_rLSM_A, function_word_rLSM_B,
pronoun_rLSM_B, ppron_rLSM_B, i_rLSM_B, we_rLSM_B, you_rLSM_B, shehe_rLSM_B,
they_rLSM_B, ipron_rLSM_B, article_rLSM_B, prep_rLSM_B,auxverb_rLSM_B,
adverb_rLSM_B, conj_rLSM_B, negate_rLSM_B) %>%
```

```
  ungroup()

#join both data frames

rLSM_means = left_join(Means_conv, Means_speaker, by = c("Filename"))
```

We will now save the summary scores into a last individual data file. Please specify your output folder and the name of the file again.

```
write.table(rLSM_means, file = "C:/Users/Lena/Desktop/Test/rLSM_means.txt",
sep = "\t", dec = ",", col.names = NA)
```

You should now find three different outputs in your specified directory. The first one contains the rLSM scores for each talk-turn and each of your conversations. The second one contains the rLSM scores per speaker for each talk-turn of each conversatio, whereas the third one contains the summary scores per conversation and per speaker for each conversation.

I very much hope that this tutorial facilitated the application of the RScript and you were able to calculate your rLSM values.

If you have any questions or suggestions for improvements, feel free to contact me.