**Speeches in front of the 39th US Congress: The program for the name correction is based on the following steps:**

**Function 1**: **name\_correction()**

This function detects correct names and misspelled names by using the comparable dictionary. Each detected misspelled name is then compared to all correct names in the dictionary in terms of similarity score and replaced with the nearest name in the dictionary. The detected correct names are directly marked with “1” as similarity score without similarity measurement. The similarity score is calculated by using stringsim() in package stringdist. The function returns a data frame that contains input uncleaned names, nearest names, and corresponding similarity scores.

This function has three arguments: uncleaned, name\_dictionary, and method.

uncleaned: a required character vector that contains misspelled names and will be detected and corrected

name\_dictionary: a required character vector that contains correct names and is treated as comparable dictionary

method: the method for distance calculation between two strings used in stringsim(); method can be chose from c("osa", "lv", "dl", "hamming", "lcs", "qgram","cosine", "jaccard", "jw", "soundex"); the default is "lcs".

Code:

**name\_correction <- function(uncleaned, name\_dictionary, method = "lcs"){**

 **## uncleaned: an input character vector that contains misspelled names and will be detected and corrected**

 **## name\_dictionary: an input character vector that contains correct names and is treated as comparable dictionary**

 **## method: the method for distance calculation used in stringsim().**

 **# method can be chose from c("osa", "lv", "dl", "hamming", "lcs", "qgram","cosine", "jaccard", "jw", "soundex").**

 **# the default is "lcs".**

 **## the function returns a data frame that contains input uncleaned names (uncleaned), nearest names, and corresponding similarity scores.**

 **# detect misspell names**

 **index\_misspell <- which(is.na(match(uncleaned, name\_dictionary)))**

 **# detect correct names**

 **index\_corre\_spell <- which(!is.na(match(uncleaned, name\_dictionary)))**

 **# correct misspelled names**

 **nearest\_name <- character(length(uncleaned))**

 **rates<- character(length(uncleaned))**

 **for (i in index\_misspell) {**

 **rate <- rate <- stringsim(name\_dictionary, meta2[i], method = "lcs")**

 **names(rate)<- name\_dictionary**

 **# get the highest similarity ratio**

 **name\_rate <- sort(rate, decreasing = T)[1]**

 **print(name\_rate)**

 **# determine the closest corrected name and enter it into the new vector "nearest\_name"**

 **nearest\_name[i]<- names(name\_rate)**

 **# enter the highest similarity ratio into the new vector "rates"**

 **rates[i]<- name\_rate**

 **}**

 **# insert the originially correct names in their position**

 **rates[index\_corre\_spell] <- "1"**

 **nearest\_name[index\_corre\_spell] <- uncleaned[index\_corre\_spell]**

 **corrected\_name\_df<- data.frame(uncleaned, nearest\_name, rates, stringsAsFactors = F)**

 **return(corrected\_name\_df)**

**}**

**Function 2**: **name\_correction\_threshold()**

This function returns a cleaned character vector in which the corrected names having similarity score above a pointed threshold are kept and those having similarity score below the threshold are replaced with a customized replacement.

The function has four arguments: corrected\_name, similarity, cutoff, replacement.

corrected\_name: a required character vector that contains names after correction.

similarity: a required vector that has corresponding similarity score of the corrected\_name compared to the names in a comparable dictionary.

cutoff: set up a threshold value.

replacement: replace the names having similarity score below the threshold value with a customized value, default is "UNKNOWN".

Code:

**name\_correction\_threshold <- function(corrected\_name, similarity, cutoff, replacement = "UNKNOWN"){**

 **## corrected\_name: a input character vector that contains corrected names.**

 **## similarity: a input vector that has corresponding similarity score of corrected\_name compared to names in dictionary.**

 **## cutoff: set up a threshold value.**

 **## replacement: replace the names having similarity score below the cutoff value with what you want, default is "UNKNOWN".**

 **## this function returns a vector that the corrected\_names having similarity score above the cutoff are kept**

 **# and those having similarity score below the cutoff are marked as replacement.**

 **cleaned <- c()**

 **cleaned[which(similarity >= cutoff)] <- corrected\_name[which(similarity>=cutoff)]**

 **cleaned[which(similarity < cutoff)] <- replacement**

 **return(cleaned)**

**}**

**More details**

**Step 1:** Replace misspelled name with its nearest name in the master list (last\_name) by using function name\_correction(). All distinct last names of the 39th congress members and eight other terms ("president", "secretary", "speaker", "clerk", "chairman", "presiding", "members", "senators") are contained in the variable last\_name (the master list). We use the longest common substring (lcs) method of the function stringsim() when calculating the similarity index between two strings. Letters in the longest common substring of ordered letters are paired between the two strings, whereas the remaining letters that are not in the longest common substring are left unpaired. The number of unpaired letters is referred to as the lcs-distance. The similarity index between two strings is defined as one minus the ratio of the number of unpaired letters (the lcs-distance) and the number of total letters of the two strings. For example, the similarity index between “buchanan” and “buckland” is given by 1 - (6/16) = 0.625; the longest common substring of ordered letters is “bucan”, there are 6 unpaired letters (the letters h, a, n, k, l, d) and 16 letters in total. For each detected misspelled name, the program checks over all terms in the master list and calculates a similarity index for each pair. The term in the master list with highest similarity index is picked as the nearest name to the misspelled name. For detected correct names, they are directly marked with “1” as similarity score without similarity measurement and remain unchanged. Function name\_correction() generates a data frame that contains three columns. The first column “uncleaned” is the original input meta2. The second column “nearest\_names” is the nearest names of the misspelled names obtained through the similarity measurement. The last column “rates” records the corresponding similarity score between the nearest name and the name in meta2.

**library(stringdist)**

**## Load the master list of last names from outside source**

**load("C:\\Users\\ledolter\\Desktop\\Jan222020\\last\_name.RData")**

**last\_name <- tolower(last\_name)**

**last\_name**

**corrected\_name\_df <- name\_correction(meta2, last\_name)**

**corrected\_name\_df**

**colnames(corrected\_name\_df)[which(colnames(corrected\_name\_df)=="uncleaned")] <- "meta2"**

**corrected\_name\_df**

**Step 2:** Determine a cutoff value. Considering the distribution of similarity indexes, and comparing the misspelled names with their closest name under different cutoffs for the similarity index, we conclude that the matches for a similarity index above 70% will give us a reliable correction. Thus, misspelled names with a similarity index above 70% are replaced by their closest name, and misspelled names with a similarity index below 70% are marked as “UNKNOWN”.

**plot(corrected\_name\_df$rates)**

**hist(as.numeric(corrected\_name\_df$rates[which(corrected\_name\_df$rates<1)]))**

**table(corrected\_name\_df$rates[which(corrected\_name\_df$rates<=0.7&corrected\_name\_df$rates>=0.6)])**

**corrected\_name\_df[which(corrected\_name\_df$rates == 0.666666666666667), ]**

**corrected\_name\_df[which(corrected\_name\_df$rates == 0.6), ]**

**table(corrected\_name\_df$rates[which(corrected\_name\_df$rates<=0.6&corrected\_name\_df$rates>=0.55)])**

**corrected\_name\_df[which(corrected\_name\_df$rates == 0.571428571428571), ]**

**Step 3:** Apply the threshold by function name\_correction\_threshold()**.** Corrected names are stored in a new vector called meta2cleaned.

**meta2cleaned <- name\_correction\_threshold(corrected\_name\_df$nearest\_name, corrected\_name\_df$rates, 0.7)**

**meta2cleaned**

**length(meta2cleaned)**

**Step 4:** Check the new column, and perform additional checks whether the procedure has done the right thing.

**corrected\_name\_df$meta2cleaned <- meta2cleaned**

**corrected\_name\_df$meta1 <- meta1**

**# the number of incorrect (unrecognized) names left after cleaning**

**index\_misspell\_cleaned <- which(is.na(match(meta2cleaned, last\_name)))**

**length(index\_misspell\_cleaned)**

**# the number of total correct names after cleaning**

**index\_corre\_spell\_cleaned <- which(!is.na(match(meta2cleaned, last\_name)))**

**length(index\_corre\_spell\_cleaned)**

**# check if there is any empty name**

**which(nchar(meta2cleaned)==0)**

**# compare meta2 and meta2cleaned**

**length(meta1)**

**length(meta2)**

**length(meta2cleaned)**

**sort(table(meta2), decreasing = T)[1:10]**

**sort(table(meta2cleaned), decreasing = T)[1:10]**

**# meta2cleaned and meta2**

**# in both meta2cleaned and meta2**

**intersect(meta2cleaned, meta2)**

**# in meta2cleaned, but not in meta2**

**setdiff(meta2cleaned, meta2)**

**# meta2cleaned and last\_name**

**# in both meta2cleaned and last\_name**

**intersect(meta2cleaned, last\_name)**

**# in meta2cleaned, but not in last\_name**

**setdiff(meta2cleaned, last\_name)**

**# in last\_name, but not in meta2cleaned**

**setdiff(last\_name, meta2cleaned)**

**Reference:**

<https://cran.r-project.org/web/packages/stringdist/stringdist.pdf>

<https://github.com/cran/stringdist/blob/master/R/stringsim.R>