**##################################################################################**

**## Program to read the raw data (speeches of 39th Congress) from the text file xxxxxxx.txt**

**## singular/plural substitutions**

**## extraction of meta variables**

**## using machine learning approach to tidy up the last names of speakers**

**## using machine learning approach to separate senate/house speakers with the same last name**

**## results stored in file PrelimData.RData to be used in this and subsequent chapters**

**##################################################################################**

**rm(list = ls())**

**library(stringr)**

**## read from txt file**

**test=readLines("C:\\Johannes Ledolter\\2020March01Book\\combine39.txt", encoding="UFT-8")**

**data=dim(length(test))**

**for (i in 1:length(test)) {**

**data[i]=test[i]**

**}**

**data[1:10]**

**## below we execute several additional cleaning steps**

**## global replacements are tricky, and we carry them out to avoid unwanted changes**

**## we are fairly confident that substitutions were made correctly**

**for (i in 1:length(data)) {**

**txt=data[i]**

**txt=tolower(txt)**

**txt=trimws(txt, which = c("both"), whitespace = "[ \t\r\n]") ## reduces whitespace at beginning and end**

**txt=gsub("[[:space:]]{2,}"," ",txt) ## reduces 2 or more blanks to one blank**

**data[i]=txt**

**}**

**data[1:10]**

**## checks to see that odd symbols are not present**

**grep("â€",data)**

**grep("Â",data)**

**grep("&amp",data)**

**## insert a space after a comma that is not separated from an alpha character (as this indicates missing blank)**

**## would not want to insert blank if comma is followed by numbers (as this indicates numbers such as 1,000)**

**## do the same for ; and : and .**

**## above change is only done if there are 3 or more letters after the symbol. For example, h.r.g.m does not get changed into single letters**

**## there are many references to bill numbers such as "bill (H.R.No. 30)" and "bill (S.No.56)". These will be left intact.**

**## After stripping off special symbols such as . ( ) and numbers, these terms will show up as "hrno" and "sno".**

**## These can be stripped of the dtm matrix.**

**## all of that may not make much of a difference**

**## important: change " u.s. " into " unitedstates "**

**for (i in 1:length(data)) {**

**txt=data[i]**

**txt=gsub(" u[.]s[.] "," unitedstates ",txt)**

**txt=gsub(" u[.]s[.]"," unitedstates ",txt)**

**txt=gsub("[.]([[:alpha:]]{3})",". \\1" ,txt)**

**txt=gsub(",([[:alpha:]]{3})",", \\1" ,txt)**

**txt=gsub(";([[:alpha:]]{3})","; \\1" ,txt)**

**txt=gsub(":([[:alpha:]]{3})",": \\1" ,txt)**

**data[i]=txt**

**}**

**data[1:10]**

**## need to do following as we will strip out all punctuation symbols (including @ and #) at a later stage**

**for (i in 1:length(data)) {**

**txt=data[i]**

**txt=gsub("@@@in senate senate","abcdefghijklmin senate senate",txt)**

**txt=gsub("###house of representatives","abcdefghijklmhouse of representatives",txt)**

**data[i]=txt**

**}**

**data[1:10]**

**for (i in 1:length(data)) {**

**txt=data[i]**

**txt=gsub("[[:punct:]]","",txt)**

**## removes punctuation and special symbols such as % and dash (but not the symbols below)**

**## does not remove numbers**

**## below removes several odd symbols**

**txt=gsub("”","",txt)**

**txt=gsub("“","",txt)**

**txt=gsub("’","",txt)**

**txt=gsub("‘","",txt)**

**txt=gsub("—","",txt)**

**txt=gsub("–","",txt)**

**txt=gsub("[[:space:]]{2,}"," ",txt) ## reduces 2 or more blanks to one blank**

**data[i]=txt**

**data[i]=txt**

**}**

**data[1:10]**

**## check**

**grep("”",data)**

**grep("“", data)**

**grep("’",data)**

**grep("‘",data)**

**grep("—",data)**

**grep("–",data)**

**## replacing the symbols @ and # to be able to split on senate/house**

**for (i in 1:length(data)) {**

**txt=data[i]**

**txt=gsub("abcdefghijklmin senate senate","@@@in senate senate",txt)**

**txt=gsub("abcdefghijklmhouse of representatives","###house of representatives",txt)**

**data[i]=txt**

**}**

**data[1:10]**

**## fixing up word roots - combining plural and singular of selected words (instead of stemming)**

**for (i in 1:length(data)) {**

**txt=data[i]**

**txt=gsub(" "," ",txt) ## needed for replacement of two consecutive terms**

**txt=str\_pad(txt,str\_length(txt)+2,side="both") ## we do this to handle edge effects if word is at beginning (end)**

**txt=gsub(" united states "," unitedstates ",txt)**

**txt=gsub(" indians "," indian ", ignore.case = TRUE,txt)**

**txt=gsub(" wars "," war ", ignore.case = TRUE,txt)**

**txt=gsub(" oligarchs "," oligarch ", ignore.case = TRUE,txt)**

**txt=gsub(" traitors "," traitor ", ignore.case = TRUE,txt)**

**txt=gsub(" betrayals "," betrayal ", ignore.case = TRUE,txt)**

**txt=gsub(" treasons "," treason ", ignore.case = TRUE,txt)**

**txt=gsub(" lands "," land ", ignore.case = TRUE,txt)**

**txt=gsub(" slaves "," slave ", ignore.case = TRUE,txt)**

**txt=gsub(" letters "," letter ", ignore.case = TRUE,txt)**

**txt=gsub(" governments "," government ",ignore.case = TRUE,txt)**

**txt=gsub(" traders "," trader ", ignore.case = TRUE,txt)**

**txt=gsub(" troops "," troop ", ignore.case = TRUE,txt)**

**txt=gsub(" letters "," letter ", ignore.case = TRUE,txt)**

**txt=gsub(" states "," state ", ignore.case = TRUE,txt)**

**txt=gsub(" claims "," claim ", ignore.case = TRUE,txt)**

**txt=gsub(" laws "," law ", ignore.case = TRUE,txt)**

**txt=gsub(" frontiers "," frontier ",ignore.case = TRUE,txt)**

**txt=gsub(" tribes "," tribe ", ignore.case = TRUE,txt)**

**txt=gsub(" agents "," agent ", ignore.case = TRUE,txt)**

**txt=gsub(" chiefs "," chief ", ignore.case = TRUE,txt)**

**txt=gsub(" treaties "," treaty ",ignore.case = TRUE,txt)**

**txt=gsub(" nations "," nation ", ignore.case = TRUE,txt)**

**txt=gsub(" soldiers "," soldier ", ignore.case = TRUE,txt)**

**txt=gsub(" affairs "," affair ", ignore.case = TRUE,txt)**

**txt=gsub(" days "," day ",ignore.case = TRUE,txt)**

**txt=gsub(" officers "," officer ", ignore.case = TRUE,txt)**

**txt=gsub(" persons "," person ", ignore.case = TRUE,txt)**

**txt=gsub(" surveys "," survey ", ignore.case = TRUE,txt)**

**txt=gsub(" years "," year ", ignore.case = TRUE,txt)**

**txt=gsub(" townships "," town ", ignore.case = TRUE,txt)**

**txt=gsub(" township "," town ", ignore.case = TRUE,txt)**

**txt=gsub(" towns "," town ", ignore.case = TRUE,txt)**

**txt=gsub(" horses "," horse ",ignore.case = TRUE,txt)**

**txt=gsub(" republicans "," republican ", ignore.case = TRUE,txt)**

**txt=gsub(" laborers "," laborer ", ignore.case = TRUE,txt)**

**txt=gsub(" slaveholders "," slaveholder ", ignore.case = TRUE,txt)**

**txt=gsub(" capitalists "," capitalist ", ignore.case = TRUE,txt)**

**txt=gsub(" offices "," office ", ignore.case = TRUE,txt)**

**txt=gsub(" freedman "," freedmen ", ignore.case = TRUE,txt)**

**txt=gsub(" negroes "," negroe ", ignore.case = TRUE,txt)**

**txt=gsub(" negroe "," negro ", ignore.case = TRUE,txt)**

**## substitutions shown below break out words from a sequence with possible blanks**

**txt=gsub("indians"," indians ", ignore.case = TRUE,txt) ## we don't do this for "indian" because of indiana**

**txt=gsub(" indians "," indian ", ignore.case = TRUE,txt)**

**txt=gsub("negroes","negroe", ignore.case = TRUE,txt)**

**txt=gsub("negroe","negro", ignore.case = TRUE,txt)**

**txt=gsub("negro"," negro ", ignore.case = TRUE,txt)**

**txt=gsub("laborers","laborer", ignore.case = TRUE,txt)**

**txt=gsub("laborer"," laborer ", ignore.case = TRUE,txt)**

**txt=gsub("capitalists","capitalist", ignore.case = TRUE,txt)**

**txt=gsub("capitalist"," capitalist ", ignore.case = TRUE,txt)**

**txt=gsub("republicans","republican", ignore.case = TRUE,txt)**

**txt=gsub("republican"," republican ", ignore.case = TRUE,txt) ## may have an impact on republican form of government**

**txt=gsub("freedman","freedmen", ignore.case = TRUE,txt)**

**txt=gsub("freedmen"," freedmen ", ignore.case = TRUE,txt)**

**txt=gsub("governments","government",ignore.case = TRUE,txt)**

**txt=gsub("government"," government ",ignore.case = TRUE,txt)**

**txt=gsub("frontiers","frontier",ignore.case = TRUE,txt)**

**txt=gsub("frontier"," frontier ",ignore.case = TRUE,txt)**

**txt=gsub("treaties","treaty",ignore.case = TRUE,txt)**

**txt=gsub("treaty"," treaty ",ignore.case = TRUE,txt)**

**txt=gsub("slaveholders","slaveholder",ignore.case = TRUE,txt)**

**txt=gsub("slaveholder"," slaveholder ",ignore.case = TRUE,txt)**

**## need below to get rid of multiple blanks**

**txt=gsub("[[:space:]]{2,}"," ",txt) ## reduces 2 or more blanks to one blank**

**txt=trimws(txt, which = c("both"), whitespace = "[ \t\r\n]") ## reduces whitespace at beginning and end**

**data[i]=txt**

**}**

**data[1:10]**

**## checks OK**

**grep(" negro ",data)**

**grep(" freedmen ",data)**

**grep(" indiana ",data)**

**grep(" indians ",data)**

**grep(" laws ",data)**

**grep(" treaties ",data)**

**grep(" slaveholders ",data)**

**grep(" unitedstates ",data)**

**grep(" us ",data)**

**grep(" lands ",data)**

**## END: fixing up word roots - combining plural and singular of selected words**

**## CREATION OF THE META VARIABLE (META2) FOR THE SPEAKER**

**length(data) ## number of speeches**

**## cutting out speeches of length 0, 1, 2, 3 and 4 (need length 5 for getting date)**

**len1=dim(length(data))**

**for (i in 1: length(data)) {**

**txt=data[i]**

**temp=strsplit(txt," ")[[1]]**

**len1[i]=length(temp)**

**}**

**inc=len1>4 ## perhaps we should cut out more: > 4**

**table(inc)**

**data=data[inc]**

**length(data) ## number of speeches**

**data[1:10]**

**len=dim(length(data))**

**meta1=dim(length(data))**

**meta2=dim(length(data))**

**for (i in 1:length(data)) {**

**txt=data[i]**

**temp=strsplit(txt," ")[[1]]**

**len[i]=length(temp)-2**

**meta1[i]=temp[1]**

**meta2[i]=temp[2]**

**tempr=dim(len[i])**

**for (j in 1:len[i]) {**

**tempr[j]=temp[j+2]**

**}**

**data[i]=toString(tempr)**

**data[i]=gsub(",","",data[i]) ## KEY here: delete , that gets introduced with toString**

 **## if we go to dtm it will strip , so we were lucky**

**}**

**data[1:10]**

**data[10001:10010]**

**## deleting skip skip: do so before creating corpus**

**length(data)**

**length(meta1)**

**length(meta2)**

**ww=!meta1=="skip"**

**table(ww)**

**data=data[ww]**

**meta1=meta1[ww]**

**meta2=meta2[ww]**

**len=len[ww]**

**length(data)**

**length(meta1)**

**length(meta2)**

**## deleting all documents with "rev" in meta1**

**length(data)**

**length(meta1)**

**length(meta2)**

**ww=meta1=="rev"**

**table(ww)**

**data=data[!ww]**

**meta1=meta1[!ww]**

**meta2=meta2[!ww]**

**len=len[!ww]**

**length(data)**

**length(meta1)**

**length(meta2)**

**length(len)**

**len**

**sort(len)**

**hist(len)**

**boxplot(len)**

**quantile(len)**

**meta2**

**sort(meta2)**

**tt=table(meta2)**

**tt=sort(tt)**

**tt**

**meta1**

**sort(meta1)**

**tt=table(meta1)**

**tt=sort(tt)**

**tt**

**## checks**

**indicator=meta1=="rev"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="potus"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="clerk"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="secretary"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="president"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="presiding"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="speaker"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="chairman"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="members"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**indicator=meta2=="senators"**

**table(indicator)**

**meta1[indicator]**

**meta2[indicator]**

**data[indicator]**

**## GETTING ADDITIONAL META DATA SUCH AS DATE AND AFFILIATION**

**meta3=dim(length(data))**

**meta4=dim(length(data))**

**meta5=dim(length(data))**

**meta6=dim(length(data))**

**for (i in 1:length(data)) {**

**txt=data[i]**

**temp=strsplit(txt," ")[[1]]**

**meta3[i]=temp[3] ## month**

**meta4[i]=temp[4] ## day**

**meta5[i]=temp[5] ## year**

**meta6[i]=temp[1] ## senate/representatives**

**}**

**length(data)**

**length(meta1)**

**length(meta2)**

**indsen=meta1=="@@@in"**

**indhouse=meta1=="###house"**

**table(indsen)**

**table(indhouse)**

**indicator=indsen|indhouse**

**table(indicator)**

**for(i in 2:length(indicator)) {**

**if(indicator[i]==FALSE)meta3[i]=meta3[i-1]**

**if(indicator[i]==FALSE)meta4[i]=meta4[i-1]**

**if(indicator[i]==FALSE)meta5[i]=meta5[i-1]**

**if(indicator[i]==FALSE)meta6[i]=meta6[i-1]**

**}**

**indicator=meta6=="senate"**

**meta6[indicator]="senate"**

**indicator=meta6=="representatives"**

**meta6[indicator]="house"**

**meta6[1:1000]**

**## deleting senate and house**

**length(data)**

**length(meta1)**

**length(meta2)**

**length(meta3)**

**length(meta4)**

**length(meta5)**

**length(meta6)**

**indicator=indsen|indhouse**

**table(indsen)**

**table(indhouse)**

**table(indicator)**

**data=data[!indicator]**

**meta1=meta1[!indicator]**

**meta2=meta2[!indicator]**

**meta3=meta3[!indicator]**

**meta4=meta4[!indicator]**

**meta5=meta5[!indicator]**

**meta6=meta6[!indicator]**

**len=len[!indicator]**

**length(data)**

**length(meta1)**

**length(meta2)**

**length(meta3)**

**length(meta4)**

**length(meta5)**

**length(meta6)**

**length(len)**

**table(meta2)**

**table(meta3)**

**table(meta4)**

**table(meta5)**

**table(meta6)**

**table(meta5)**

**meta5f=factor(meta5)**

**table(meta3)**

**table(meta5,meta3)**

**group=dim(length(meta3))**

**for (i in 1:length(meta3)) {**

**if(meta5[i]==1865&meta3[i]=="march")group[i]=1**

**if(meta5[i]==1865&meta3[i]=="december")group[i]=2**

**if(meta5[i]==1866&meta3[i]=="january")group[i]=3**

**if(meta5[i]==1866&meta3[i]=="february")group[i]=4**

**if(meta5[i]==1866&meta3[i]=="march")group[i]=5**

**if(meta5[i]==1866&meta3[i]=="april")group[i]=6**

**if(meta5[i]==1866&meta3[i]=="may")group[i]=7**

**if(meta5[i]==1866&meta3[i]=="june")group[i]=8**

**if(meta5[i]==1866&meta3[i]=="july")group[i]=9**

**if(meta5[i]==1866&meta3[i]=="december")group[i]=10**

**if(meta5[i]==1867&meta3[i]=="january")group[i]=11**

**if(meta5[i]==1867&meta3[i]=="february")group[i]=12**

**if(meta5[i]==1867&meta3[i]=="march")group[i]=13 ## corrected missing group**

**}**

**group=factor(group)**

**table(group) ## TIME**

**################################################################################**

**## Name Correction program starts here**

**################################################################################**

**library(stringdist)**

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

**load("C:\\Johannes Ledolter\\2020March01Book\\Chapter4WEB\\last\_name.RData")**

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

**last\_name**

**### name\_correction function: correct misspelled names to the nearest name**

**### function returns data frame containing input uncleaned names (uncleaned), nearest names, and their similarity scores.**

**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").**

 **# default is "lcs".**

 **## detect misspelled 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 <- stringsim(name\_dictionary, uncleaned[i], method = method)**

 **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 originally 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)**

**}**

**### name\_correction\_threshold function: function returns vector with corrected names having similarity score above cutoff**

**### corrected names with similarity score below cutoff are labeled UNKNOWN**

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

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

 **## similarity: 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**

 **## 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)**

**}**

**# step 1: replace the misspelled name with its nearest name in the master list (name\_dictionary)**

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

**corrected\_name\_df**

**colnames(corrected\_name\_df)**

**# step 2: determine a cutoff value**

**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<=0.8&corrected\_name\_df$rates>=0.70)])**

**corrected\_name\_df[which(corrected\_name\_df$rates == 0.80), ]**

**corrected\_name\_df[which(corrected\_name\_df$rates == 0.70), ]**

**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 cutoff on the corrections and generate a cleaned vector of names**

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

**meta2cleaned**

**length(meta2cleaned)**

**# step 4: check**

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

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

**# the number of unrecognized names left after cleaning**

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

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

**table(meta2cleaned)["UNKNOWN"]**

**# the number of identified 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(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 (outside list of speakers)**

**# 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)**

**meta2 <-meta2cleaned ## replace meta2 with meta2cleaned**

**################################################################################**

**## Name Correction program ends here**

**################################################################################**

**################################################################################**

**## program (machine learning) to separate senate/house speakers with same last name**

**################################################################################**

**comparison=c("foot","wade","sumner","trumbull","collamer","foster","harlan","chandler","doolittle","clark","anthony","saulsbury","grimes","howe","mcdougall","lane","nesmith","cowan","sherman","howard","ramsey","sprague","conness","hendricks","buckalew","willey","vanwinkle","brown","riddle","stewart","nye","cragin","yates","fessenden","norton","guthrie","creswell","stockton","poland","kirkwood","edmunds","fowler","fogg","cattell","frelinghuysen")**

**## comparison includes last names of unambigous senators**

**## comparison does not include ambiguous last names of members who are in senate as well as house**

**## there are 13 last names occurring in both Senate and House. Shown below**

**## "henderson","wilson" (2 house/1 senate),"johnson","davis","harris","williams"**

**## "dixon","morgan","wright","ross","patterson","pomeroy","morrill"**

**## the program allocates S/H designation**

**## still unable to distinguish a few speakers with the same last name in either house (or senate)**

**## "lane" 2 in senate; "washburn" 2 in house; "washburne" in house**

**ncomp=length(comparison)**

**ncomp**

**meta2orig=meta2**

**## below is done so not to get into boundary problems**

**last=length(meta2)**

**for (i in 1:20) {**

**meta2[i]="ggg"**

**meta2[last+1-i]="ggg"**

**}**

**qlim=5**

**## cutoff of 5 makes sense. It is a good compromise**

**## considering that our comparison group consists of all senators.**

**## considering that we have misspelled last names and exclude senators with ambiguous last name**

**## taking 1 as the cutoff would be dangerous as then we would possibly misidentify person if the**

**## name is close to the senate/house boundary**

**##henderson**

**num=c(1:length(meta2))**

**ind=meta2=="henderson"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="henderson(H)"**

**if(sum[i]>=qlim) meta2[index]="henderson(S)"**

**}**

**##wilson**

**num=c(1:length(meta2))**

**ind=meta2=="wilson"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="wilson(H)"**

**if(sum[i]>=qlim) meta2[index]="wilson(S)"**

**}**

**##johnson**

**num=c(1:length(meta2))**

**ind=meta2=="johnson"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="johnson(H)"**

**if(sum[i]>=qlim) meta2[index]="johnson(S)"**

**}**

**##davis**

**num=c(1:length(meta2))**

**ind=meta2=="davis"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="davis(H)"**

**if(sum[i]>=qlim) meta2[index]="davis(S)"**

**}**

**##harris**

**num=c(1:length(meta2))**

**ind=meta2=="harris"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="harris(H)"**

**if(sum[i]>=qlim) meta2[index]="harris(S)"**

**}**

**##williams**

**num=c(1:length(meta2))**

**ind=meta2=="williams"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="williams(H)"**

**if(sum[i]>=qlim) meta2[index]="williams(S)"**

**}**

**##dixon**

**num=c(1:length(meta2))**

**ind=meta2=="dixon"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="dixon(H)"**

**if(sum[i]>=qlim) meta2[index]="dixon(S)"**

**}**

**##morgan**

**num=c(1:length(meta2))**

**ind=meta2=="morgan"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="morgan(H)"**

**if(sum[i]>=qlim) meta2[index]="morgan(S)"**

**}**

**##wright**

**num=c(1:length(meta2))**

**ind=meta2=="wright"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="wright(H)"**

**if(sum[i]>=qlim) meta2[index]="wright(S)"**

**}**

**##ross**

**num=c(1:length(meta2))**

**ind=meta2=="ross"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="ross(H)"**

**if(sum[i]>=qlim) meta2[index]="ross(S)"**

**}**

**##patterson**

**num=c(1:length(meta2))**

**ind=meta2=="patterson"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="patterson(H)"**

**if(sum[i]>=qlim) meta2[index]="patterson(S)"**

**}**

**##pomeroy**

**num=c(1:length(meta2))**

**ind=meta2=="pomeroy"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="pomeroy(H)"**

**if(sum[i]>=qlim) meta2[index]="pomeroy(S)"**

**}**

**##morrill**

**num=c(1:length(meta2))**

**ind=meta2=="morrill"**

**num=num[ind]**

**n1=length(num)**

**n1**

**sum=dim(n1)**

**for (i in 1:n1) {**

**nlow=num[i]-20**

**nhigh=num[i]+20**

**sum[i]=0**

**for (j in nlow:nhigh) {**

**for (k in 1:ncomp) {**

**sum[i]=sum[i]+as.numeric(meta2[j]==comparison[k])**

**}**

**}**

**}**

**hist(sum,nclass=40)**

**for (i in 1:n1) {**

**index=num[i]**

**if(sum[i]<qlim) meta2[index]="morrill(H)"**

**if(sum[i]>=qlim) meta2[index]="morrill(S)"**

**}**

**for (i in 1:20) {**

**meta2[i]=meta2orig[i]**

**meta2[last+1-i]=meta2orig[last+1-i]**

**}**

**## fix the first and last 20 by hand**

**meta2[1:20]**

**meta2[(last-19):last]**

**meta2[2]="wright(S)"**

**meta2[3]="wright(S)"**

**meta2[(last-14)]="harris(S)"**

**meta2[(last-12)]="harris(S)"**

**meta2[(last-10)]="harris(S)"**

**meta2[(last-9)]="davis(S)"**

**meta2[(last-7)]="davis(S)"**

**meta2[1:20]**

**meta2[(last-19):last]**

**sort(table(meta2))**

**################################################################################**

**## program (machine learning) to separate senate/house speakers with same last name**

**################################################################################**

**## store worksheet**

**save.image(file = "C:\\Johannes Ledolter\\2020March01Book\\Chapter4WEB\\PrelimData.RData")**

**rm(list = ls())**

**load("C:\\Johannes Ledolter\\2020March01Book\\Chapter4WEB\\PrelimData.RData")**