

# How to perform a sociolinguistic analysis in R

## – curse word use in Irish English

Martin Schweinberger

June 15, 2016

### Introduction

This post<sup>1</sup> will exemplify how to perform a simple corpus analysis with R. The example we are going to deal with is an investigation of whether males or females use more curse words in Irish English and whether curse word use is related to the age of speakers.

This example shows how you could use R (from beginning to end) to conduct a corpus analysis. This means that the entire analysis is performed with the help of R. Using R for your corpus analyses is extremely advantageous over using other software applications such as AntConc, WordSmith, Monoconc or the like as it allows you to provide a script which details each and every step of your study which in turn renders your study reliable and allows other researchers or reviewers to replicate your results. This is really a crucial point as replications is one of the cornerstones of the science process. I would argue that enabling easy replication is not only good scientific practice but even a defining criteria that allows separating science from pseudoscience. And let's be honest here for a moment: if no script is provided but traditional software applications are used, then replication is not feasible either because many decisions during data analysis are not communicated in the respective paper or because replication is simply too cumbersome! But let's begin with our example analysis.

---

<sup>1</sup>Please cite as:

Schweinberger, Martin. 2016. *How to perform a sociolinguistic analysis in R – curse word use in Irish English*. <http://www.martinschweinberger.de/blog/sociolinguanalysisr/>, date.

To answer the research questions, we need to perform at least the following steps:

1. Set up the analysis
2. Extract potential curse words;
3. Determine the gender and age of speakers;
4. Calculate the relative frequency of curse word use for each speaker;
5. Visualize the results.

Let's begin with our analysis by initializing our session and setting the parameters:

## Setting up the analysis

In a first step, we will set up our analysis by providing a short description of what our script will do and loading the necessary functions and packages.

```

1 #####
2 ### CASE STUDY: CURSE WORDS IN ICE IRELAND 1.2.2
3 #####
4 # Title: Case Study - An Analysis of Gendered Use of
5 # Curse Words in ICE Ireland 1.2.2
6 # Author: Martin Schweinberger
7 # Date: 2016-02-11
8 # Description: This script extracts selected curse words
9 # from the Irish component of the International Corpus
10 # of English (ICE) and determines whether males or
11 # females use more curse words per-1,000-words.
12 # WARNING: To use this script you need to set our own paths!
13 # Your path should be the path to the corpus on your own
14 # computer! Remember to use double backslash instead
15 # of single backslash, if you use Windows on your machine.
16 # The outpath should be the location where
17 # you would like to store the final biodata data set.
18 #####
19 # Remove all lists from the current workspace
20 rm(list=ls(all=T))
21 # load self written functions
22 source("http://martinschweinberger.de/docs/scripts/ConcR_2.3.
    r")
23 bio <- read.table("http://martinschweinberger.de/docs/data/
    BiodataIceIreland.txt", header = T, sep = "\t")
24 # setting options

```

```

25 options(stringsAsFactors = F)
26 options(scipen=999)
27 #####

```

After setting up our session in R, we will now begin with our analysis by extracting strings that represent potential swear words.

## Extracting potential curse words

After initializing the session, we can define the search patterns, i.e. the strings representing potential curse words that R will look for in the corpus and provide the concordancing function with the required arguments, e.g. the desired context size. It should be noted that searches which did not return any matches are omitted. E.g. the search string "[A|a]sshole[a-z]{0,}" which originally represented search.pattern3 did not return any matches and is thus not displayed.)

```

1 #####
  search for swear words
2 # set parameters
3 pathname <- "C:\\Corpora\\original\\ICE Ireland version
  1.2.2\\ICE-Ireland txt\\ICE spoken running txt"
4 context <- 30
5 all.pre = T
6 # define search strings
7 search.pattern1 <- c(" [A|a]ss[es|ed]{0,2} ")
8 search.pattern2 <- c(" [A|a]rse[s|d]{0,1} ")
9 search.pattern4 <- c(" [B|b]all[s]{0,1} ")
10 search.pattern5 <- c(" [B|b]astard[a-z]{0,}")
11 search.pattern6 <- c(" [B|b]ird[a-z]{0,}")
12 search.pattern7 <- c(" [B|b]itch[a-z]{0,}")
13 search.pattern8 <- c(" [B|b]loody[a-z]{0,}")
14 search.pattern9 <- c(" [B|b]ollock[a-z]{0,}")
15 search.pattern11 <- c(" [B|b]ugger[a-z]{0,}")
16 search.pattern12 <- c(" [C|c]ow[s]{0,1} ")
17 search.pattern13 <- c(" [C|c]rap[a-z]{0,}")
18 search.pattern15 <- c(" [D|d]amn[a-z]{0,}")
19 search.pattern16 <- c(" [D|d]ick[s|head]{0,}")
20 search.pattern17 <- c(" [M|m]{0,1}[other]{0,1}[F|f]uck[a-z
  ]{0,}")
21 search.pattern18 <- c(" [G|g]ay[a-z]{0,}")
22 search.pattern19 <- c(" [G|g]it[a-z]{0,}")
23 search.pattern20 <- c(" [H|h]ell[s]{0,} ")
24 search.pattern21 <- c(" [H|h]uss[a-z]{0,}")
25 search.pattern22 <- c(" [I|i]diot[a-z]{0,}")

```

```

26 search.pattern23 <- c(" [J|j]ew[s]{0,1} ")
27 search.pattern27 <- c(" [P|p]ig[s]{0,1} ")
28 search.pattern29 <- c(" [P|p]iss[a-z]{0,}" )
29 search.pattern31 <- c(" [P|p]rick[s]{0,1}" )
30 search.pattern32 <- c(" [S|s]crew[a-z]{0,}" )
31 search.pattern33 <- c(" [S|s]hag[a-z]{0,}" )
32 search.pattern34 <- c(" [S|s]hit[a-z]{0,}" )
33 search.pattern35 <- c(" [S|s]lag[a-z]{0,}" )
34 search.pattern37 <- c(" [S|s]pastic[a-z]{0,}" )
35 search.pattern38 <- c(" [T|t]art[a-z]{0,}" )
36 search.pattern41 <- c(" [T|t]wat[a-z]{0,}" )
37 search.pattern42 <- c(" [W|w]anker[a-z]{0,}" )
38 search.pattern43 <- c(" [W|w]hore[a-z]{0,}" )
39 #####

```

After defining the search patterns, we will perform the actual searches.

```

1 #####
2 # start searches
3 sw1 <- ConcR(pathname, search.pattern1, context, all.pre=T)
4 sw2 <- ConcR(pathname, search.pattern2, context, all.pre=T)
5 sw4 <- ConcR(pathname, search.pattern4, context, all.pre=T)
6 sw5 <- ConcR(pathname, search.pattern5, context, all.pre=T)
7 sw6 <- ConcR(pathname, search.pattern6, context, all.pre=T)
8 sw7 <- ConcR(pathname, search.pattern7, context, all.pre=T)
9 sw8 <- ConcR(pathname, search.pattern8, context, all.pre=T)
10 sw9 <- ConcR(pathname, search.pattern9, context, all.pre=T)
11 sw11 <- ConcR(pathname, search.pattern11, context, all.pre=T)
12 sw12 <- ConcR(pathname, search.pattern12, context, all.pre=T)
13 sw13 <- ConcR(pathname, search.pattern13, context, all.pre=T)
14 sw15 <- ConcR(pathname, search.pattern15, context, all.pre=T)
15 sw16 <- ConcR(pathname, search.pattern16, context, all.pre=T)
16 sw17 <- ConcR(pathname, search.pattern17, context, all.pre=T)
17 sw18 <- ConcR(pathname, search.pattern18, context, all.pre=T)
18 sw19 <- ConcR(pathname, search.pattern19, context, all.pre=T)
19 sw20 <- ConcR(pathname, search.pattern20, context, all.pre=T)
20 sw21 <- ConcR(pathname, search.pattern21, context, all.pre=T)
21 sw22 <- ConcR(pathname, search.pattern22, context, all.pre=T)
22 sw23 <- ConcR(pathname, search.pattern23, context, all.pre=T)
23 sw27 <- ConcR(pathname, search.pattern27, context, all.pre=T)
24 sw29 <- ConcR(pathname, search.pattern29, context, all.pre=T)
25 sw31 <- ConcR(pathname, search.pattern31, context, all.pre=T)
26 sw32 <- ConcR(pathname, search.pattern32, context, all.pre=T)
27 sw33 <- ConcR(pathname, search.pattern33, context, all.pre=T)
28 sw34 <- ConcR(pathname, search.pattern34, context, all.pre=T)
29 sw35 <- ConcR(pathname, search.pattern35, context, all.pre=T)
30 sw37 <- ConcR(pathname, search.pattern37, context, all.pre=T)
31 sw38 <- ConcR(pathname, search.pattern38, context, all.pre=T)

```

```

32 sw41 <- ConcR(pathname, search.pattern41, context, all.pre=T)
33 sw42 <- ConcR(pathname, search.pattern42, context, all.pre=T)
34 sw43 <- ConcR(pathname, search.pattern43, context, all.pre=T)
35 #####

```

We will now combine the kwic (key word in context) tables into one large table and then convert that table into a data frame and convert the potential curse word to lower case.

```

1 #####
2 # combine results
3 swire <- rbind(sw1, sw2, sw4, sw5, sw6, sw7, sw8, sw9,
4 sw11, sw12, sw13, sw15, sw16, sw17, sw18, sw19, sw20,
5 sw21, sw22, sw23, sw27, sw29, sw31, sw32, sw33, sw34,
6 sw35, sw37, sw38, sw41, sw42, sw43)
7 # convert matrix into a data frame
8 swire <- as.data.frame(swire)
9 # convert tokens to lower case
10 swire$token <- tolower(swire$token)
11 #####

```

We will now clean our data frame and extract the relevant meta information, i.e. the file name, the subfile, the speaker who uttered the potential curse word, and the text type in which the swear word appeared.

```

1 #####
2 # clean data frame
3 # remove items that are not swear words
4 swire <- swire[swire$token != "asse",]
5 swire <- swire[swire$token != "dicke",]
6 swire <- swire[swire$token != "hussein",]
7 # inspect tokens
8 #table(swire$token)
9
10 # clean file
11 swire$file <- gsub(" .*", "", swire$file)
12 # add column to data frame
13 swire <- data.frame(swire, swire$all.pre)
14 colnames(swire)[6] <- "spk"
15 # extract speaker
16 swire$spk <- gsub("$", "qwertz", swire$spk, fixed = T)
17 swire$spk <- gsub(".*qwertz", "", swire$spk)
18 swire$spk <- gsub(">.*", "", swire$spk)
19 # extract subfile
20 swire$all.pre <- as.vector(unlist(sapply(swire$all.pre,
      function(x){

```

```

21 x <- str_count(x, "<I>") } )))
22 # add id column
23 swire <- data.frame(1:nrow(swire), swire, gsub("-.*", "",
      swire$file))
24 # update column names
25 colnames(swire)[c(1, 6, 8)] <- c("id", "subfile", "txttyp")
26 # extract txttyp
27 swire$txttyp <- gsub("-.*", "", swire$txttyp)
28 #####

```

We are now in a position, where we need to code the potential curse words, i.e. we need to determine which of the potential swear word is in fact a curse word and which is not. To perform the manual coding, I usually save the data frame to my disc and then code the data in some spreadsheet software (you can also do that in the in-built spreadsheet surface in R but I never really got used to it so far) So in a next step, you may save the data frame to your disc or hard drive or wherever and perform the manual coding.

```

1 #####
2 # Store data on disc for manual coding
3 write.table(swire, "C:\\03-MyProjects\\10
      SwearWordsInIrishEnglish/swire.txt",
4 sep="\t", row.names = F)
5 #####

```

I have then coded each instance manually by assigning a 1 to actual curse words and a 0 to non-curse word uses of potential swear words. Then, I copied that coding column and now, I am going to attach it to the data frame we have created before. In a next step I remove all non-curse word uses from the data and determine how many curse words a given speaker has uttered. I then set up a table which contains the speakers who have used swear words and the number of actual curse words they have used.

```

1 #####
2 # add coding to table
3 swire$hit <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
4 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
5 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
7 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
8 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
9 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
10 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
11 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```

```

12 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
13 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
14 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
15 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
16 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
17 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
18 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
19 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
20 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
21 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
22 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
23 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
24 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
25 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
26 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
27 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
28 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
29 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
30 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
31 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
32 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
33 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
34 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
35 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
36 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
37 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1)
38 #####
39 swire <- swire[swire$hit == 1, ]
40 # create filesubfilespeakerid
41 swire$filesubfilespeakerid <- as.vector(unlist(apply(swire,
42 1, function(x){
43 x <- paste("<", x[2], "-", x[6], ":", x[7], ">", sep = "",
44 collapse = "")
45 })))
46 # determine how many swearwords a speaker has used
47 swirefsfspk <- table(swire$filesubfilespeakerid)
48 swirefsfspk <- data.frame(swirefsfspk)
49 colnames(swirefsfspk) <- c("file.speaker.id", "freq")
50 #####

```

In a next step, I combine the table we have created so far with the socio-demographic information of the speakers contained in the corpus.

## Determining the gender and age of speakers

After having extracted and cleaned our data, we will now combine the data with the socio-demographic information of the speakers. After having done so, the resulting data frame is cleaned, e.g. all speakers whose gender or age is not known are removed from the data and speakers who uttered less than 100 words are removed to avoid creating outliers. In addition, I remove all files that do not represent private dialogue as such files hard contain any curse words.

```

1 #####
2 # clean bio data: extract only relevent columns
3 bio <- data.frame(bio$text.id, bio$subfile, bio$spk.ref,
4   bio$sex, bio$age, bio$word.count)
5 colnames(bio) <- c("file", "subfile", "spk", "sex", "age", "
6   wc")
7 # create filesubfilespeakerid
8 bio$file.speaker.id <- as.vector(unlist(apply(bio, 1,
9   function(x){
10 x <- paste("<", x[1], "-", x[2], ":", x[3], ">", sep = "",
11   collapse = "")
12 })))
13 #####
14 # combine frequencies and biodata
15 swire <- join(bio, swirefsfspk, by = c("file.speaker.id"),
16   type = "left")
17 # clean data
18 swire <- swire[is.na(swire$sex) == F, ]
19 swire <- swire[is.na(swire$age) == F, ]
20 swire$freq <- sapply(swire$freq, function(x){
21 ifelse(is.na(x) == T, 0, x) } )
22 swire <- swire[swire$wc != 0, ]
23 #####

```

After having combined the data with the speaker information, we will now calculate the relative frequency of swear word use for each speaker.

## Calculating relative frequencies

We now calculate the normalized or relative frequencies of swear words for each speaker, so that we can compare speakers and cohorts although they differ with respect to the absolute frequency of words they have produced. In our example, we will calculate per-1,00-word frequencies.



```

1 #####
2 # calculate per-1,000-words frequency
3 swire$ptw <- round(swire$freq/swire$wc*1000)
4 # select only files with headers S1A
5 swire$txtttype <- gsub("<", "", swire$file.speaker.id)
6 swire$txtttype <- gsub("-.*", "", swire$txtttype)
7 swire <- swire[swire$txtttype == "S1A", ]
8 swire <- swire[swire$wc > 99, ]
9
10 #####

```

In a final step, we visualize the results of the analysis in a neat and easy to understand fashion.

## Visualizing the results

The last section exemplifies, how results could be visualized.

```

1 #####
2 # change graphic parameters
3 par(mfrow = c(1,2))
4 # perform preliminary statz on age
5 wilcox.test(swire$ptw ~ swire$sex)
6 # plot data (sex)
7 boxplot(swire$ptw ~ swire$sex, ylim = c(-5, 20), col = c("
8   lightpink", "lightblue"),
9   notch = T, ylab = "Relative Frequency (per 1,000 words)",
10  xlab = "Sex")
11 grid()
12 text(1:2, tapply(swire$ptw, swire$sex, mean), "+")
13 text(1:2, c(-3.5, -3.5), paste("mean=\n", round(tapply(
14   swire$ptw, swire$sex, mean), 2), sep = ""))
15 # include statz in graph
16 text(1, 20, "Wilcox Test")
17 text(1, 18, paste("W=", wilcox.test(swire$ptw ~ swire$sex)
18   [1], sep = ""))
19 text(1, 16, paste("p=", round(wilcox.test(swire$ptw ~
20   swire$sex)[[3]], 4), sep = ""))
21
22 # plot data (sex)
23 combined <- round(tapply(swire$ptw, swire$age, mean), 3)
24 m <- swire[swire$sex == "male", ]
25 male <- round(tapply(m$ptw, m$age, mean), 3)
26 f <- swire[swire$sex == "female", ]
27 female <- round(tapply(f$ptw, f$age, mean), 3)
28 # remove first item

```

```

24 combined <- combined[2:length(combined)]
25 male <- male[2:length(male)]
26 female <- female[2:length(female)]
27 # start plotting
28 plot(female, type = "o", lwd = 1, lty = 1, pch = 19, ylim = c
      (0, 10), xlab = "Age",
29 ylab = "Relative Frequency (per 1,000 words)", axes = F, cex
      = 1, col = "red")
30 # add a line for the data points in vector male (pch = 0: use
      empty squares as point characters)
31 lines(male, type = "o", lwd = 1, lty = 1, pch = 0, cex = 1,
      col = "blue")
32 # add a line for the data points in vector combined
33 lines(combined, type = "o", lwd = 1, lty = 3, pch = 4, cex =
      1, col = "darkgrey")
34 # add x-axes with specified labels at specified intervals
35 axis(1, at = 0:5, lab = c("", "19-25", "26-33", "34-41", "
      42-49", "50+"))
36 # add y-axes with specified labels at specified intervals
37 axis(2, at = seq(0, 10, 2), las = 1, lab = seq(0, 10, 2))
38 # create a legend
39 # define vector with linetypes
40 linetype <- c(1, 1, 3)
41 # define vector with point characters
42 plotchar <- c(19, 0, 4)
43 # define vector with colors
44 plotcol <- c("red", "blue", "darkgrey")
45 # set up legend
46 legend("topright", inset = .05, c("female", "male", "all"),
47 horiz = F, pch = plotchar, lty = linetype, col = plotcol)
48 # create a box around the plot
49 box()
50 # create a grid in the plot
51 grid()
52 # restore graphical parameters
53 par(mfrow = c(1,1))
54 #####
55 # THE END
56 #####

```

The resulting visualization is displayed below.

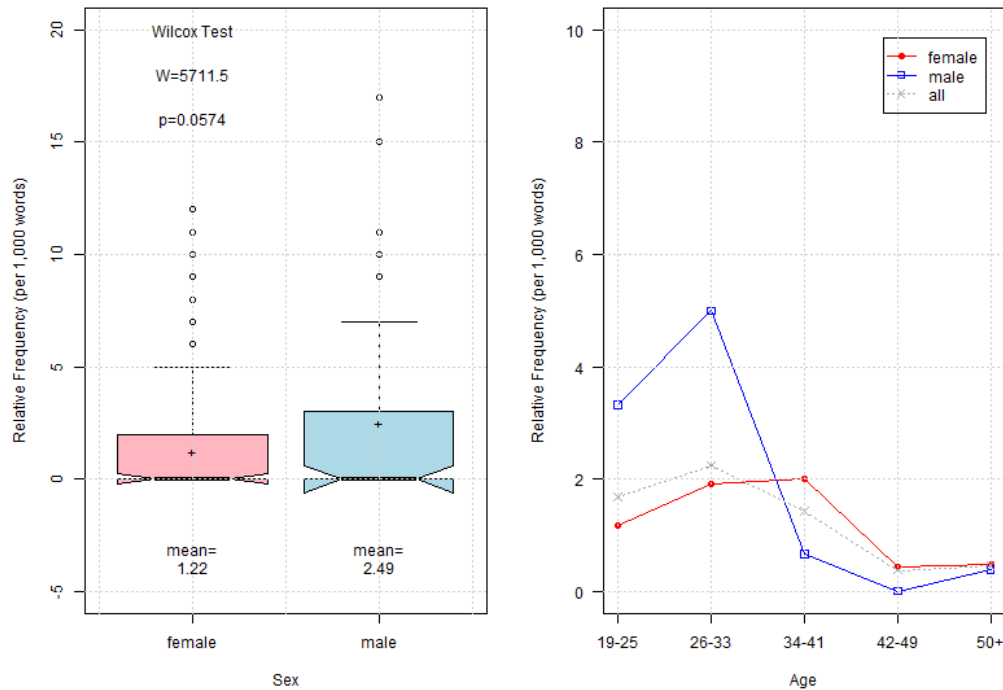


Figure 1: Overview of swear word use in Irish English based on the private dialogue files of ICE Ireland 1.2.2 by age and gender.

The panels in Figure 1 strongly suggest that while males and females differ with respect to their use of swear words among younger speakers, this is neither true for speakers above an age of 34 nor for the males and females in general (left panel). Indeed, a Wilcoxon test (see left panel) confirms that males and females do not differ significantly if the age of speakers is neglected. In contrast, the right panel strongly suggests an interaction between swear word use and the age of speakers on the one hand and their gender on the other. A more in-depth analysis of which predictors affect the use of swear words is, however, beyond the scope of this tutorial.

I hoped this short tutorial might help you perform your own corpus analyses with R.