Concordancing with R-V2

Rodrigo Esteves de Lima-Lopes
State University of Campinas
rll307@unicamp.br

Contents

1 Introduction

Today we are going to improve our skills in making KWIC using R. We will start from our last tutorial and built some new skills in the top of it.

This tutorial is based on:

> Jockers, Matthew Lee. 2014. *Text Analysis with R for Students of Literature: Quantitative Methods in the Humanities and Social Sciences.* Cham: Springer.

The ebook is available for UNICAMP students. To download it, you will need to access the university's library using our VPN system.

In this tutorial, we will continue to do some concordancing, using loops and custom created functions.

The data here is the same Jokers (2014) uses, but we might use a different dataset in classroom.

1.1 Before you start

1. Set a working directory for this tutorial
   - It is essential you know where files are going to be
2. Download the folder `data`, it is available in Module_3 root directory and keep it in your working directory.
3. Download the file `00_functions.R`, it is available in Module_3 root directory
4. Place both files in the root directory

2 Loading previous defined functions

Our fist step will be loading two previously defined functions we saved in a previous file. That is a very easy way of recycling the previous written function. `source()` is a command that reads the content of a *R* file and executes it, it is a way of importing data and functions into *R*. It means that we can save some functions we create in files and importing them when we need, no need for writing or cut/paste it.

```
source('00_functions.R')
```

3 What do I want my KWIC to do?

We would like a KWIC function that would allow us to:

1. Display the name of the files available for search
2. Allow me to choose one of these files
3. Ask me how far it needs to go (left of right)
4. Ask which word it should search
5. Display the results at console

So, in a nutshell my function would:

```
my.function <- function{
  Display the name of the files available for search
  Allow me to choose one of these files
  Ask me how far it needs to go (left of right)
  Ask which word it should search
  Display the results at console
}
```

4 Building our 2nd KWIC funtion

Now we can build our second KWIC function. We will some of the knowledge we got from our previous tutorial. My first step will be name my function and add arguments to it:

```
KWIC.2 <- function(my.corpus.l)
```

Our function's name is `KIWC.2` and it will take only a single argument, `my,corpus.l`. This file will be created using the function `corpus.list` which we created and recycled earlier. But never mind, we will review this step before applying such functions.

Our next step is to add the function `file.listing` for displaying the name of the files in my corpus. We also will define this function locally and we will discuss it further ahead.

```
KWIC.2 <- function(my.corpus.l){
  file.listing(names(my.corpus.l)) # Displays the names of the files in my list
}
```

Now we will make our function interactive. We will ask for three pieces of information in order to get concordances according to our research needs:

```
KWIC.2 <- function(my.corpus.l){
  file.listing(names(my.corpus.l)) # Displays the names of the files in my list
  file.id <- as.numeric( #Two lines to ask the file you want to examine
    readline("Which file would you like to examine? Enter a number: \n"))
  context <- as.numeric(#Two lines to ask the context you want to examine(left or right)
    readline("How much context do you want to see? Enter a number: \n"))
  keyword <- tolower((readline("Enter a keyword: \n"))) #Which key word do you want to search for?
}
```

Now we will actually search the word we are interested in. Our approach will be to search inside the list we have identified as our source:

```
KWIC.2 <- function(my.corpus.l){
  file.listing(names(my.corpus.l)) # Displays the names of the files in my list
  file.id <- as.numeric( #Two lines to ask the file you want to examine
    readline("Which file would you like to examine? Enter a number: \n"))
  context <- as.numeric(#Two lines to ask the context you want to examine(left or right)
    readline("How much context do you want to see? Enter a number: \n"))
  keyword <- tolower((readline("Enter a keyword: \n"))) #Which key word do you want to search for?
  hits.v <- which(my.corpus.l[[file.id]] == keyword) #searching the keyword
}
```

`hits.v <- which(my.corpus.l[[file.id]] == keyword)` is responsible for searching the positions our node is inside the vector, which is inside a list, we have decided as our corpus.

The next step is to make a loop. This loop will use the number we defined in the context as a criterion for getting the occurrences to the left of our node. Since some KWIC might be at the begging of a sentence, it is important to add and extra condition, we will consider left collocates only if our node is not is first word in a sentence. The next line prints the occurrences on the right.

```
KWIC.2 <- function(my.corpus.l){
  file.listing(names(my.corpus.l)) # Displays the names of the files in my list
  file.id <- as.numeric( #Two lines to ask the file you want to examine
    readline("Which file would you like to examine? Enter a number: \n"))
  context <- as.numeric(#Two lines to ask the context you want to examine(left or right)
    readline("How much context do you want to see? Enter a number: \n"))
  keyword <- tolower((readline("Enter a keyword: \n"))) #Which key word do you want to search for?
  hits.v <- which(my.corpus.l[[file.id]] == keyword) #searching the keyword
  if(length(hits.v)>0){ ## loop for printing occurrences to the left
    result <- NULL
    for(h in 1:length(hits.v)){
      start <- hits.v[h]-context
      if(start < 1){
        start <- 1
      }
      end <- hits.v[h]+context ## For printing occurrences to the
}
```

The next piece of code organises the concordance in the visual way we are used to: context + node + context. The result is, then, printed back to us.

```
KIWC.2 <- function(my.corpus.l){
  file.listing(names(my.corpus.l)) # Displays the names of the files in my list
  file.id <- as.numeric( #Two lines to ask the file you want to examine
    readline("Which file would you like to examine? Enter a number: \n"))
  context <- as.numeric(#Two lines to ask the context you want to examine(left or right)
    readline("How much context do you want to see? Enter a number: \n"))
  keyword <- tolower((readline("Enter a keyword: \n"))) #Which key word do you want to search for?
  hits.v <- which(my.corpus.l[[file.id]] == keyword) #searching the keyword
  if(length(hits.v)>0){ ## loop for printing occurrences to the left
    result <- NULL
    for(h in 1:length(hits.v)){
      start <- hits.v[h]-context
      if(start < 1){
        start <- 1
      }
      end <- hits.v[h]+context ## For printing occurrences to the
      cat(my.corpus.l[[file.id]][start:end], "\n") # cat the concordancing line
      myrow <- cbind(hits.v[h],#organising
                     paste(my.corpus.l[[file.id]][start:(hits.v[h]-1)], #start
                           collapse=" "),
                     paste(my.corpus.l[[file.id]][hits.v[h]], #hit
                           collapse=" "),
                     paste(my.corpus.l[[file.id]][(hits.v[h]+1):end],collapse=" ")) #end
      result <- rbind(result,myrow)
    }
  }
  else {cat("YOUR KEYWORD WAS NOT FOUND\n")}
  colnames(result) <- c("position", "left", "keyword", "right")
  return(result)
}
```

5 Concordancing

Now let make our concordance step by step:

1. Source the file with the R functions we saved `source('00_functions.R')`
2. Run the last code (the one just above to save our function in the memory)
3. Set your work directory accordingly and create a directory `data` in it and copy the text files provided in the course to it.
4. Run the command `input.dir <- "data"` to tell R where the data is.
5. Run the command `my.files <- dir(input.dir, "\\.txt$")` to tell R which are our files
6. Run the command `my.corpus <- corpus.list(my.files,input.dir)` to create a list with two vectors of words
7. Run the command `my.results<-KIWC.2(my.corpus)` to get the concordances in a variable

```
source('00_functions.R')
input.dir <- "data"
my.files <- dir(input.dir, "\\.txt$")
my.corpus <- corpus.list(my.files,input.dir)
my.results<- KIWC.2(my.corpus)
```

Voilà!