# Concordancing with R-V1

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

## Contents



# 1 Introduction

Today we are going to learn how to make concordances (KWIC), using R and no special package. This will take us two lessons, and possibly one class or more. Please, follow the instructions in this documentation if you have any doubts.

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, some new themes will be introduced:

1. Functions
   - A set of commands that are executed at once and applied in a file
2. Loops
   - A set of functions that are applied recursively.

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

## 1.1 Before you start

Download the folder `data`, it is available in Module_3 root directory.

# 2 Concordancing

## 2.1 Step 1 reading files

Our fist step is to set up our work directory and a couple of variables.

```
setwd("my directory")
input.dir <- "data"
my.files <- dir(input.dir, "\\.txt$")
```

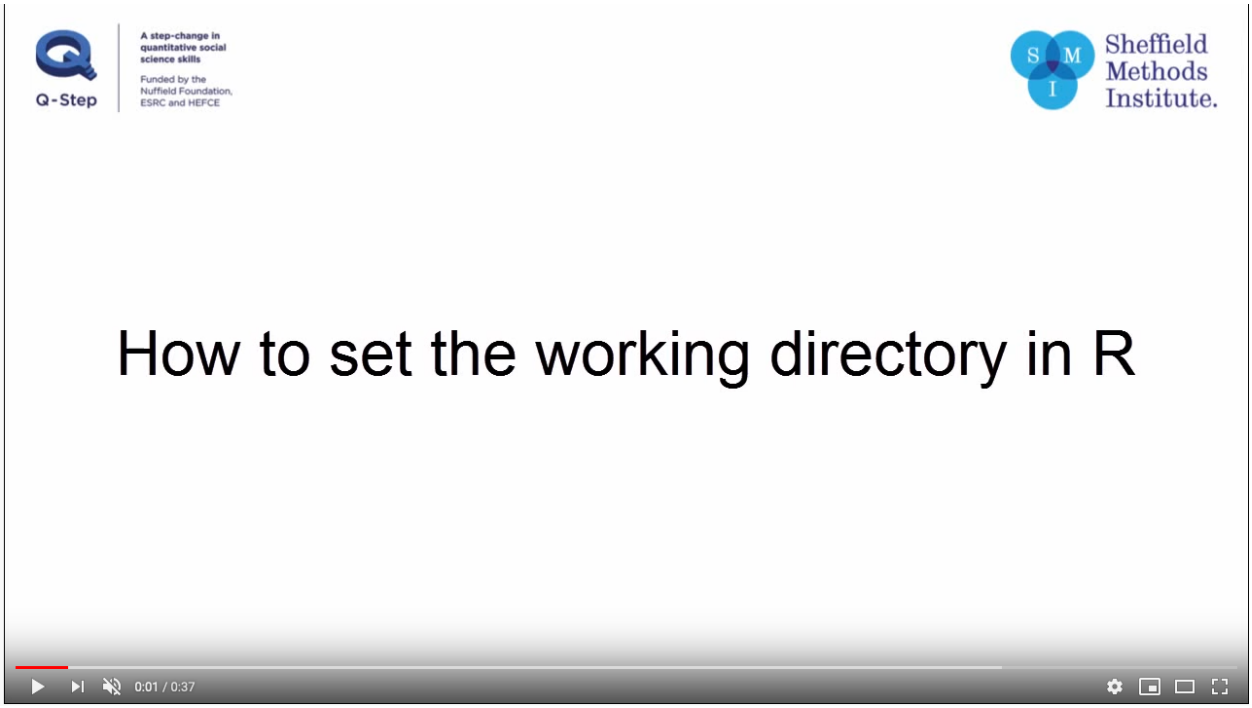
Two things are important in the commands above

### 2.1.1 setwd() and data file directory

`setwd()` is a command line instruction. It makes R point to a specific directory "*my directory*" where it reads information from and prints information to. It will be necessary that know precisely the address (path) of the directory in your computer and also it is imperative that you write it according to the Operation system notation you are using.

**If you do not know how to write it, used the gear icon at Rstudio window to do so**. There you will see the option to set your work directory. Other option is to use the menu to change your working directory under Session > Set Working Directory > Choose Directory.

A good example how todo so is presented by this video:




`input.dir <- "data"` is an ordinary command that will make the directory `data` to be represented in one variable, while `my.files <- dir(input.dir, "\\.txt$")` will read all *txt* files in "data" as a variable.

Note that we will be reading the files from inside our computer.

## 2.2 Creating our custom functions

Functions are chunks of codes that combine lines of code (redundant but correct) in order to make things faster and more automated. **R** comes with many pre-coded functions and, when we load a new package, it also brings a whole lot of new ones. However, sometimes, we either need some customised functions to perform a very specific function.

For example, there is just one package that makes KWIC available in R Quanteda. If this package's development is shut down for any reason, we might have to get a workaround.

In the next tutorials we will see some more details about functions, now let us go to our concordancer.

## 2.3 Creating the base list()

Our fist step will be creating a function to make a `list()` out of my corpus. This list will be used for concordancing.

```
corpus.list <- function(my.files, input.dir){
  "my code"
}
```

The chunk above shows us the basics of a function: `1. The name of my function is`corpus.list`1.` funtion()`defines that the following code will perform my function we named corpus.list` - the code between {...} will perform the function itself - The function will carry two arguments - `my.files` is the vector of files we created just above - `input.dir` is a vector with the name of the file's directory we created just above

So, let us star filling in the code:

```
corpus.list <- function(my.files, input.dir){
  #Creates an empty list
  corpus.list <- list()
```

The command `corpus.list <- list()` creates a empty list I will "fill in" with my data.

```
corpus.list <- function(my.files, input.dir){
  #Creates an empty list
  corpus.list <- list()
  # loop over the files
  for(i in 1:length(my.files)){
    # read the file in,  we need to know the input directory
    text.v <- scan(paste(input.dir, my.files[i], sep="/"),
                   what="character", sep="\n")
```

Then we created a loop. A loo is command that does an action recursively. We will discuss a it later, but for the time, it is important to notice:

`for` is a loop that applies a command for each element in a vector:

- So for each item `i in` the length of my vector "my files", it will paste the input directory followed my the name of the file. It will treat those names as `character` and separate them by a line feed `\n`.
- Finally it will be stored in a vector `text.v`

```
corpus.list <- function(my.files, input.dir){
  #Creates an empty list
  corpus.list <- list()
  # loop over the files
  for(i in 1:length(my.files)){
    # read the file in,  we need to know the input directory
    text.v <- scan(paste(input.dir, my.files[i], sep="/"),
                   what="character", sep="\n")
    #convert to single string
    text.v <- paste(text.v, collapse=" ")
    #lowercase and split on non-word characters
    text.lower.v <- tolower(text.v)
    text.words.v <- strsplit(text.lower.v, "\\W")
    text.words.v <- unlist(text.words.v)
    #remove the blanks
    text.words.v <- text.words.v[which(text.words.v!="")]
```

The code we added here makes a number of changes in our texts:

1. `text.v <- paste(text.v, collapse=" ")` converts it to a single character string
2. `text.lower.v <- tolower(text.v)` lowers all cases
3. `text.words.v <- strsplit(text.lower.v, "\\W")` breaks all strings by line and delete "non-words" characters
4. `text.words.v <- unlist(text.words.v)` unlists, so we do not have a list inside a list.
5. `text.words.v <- text.words.v[which(text.words.v!="")]` removes all lines containing only spaces

```
corpus.list <- function(my.files, input.dir){
  #Creates an empty list
  corpus.list <- list()
  # loop over the files
  for(i in 1:length(my.files)){
    # read the file in,  we need to know the input directory
    text.v <- scan(paste(input.dir, my.files[i], sep="/"),
                   what="character", sep="\n")
    #convert to single string
    text.v <- paste(text.v, collapse=" ")
    #lowercase and split on non-word characters
    text.lower.v <- tolower(text.v)
    text.words.v <- strsplit(text.lower.v, "\\W")
    text.words.v <- unlist(text.words.v)
    #remove the blanks
    text.words.v <- text.words.v[which(text.words.v!="")]
    #use the index id from the my.files vector as the "name" in the list
    corpus.list[[my.files[i]]] <- text.words.v
  }
  return(corpus.list)
}
```

Now changes now are:

1. `corpus.list[[my.files[i]]] <- text.words.v` used the my.files index to name the elements in my list accordingly
2. `return(corpus.list)` returns my final list as the result

Now all I have to do is to run it:

```
my.corpus.list <- corpus.list(my.files, input.dir)
```

The results are:

```
str(my.corpus.list)
```

```
## List of 2
##  $ austen.txt  : chr [1:123969] "the" "project" "gutenberg" "ebook" ...
##  $ melville.txt: chr [1:221908] "the" "project" "gutenberg" "ebook" ...
```

## 2.4 Observing postions

Now we can observe in which positions a given word is in our novels:

```
e.positions.sense <- which(my.corpus.list[[1]][]=="women")
head(e.positions.sense,15)
```

```
##  [1]   3282  71318  71404  74581  74808  79460  82736 106257 114776 117774
## [11] 119135
```

```
d.positions.moby <- which(my.corpus.list[[2]][]=="dog")
head(d.positions.moby,15)
```

```
##  [1]  14555  16376  27192  51031  51107  51565  73930 107614 107700 137047
## [11] 137077 147004 167296 170197 170577
```

## 2.5 Let us make a concordance

For concordancing, we will be using the vector positions we just created and a new loop:

```
# setting the context
context <- 5
for(i in 1:length(e.positions.sense)){
start <- e.positions.sense[i]-context
end <- e.positions.sense[i]+context
cat(my.corpus.list[[1]][start:end], "\n")
}
```

```
## what on earth can four women want for more than that
## daughters were such kind of women as fanny would like to
## one of the most charming women in the world lady middleton
## sister they are both delightful women indeed i wonder i should
## me they are such charming women i am sure if ever
## much pleased with any young women in her life as she
## she had asked these young women to her house merely because
## herself one of the happiest women in the world elinor could
## seen so little of other women that i could make no
## was the most unfortunate of women poor fanny had suffered agonies
## of the most fortunate young women in the world as it
```

If we beak this looping:

1. `context <- 5` sets the KWIC to be five words long
2. `for(i in 1:length(e.positions.sense))` tells that from one up to the length of `e.positions.sense` the computer should apply this loop
3. `start <- e.positions.sense[i]-context` the start of my line is the occurrence of women, and it should consider five words to the left
4. `end <- e.positions.sense[i]+context` the end of my line is the occurrence of women, and it should consider five words to the right
5. `cat(my.corpus.list[[1]][start:end], "\n")` searches in the fist element of my list and prints it in the console

Now for Moby Dick, the same loop

```
# setting the context
context <- 5
for(i in 1:length(e.positions.sense)){
start <- e.positions.sense[i]-context
end <- e.positions.sense[i]+context
cat(my.corpus.list[[1]][start:end], "\n")
}
```

```
## what on earth can four women want for more than that
## daughters were such kind of women as fanny would like to
## one of the most charming women in the world lady middleton
## sister they are both delightful women indeed i wonder i should
## me they are such charming women i am sure if ever
```

```
## much pleased with any young women in her life as she
## she had asked these young women to her house merely because
## herself one of the happiest women in the world elinor could
## seen so little of other women that i could make no
## was the most unfortunate of women poor fanny had suffered agonies
## of the most fortunate young women in the world as it
```

However, it is not aesthetically pleasing, so I will modify the code a little:

```
for(i in 1:length(d.positions.moby)){
  start <- d.positions.moby[i]-context
  end <- d.positions.moby[i]+context
  before <- my.corpus.list[[2]][start:(start+context-1)]
  after <- my.corpus.list[[2]][(start+context+1):end]
  keyword <- my.corpus.list[[2]][start+context]
  cat("----------------------", i, "----------------------", "\n")
  cat(before,"[",keyword, "]", after, "\n")
}
```

```
## ---------------------- 1 ----------------------
## all over like a newfoundland [ dog ] just from the water and
## ---------------------- 2 ----------------------
## a fellow that in the [ dog ] days will mow his two
## ---------------------- 3 ----------------------
## was seen swimming like a [ dog ] throwing his long arms straight
## ---------------------- 4 ----------------------
## filling one at last down [ dog ] and kennel starting at the
## ---------------------- 5 ----------------------
## not tamely be called a [ dog ] sir then be called ten
## ---------------------- 6 ----------------------
## t he call me a [ dog ] blazes he called me ten
## ---------------------- 7 ----------------------
## sacrifice of the sacred white [ dog ] was by far the holiest
## ---------------------- 8 ----------------------
## life that lives in a [ dog ] or a horse indeed in
## ---------------------- 9 ----------------------
## the sagacious kindness of the [ dog ] the accursed shark alone can
## ---------------------- 10 ----------------------
## boats the ungracious and ungrateful [ dog ] cried starbuck he mocks and
## ---------------------- 11 ----------------------
## intense whisper give way greyhounds [ dog ] to it i tell ye
## ---------------------- 12 ----------------------
## to the whale that a [ dog ] does to the elephant nevertheless
## ---------------------- 13 ----------------------
## aries or the ram lecherous [ dog ] he begets us then taurus
## ---------------------- 14 ----------------------
## is dr bunger bunger you [ dog ] laugh out why don t
## ---------------------- 15 ----------------------
## to die in pickle you [ dog ] you should be preserved to
## ---------------------- 16 ----------------------
## round ahab and like a [ dog ] strangely snuffing this man s
## ---------------------- 17 ----------------------
## lad five feet high hang [ dog ] look and cowardly jumped from
## ---------------------- 18 ----------------------
## as a sagacious ship s [ dog ] will in drawing nigh to
```

```
## ---------------------- 19 ----------------------
## the compass and then the [ dog ] vane and then ascertaining the
```

Some little changes make all the difference:

1. the line `cat("---i---","\n")` inserted a line between each occurrence
2. `cat(before,"[",keyword, "]", after, "\n")` made the KWIC to be displayed between squared brackets