Reading and Writing an Excel File in R

This is part of a series on getting started with patent analysis in R using [RStudio](http://www.rstudio.com). If you do not have a copy of RStudio follow the simple instructions for installing on your platform [here](http://www.rstudio.com/products/rstudio/download/). There are lots of resources on the site to help you get started including [online learning](http://www.rstudio.com/resources/training/online-learning/), [videos](http://www.rstudio.com/resources/webinars/), and [cheatsheets](http://www.rstudio.com/resources/cheatsheets/). The excellent [R-Bloggers site](http://www.r-bloggers.com) will demonstrate why it is worth investing time in R when working with patent data.

A lot of patent analysts will probably use Excel as part of their daily routine. However, when it comes to developing a patent analysis workflow that involves R the [CRAN Project](http://cran.r-project.org/doc/manuals/r-release/R-data.html#Reading-Excel-spreadsheets) has the following to say about importing Excel files into R.

"The first piece of advice is to avoid doing so if possible! If you have access to Excel, export the data you want from Excel in tab-delimited or comma-separated form, and use read.delim or read.csv to import it into R. (You may need to use read.delim2 or read.csv2 in a locale that uses comma as the decimal point.)."

This is very sound advice. The best option when dealing with Excel is generally to use save as to save the file as a .csv and then import it into R. However, there are a number of ways of reading an Excel file into R. We will deal with two of them in this walk through focusing on the patent datasets in our [open access patent datasets folder](https://drive.google.com/open?id=0B4piiKOCkRPDNThTWU1QQVYyRnM&authuser=0). Download the GitHub .zip file [here](https://github.com/poldham/opensource-patent-analytics/blob/master/2_datasets/datasets.zip?raw=true). Feel free to use your own dataset.

One challenge with R and Excel files is that no one package seems to do everything that you want. In particular, reading from URLs is a bit of a minefield particularly on secure connections (https:). If this walk through doesn't meet your needs then try this R-bloggers [overview](http://www.r-bloggers.com/read-excel-files-from-r/) on the range of available packages. The [R-bloggers excel topic listing](http://www.r-bloggers.com/search/excel) also has lots of useful articles covering working with Excel in more depth than this short article. To find additional help try [stackoverflow](http://stackoverflow.com/questions/tagged/r).

We will focus on:

1. Using the [xlsx](http://www.r-bloggers.com/importexport-data-to-and-from-xlsx-files/) package.
2. Testing the new [readxl](http://blog.rstudio.org/2015/04/15/readxl-0-1-0/) package.

To read an Excel file into R first install the package or tick the box in the Packages list to load it or load the library.

install.packages("xlsx")

Load the library

library(xlsx)

## Loading required package: rJava  
## Loading required package: xlsxjars

You can use your own local excel file but we will use the file [wipotrends](https://drive.google.com/file/d/0B4piiKOCkRPDNWhrdGxXc0YwTk0/view?usp=sharing) in the [patent dataset folder](https://drive.google.com/open?id=0B4piiKOCkRPDNWhrdGxXc0YwTk0&authuser=0) for this example. Other test Excel datasets in the folder are [ewaste](https://drive.google.com/open?id=0B4piiKOCkRPDZGZ4dlJsVEN4TEk&authuser=0) and [solarcooking](https://drive.google.com/open?id=0B4piiKOCkRPDMUVSaFJtdXlOX28&authuser=0) if you would like to experiment later. Download the file and save it to your computer. Then copy the local file path.

## Reading a local file

We will use a file called [wipotrends](https://drive.google.com/open?id=0B4piiKOCkRPDNWhrdGxXc0YwTk0&authuser=0)

Let's open the file up to inspect it briefly. We will see that it contains one worksheet and that the column headings begin at row 5. Normally we would remove the introductory text so that the columns are in row 1. In this case we will load it into R as is. We will use the read.xlxs function and specify arguments to tell R where to look for and handle the data.

wipotrends <- read.xlsx("/Users/pauloldham17inch/Desktop/open\_source\_master/2\_datasets/wipo/wipotrends.xlsx", sheetIndex = 1, startRow = 5, endRow = 23, as.data.frame = TRUE, header=TRUE)

We have specified a number of arguments for the read.xlsx function in the pattern read.xlsx("filename", arguments). Let's call up the full range of arguments for the function.

?read.xlsx()

The available arguments for the function are:

read.xlsx(file, sheetIndex, sheetName=NULL, rowIndex=NULL, startRow=NULL, endRow=NULL, colIndex=NULL, as.data.frame=TRUE, header=TRUE, colClasses=NA, keepFormulas=FALSE, encoding="unknown", ...)

* sheetIndex = n tells R to import the first worksheet (working numerically).
* sheetName = x specify the name of the sheet as a character string "x".
* rowIndex = n the rows that you want to read (if start and end row are not specified)
* startRow = n tells R where to start reading the data (if not the first row). In this case if we had not started at startRow = 5, the header would have appeared as "Figure.A.1.1.1.Trend.in.patent.applications.worldwide" followed by more text. To try this for yourself change the startRow to 1 and reimport the data giving wipotrends a different name.
* endRow = n tells R where to stop reading the data. Note that in this case the data stops at row 23 from the first row. You do not need to specify this value but in some cases R will read in NA values for extra rows below the actual data (try excluding endRow = and reimport the data to test this).
* colIndex = n The number or numbers of the columns you want to select for import.
* as.data.frame = tells R whether to convert the data into a data frame. Generally this is a good thing. The default will import the data as a list.
* header = TRUE tells R whether or not there are column headings in the start row.
* colClasses = x characters representing the class of each column (numeric, character, Date, POSIXct only).Otherwise converted to character columns.
* keepFormulas = TRUE or FALSE. If TRUE will convert formulas into text. Useful for some calculated fields but beware sub-totals and totals in row fields for later calculations in R.
* encoding = x If you know what the encoding is (e.g. Latin-1 or UTF-8) you may want to specify it. Note that this argument simply reads as the encoding and does not convert.
* ... additional arguments for data.frame, such as stringsAsFactors = TRUE or FALSE

In general it is good practice in your work to create Excel workbooks with 1 sheet and headings in the first row. However, as we can see from the WIPO example, reality tends to be different. That means that it is important to inspect and clean the data before hand. Keep a copy of the original file for reference by creating a .zip file. Other things to consider are:

1. Checking for corrupted characters and correcting them using find and replace in Excel or Open Office (see this [video](https://youtu.be/YYaMEbJW7Qw?list=PLsZOGmKUMi54n8R06U1HmxNywt0bAFays)).
2. Tidy up column names by removing characters such as '' or brackets that could cause problems (for example R will generally import inventor(s) as inventor.s). Consider removing blank spaces in column titles or replacing with '\_' and regularising the case (e.g. all lower case ). This will make life easier later.
3. Dealing with any leading or trailing spaces using =TRIM() in Excel or Open Office.
4. Filling blank cells with NA (see this quick [video](https://youtu.be/40isuia2w3w?list=PLsZOGmKUMi54n8R06U1HmxNywt0bAFays))
5. Any formulas, such as column or row sum functions, may not be wanted and could cause confusion when you run your own calculations.

The above preparation steps will generally take a few minutes but can save a lot of work later on. Jeff Leek provides a very good guide to preparatory steps in [The Elements of Data Analytic Style](https://leanpub.com/datastyle) and we will be following these steps in our patent analysis work.

Let's take a look at wipotrends in the console:

wipotrends

In reviewing wipotrends note that the row numbers refer to data rows (we have excluded the padding in rows 1 -4). If we were spending time with this data we might also want to turn the columns to lowercase and growth rate to growth\_rate (but see below on readxl).

## Writing Excel Files

It is generally better to write a .csv file rather than an Excel file because the results can be used in a wider range of tools (including Excel) and will be cleaner (see below). However, to write an Excel file with the new data frame use the write.xlsx() function. Before running the command it is generally a good idea to use the command getwd() to display the working directory you are in so that you know where the file will be saved. To change the directory to a new location use setwd("yourpathtofile").

write.xlsx(wipotrends, "yourfilenamepath\_new.xlsx", sheetName="Sheet1", col.names = TRUE, row.names = TRUE, append = FALSE, showNA = TRUE)

This will create a new file called wipotrends\_new. Note three points here:

1. Give your file a **new name** if writing into the same directory. Otherwise R will overwrite your existing file. Assuming you don't want to overwrite the original give the new file a sensible name.
2. If you select row.names = FALSE R will write a new column with row numbers (in this case).
3. append = TRUE or FALSE. Decide whether to append the data in an existing file or not.
4. Selecting showNA = TRUE will fill any blank cells with NA. That is useful when coming back into R to tidy up and select data. Blank cells are the enemy of calculations and it is better to fill the cells with a value where possible.

## Writing Excel to CSV

While Excel is popular in reality it is better to use .csv when using or sharing data across a range of software tools. To write results into .csv use write.csv(). Call up the description for write.csv with ?write.csv in console. See the .csv walk through for further details.

write.csv(wipotrends, file = "yourfilenamepath\_new.csv", row.names = FALSE)

# Using the readxl package

readxl is a new package from RStudio and is still a work in progress. We will cover it here because as the package develops it will become more popular and you are more likely to use it.

install.packages("readxl")

library(readxl)

At the moment readxl version 0.1.0 has two functions.

1. excel\_sheets(path) where path is the path to the xls/xlsx file. This function will list all the sheets in an excel spreadsheet to help you select the sheet that you want to import.

For example, if we add a couple of random sheets to wipotrends and then use excel\_sheets("myfilenamepath") will provide names that look something like this:

[1] "Sheet1" "my sheet" "another sheet"

This is very helpful if you don't know how many sheets are in a workbook or you want to call them by name. 2. read\_excel()

read\_wipo <- read\_excel("/Users/pauloldham17inch/Desktop/open\_source\_master/2\_datasets/wipo/wipotrends.xlsx", col\_names = TRUE, na = "", skip = 5)

When we read in this file and print it to the console we will notice something unusual.

read\_wipo

That is while we have specified that col\_names = TRUE and skip = 5 the function has not returned the column names in the dataset. While this is a bit puzzling ( the package was released less than a month ago), it suggests that it is still a work in progress. Unless this is a glitch with our data then one option would be to specify col\_names = FALSE and then rename the X0 X1 X2 column names that are generated. readr is under active development and you can follow its progress [here](%22https://github.com/hadley/readxl%22).

This is a useful reminder of one of the important principles of clean and tidy data. The first row should contain the column names.

Bear in mind that readxl may struggle with reading dates correctly, because they can be ambiguous, but expect that to also change in the future.

At the time of writing there is no write\_excel function but that may change.

The main advantage of read\_excel (as with read\_csv in the readr package) is that the data imports into an easy to print object with three attributes a tbl\_df a tbl and a data.frame. However, it is best to ensure that the data has column headings in the first row before using these functions.

If you are using dplyr and tidyr (and we assume that you will be) then the creation of a tbl\_df makes life much easier.

In summary, readxl is a welcome development for those who prefer using Excel (or are forced too), but it is very recent. It's main strength is the ability to easily see what worksheets are in a workbook and also the automatic creation of a data frame or data frame table at the time of import. The absence of a write function will hopefully encourage hardened Excel uses to adopt comma separated or tab delimited files as their standard and to take advantage of the fuller functionality of the readr package. You know it makes sense.

## Reading Excel files from URL locations

It is faster to simply download the file to your drive, or swim the Atlantic ocean, than to successfully download an excel file on http: or, in particular https:. So maybe ask yourself what is the path of least resistance and run with that. If you find an effective method please contact me and I will update this article.

## Getting Help and Further Resources

1. For additional functionality experiment with the very useful XLConnect package in Packages. Read the documentation on [CRAN](http://cran.r-project.org/web/packages/XLConnect/index.html). This adds a lot of functionality in working with Excel files in R.
2. See the R-bloggers [overview](http://www.r-bloggers.com/read-excel-files-from-r/) article on the range of packages for working with Excel files.
3. See this [tutorial](http://www.r-bloggers.com/r-tutorial-on-reading-and-importing-excel-files-into-r/).
4. And this [tutorial](http://www.milanor.net/blog/?p=779).