R

R is an open-source programming language and is often the go-to in fields such as statistics, visualization and other branches concerned with huge amounts of data. The language is heavily inspired by *S* developed at Bell Laboratories and thus one may find them to be quite similar. Being an open-source project under *GNU*, *R* is free to use both personally and commercially.

Even though *R* is often used as a tool to explore data, *R* is in fact a true programming language allowing the user to write and reuse procedures. Doing so, eases the work flow working on datasets with a reappearing structures in the sense that once a procedure is defined, it can easily be reused. Due to the fact that it combines the best of both world in statistics and programming it is a good candidate for areas such as machine learning too. Other than being a strong environment in itself, offering many standard operation, one of the true strength in *R* is its extensive range of packages. Being open-source, developers are constantly adding new packages to the environment extending the functionality of *R* whereof most of them are also free to use. The packages are what makes *R* really shine, as it adds some crucial functionality which has undoubtedly contributed to its mass popularity. Some of the packages such as *knitr* allows the user to write full reports alongside their results an convert them to various formats. Other packages makes some procedures less complicated by wrapping them into single methods, making *R* feel slightly more like a high-level programming language than it already is and allows the user to get results faster. Another strength of *R* which aided it in gaining mass popularity amongst the scientific community is its outstanding possibilities to visually display data. Other than having an array of different options in regards of how to visualize the data, being a programming language, one can alter how the specific visualization will look, making it highly customizable and almost limitless to the advanced user. Once again, packages extend the integrated functionalities and some mentionable ones would be *ggplot2* which makes it easier to plot data and takes care of the more complicated aspects such as drawing legends or making multi-layers. Another one would be *Shiny*, which also allows the user to make dynamic visualizations and add components to alter the plotted view.

Before visualizing data one has to clean and reformat it in respect to the preparation phase described by Jonas Schley. *R* offers a range of different options when reformatting and supports a various number of data types to store an alter the data within. Other than supporting some basic data types such as arrays and lists, it also supports *vectors*. A vector is simply put a sequence of elements in an array-like structure. Vectors can be accessed through indexes like an array but the size of it will grow an shrink according to how much data it contains. Vectors have multiple purposes and may be used to describe a sequence of values along an axis in a coordinate system or may just be used as a middle-ground to construct some other rather sophisticated data structures such as the *data frame*. Data frames are commonly used as the go-to input type in different methods and thus, the user will often need the data converted into this format at some point. Data frames are very much like a matrix structure-wise but they are slightly more general purpose in the sense that each column does not need to have the same data type. Having this attribute makes it a very good candidate for storing large dataset, as it is more flexible in terms of what each cell actually contains. Often R will have an built-in method to import a data set into the environment and convert it into a data frame but sometimes a package will need to do the trick. However, as some datasets may have a more complicated structure than a data frame, for instance by having multiple in-going layers, a direct conversion is not always an option and some manual work must be done to built the dataset. When a data frame is constructed, the user often want to restructure is. Unfortunately, *R* does not offer an easy and intuitive way itself to get such job done. Luckily however the packages *tidyr* and *dplyr* comes in handy when facing such issues. *Tidyr* helps obtaining tidy data set, meaning that each column represents a one variable only and each row contains a single record. To obtain a tidy dataset, the package allows operations on a data frame to reshape the structure which may be to split column into two or converting a row into columns. *Dplyr* on the other hand allows the user to do operations on the data frame like selecting single columns, add new ones and much more. Combined, these to makes up for an excellent framework to sanitize data before having to visualize it.

However, even though R in some cases are indeed an excellent choice it also have its limitation where other technologies may be better of. The way *R* is built makes it very prone to heavy memory costs partly because it allows the user to do single line execution and thus have to save all assignments of variables. Singly line execution is indeed a neat feature whilst developing code but may not always be desirable. In terms of speed it is very often criticized for being really slow which is often a rather common trait in high-level programming. In a test done by Jacob Simmering he found that *R* performed 270 times slower than *C* whilst looking for prime numbers. Some factors such as the fact that `C` is compiled at run-time may have an impact on the result for sure. However, another language which is neither compiled is python which turned out to have 17 times faster performance relative to *R*. Some tricks can however be made to boost the performance of *R*, as Simmering purposes a *byte code compiler* which can increase the speed of *R* tremendously. In regards to competitors, some users tends towards technologies with a more WYSIWYG focused approach such as *Excel*. Whilst R is really good at exploring data once the method to get there are defined, it takes a while to wind up the configuration. Spreadsheets like *Excel* are rather fast at getting smaller tasks done and are arguably visually prettier for presentation purposes which makes it a viable option in some scenarios. However, excel may fall short if the tasks become more complex, for instance if the data has to be formatted before being visualized or has to be presented in some highly customizable way where the bounds of *R* are almost limitless. *Python* is another viable option when doing statistics and is a more programming focused solution. Both solutions are indeed viable options and it is a matter of taste to which one may prefer. As argued earlier, Python may win when it comes to speed and due to the fact that it is more programmer oriented language, small task which no one has done before may be accomplished more easily where *R* tends to be slightly more reliant on packages which is a doubled-edged sword.