Seminar 4

Packages and Data Import

Packages

Packages are a collection of functions, documentation, and in some cases data files. These are similar to libraries in other programing languages. When loaded into memory, they provide specific functionality, analysis, visualization, or reporting.

R being open source enabled a great number of packages serving any function imaginable, making it one of the most popular tools for analytics. If a feature is missing, you can easily write your own functions (as we learned in the last class) and bundle them into a package for others to use. In this tutorial, we will go over the basics of package management in R.

- **Basics**

Before getting into the details of package management, let us learn how to obtain and activate a package.

If the package you need is not already installed in your system, you can easily obtain it from CRAN via the install.packages() function. Below you can see the command to install the rgl package, a set of functions for generating 3D interactive graphics (e.g., the plot3d function returns 3D plots.).

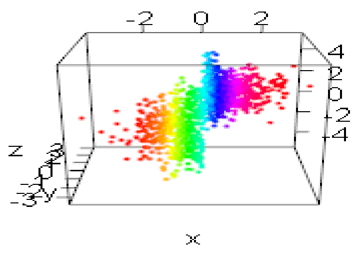
```
> install.packages(“rgl”)          # Notice the quotes around the package name
```

Pretty soon the package will be installed. Installing a package does not mean it is ready to use. You also have to activate it. For this purpose you can use the library() function.

```
> library(rgl)                     # Notice the lack of quotes around the package name
```

Let us see the example of plot3D function included in the rgl package.

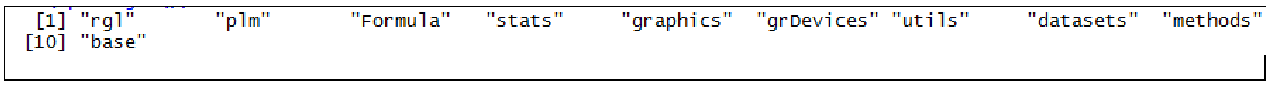
```
> example(plot3D)
```



- **Details**

If you want to get a list of all the packages that are currently loaded in the memory (activated) use the .packages() command.

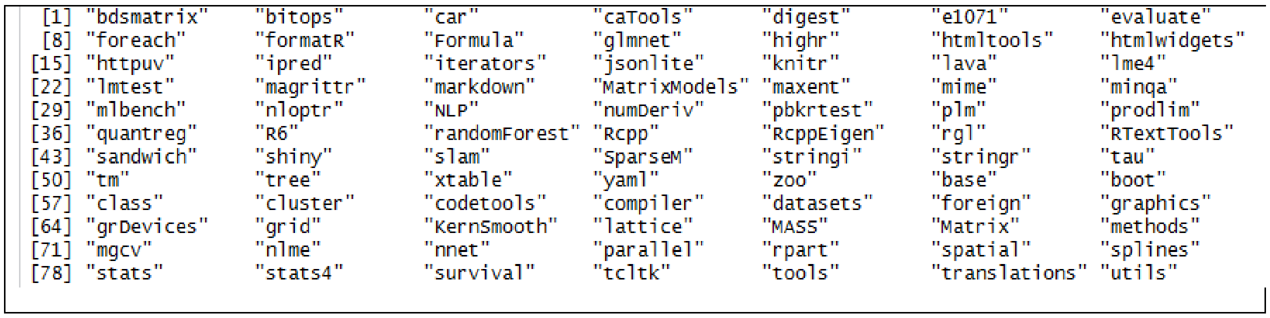
> (.packages()) # Notice the parentheses



```
 [1] "rgl"       "plm"       "Formula"   "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"  
[10] "base"     
```

If you want to get a list of all packages installed in the system use the parameter. I have installed over 80 packages.

.packages(all.available = TRUE) # or, library()



```
 [1] "bdsmatrix"    "bitops"       "car"          "caTools"      "digest"       "e1071"        "evaluate"    
 [8] "foreach"      "formatR"      "Formula"      "glmnet"       "highr"        "htmltools"    "htmlwidgets" 
[15] "httpuv"       "ipred"        "iterators"    "jsonlite"     "knitr"        "lava"         "lme4"        
[22] "lmtest"       "magrittr"     "markdown"     "MatrixModels" "maxent"       "mime"         "minqa"       
[29] "mlbench"      "nloptr"       "NLP"          "numDeriv"     "pbkrtest"     "plm"          "prodlim"     
[36] "quantreg"     "R6"           "randomForest" "Rcpp"         "RcppEigen"    "rgl"          "RTextTools"  
[43] "sandwich"     "shiny"        "slam"         "SparseM"      "stringi"      "stringr"      "tau"         
[50] "tm"           "tree"         "xtable"       "yaml"         "zoo"          "base"         "boot"        
[57] "class"        "cluster"      "codetools"    "compiler"     "datasets"     "foreign"      "graphics"    
[64] "grDevices"    "grid"         "KernSmooth"   "lattice"      "MASS"         "Matrix"       "methods"     
[71] "mgcv"         "nlme"         "nnet"         "parallel"     "rpart"        "spatial"      "splines"     
[78] "stats"        "stats4"       "survival"     "tcltk"        "tools"        "translations" "utils"       
```

- **How to find out about packages?**

If you want to find packages to carry out the tasks you have at hand, the first place to start would be either CRAN(https://cran.r-project.org/web/packages/) or R Forge (https://r-forge.r-project.org/). R site search (http://search.r-project.org, also available with command RSiteSearch()) provides a great search engine for documentation. R seek (http://rseek.org/) is a great search engine dedicated to scanning R related sources. Of course, you can always use Google as well but the letter R is a bit ambiguous in some cases.

If you want to search a certain word in installed packages' documentation, you can always use ?? or help.search()

> ??regression

> help.search("regression")

- **Commonly Used Packages**

Data Manipulation

dplyr: This is a package that adds additional functions to efficiently manipulate data. It uses data.frames so you do not have compatibility issues in your code.

Statistics

R base package (that comes loaded) includes your basic statistics and statistical models such as regressions.

Machine Learning

caret: This package provides a nice set of functions for almost all predictive analytics needs. It is discussed in deeper detail later.

Classifiers (KNN, LVQ)

K Nearest Neighbor matching and Learning Vector Quantization and more are available through class package.

Support Vector Machines (SVM)

Kernlab and e1071: These packages are two among many packages that provide this functionality.

Clustering

Base: this package has functionality to enable basic clustering techniques like K means (kmeans()), or hierarchical (hclust()) clustering. For model based clustering there is the mclust package.

Data Import

We will learn the most common data import functions. Once you learn the basics, you can use the same principles to any type of analysis task at hand. Make sure you are using the same working directory (where your R script files are saved in) as we did in the last session. Copy the files (admission.csv and admission.xlsx) on our course site (Course Documents > Seminar Slides > Seminar 4 – Exploring Data) to your working directory. We will use the data from UCLA Institute of Digital Research Education website (https://idre.ucla.edu/) which presents a valuable resource for statistics. The dataset is about graduate school admissions.

- **Comma Separated Value Files**

This is the most common file format there is to transfer data. It is human readable, and is compatible with almost any system out there. The basic idea is, the columns (variables) are separated by commas and rows (observations or records) are separated by new lines. I would recommend you use this format unless you have compelling reason to do otherwise.

R uses the read.csv function to import this file type. Let us learn more about this function.

```
> help(read.csv)
```

As you can see, the function takes quite a few parameters but most are optional with reasonable defaults (meaning you can safely leave them blank).

Once you placed this file in your working (project) directory you can import its contents as follows.

```
> admit_csv <- read.csv("admission.csv")    # Save file contents into a variable called admit_csv
```

```
> admit_csv                                 # View the contents
```

To view the contents of this new variable in a spreadsheet format, double click the csv in Environment pane of R studio (Area B), or write the command below.

```
>  View(admit_csv)                          # We use the View function to display results in R Studio
```

You can also explore the file contents via R functions.

```
> str(admit_csv)                            # Structure
```

```
> summary(admit_csv)                        # Descriptive Statistics
```

- **Excel Files**

This file format is very common due to the popularity of Microsoft's office suite. To import this file type we need two additional steps, installing and loading packages of which we leaned just before. If needed, you have to install *Java* on your laptop or PC (see: https://www.java.com/en/download/help/download_options.xml).

```
> install.packages("xlsx")        # Install the xlsx package that enables xlxs import function
```

```
> library(xlsx)                   # Activate the package
```

Now let us read the xlsx file into R. Note that I had to specify which worksheet to import.

```
> admit_xlsx <- read.xlsx("admission.xlsx", sheetIndex = 1)  # Import the file
```

```
> admit_xlsx                                                 # View contents
```

The package has other functionality that you can discover on your own.

- **Data Import from Other Statistical Packages**

R enables importing data from other statistical packages as well. I would recommend exporting the data from other packages in a csv, and then importing it that way to minimize incompatibilities. If you have no access to the offending statistical package, then you can easily use one of the packages listed below.

SAS: The package used for SAS data format after version 7 is called 'sas7bdat'. The function used is called read.sas7bdat(). If you have exported the data from SAS, you can use 'foreign' package with read.xport() function.

SPSS: SPSS sav files can be imported with 'foreign' package's read.spss() function. Exported por files can be imported with 'Hmisc' package's spss.get() function.