Tutorial: The R Programming Language

by Kevin Kotzé

The original development of **R** was undertaken by Robert Gentleman and Ross Ihaka at the University of Auckland during the early 1990's and it is arguably the most popular statistical programming language.[1] The continuing development of this open source programming language has since been taken over by an international team of academics, computer programmers, statisticians and mathematicians. One of the reasons for the popularity of **R** is that it is free of cost, which implies that users can contribute towards a world that values data literacy, irrespective of personal means (to a certain extent). In addition, users are also able to implement what is almost certainly the most extensive range of statistical models, using existing **R** code.

One of the key features of this open-source language is that users can extend, enhance, and replace any part of it. In addition, such modifications may be made available to other users, in the form of various **R** packages. For example, there are several thousand packages on the Comprehensive R Archive Network (**CRAN**), which adhere to a coding standard and are subject to unit testing. In addition, there are also many more **R** packages that may be found on user repositories, which may be located on **GitLab**, **GitHub**, **BitBucket**, etc. Since one does not need to obtain a user licence to make use of this software, it is relatively painless to move a specific task onto the cloud, which provides users with impressive computing power and storage facilities that may be provided by vendors, such as Amazon Web Services (**AWS**), Google Cloud Platform (**GCP**), Microsoft **Azure**, **DigitalOcean**, etc.

The **R** language can be run on almost any operating system and it facilitates object-oriented programming, functional programming, and more. It can be used to work with a wide variety of different types of datasets and there are a number of interfaces that have been developed which allow for **R** to work with other programs, source code, databases and application programming interfaces (APIs). The list of interfaces is too long to mention here, but would include **Excel**, **Matlab**, **Stata**, **Python**, **Julia**, **C++**, **Octave**, **JavaScript**, **Fortran**, **Spark**, **Hadoop**, **SQL**, **Oracle**, **Bloomberg**, **Datastream**, etc.

Many research institutes, companies, and universities have also migrated to **R**, so there is a good chance that you will need to be able to work with this software at some point in the future. In my subjective opinion, the **R** language is particularly well suited for the following three tasks (a) *data wrangling*, (b) *statistical modelling*& *machine learning*, and (c) *data visualisation*.

1 Resources

There are potentially more resources about learning various features of **R** than one could read in a lifetime. Therefore, what follows is a subjective view of selected resources that I have found helpful.

If you are new to **R** and would like a comprehensive introduction, then I would suggest that you may want to go through Grolemund (2020), which may be read in the form of an ebook at:

https://rstudio-education.github.io/hopr/ (https://rstudio-education.github.io/hopr/)

Alternatively, you may find that Locke (2017b) provides a reassuring introduction to the modern **R** environment. A PDF version of the book can be downloaded from the authors website at:

https://itsalocke.com/files/workingwithr.pdf (https://itsalocke.com/files/workingwithr.pdf)

If you find that neither of these are helpful then you can look at **CRAN**'s link to Contributed Documentation, which may be found here:

https://cran.r-project.org/other-docs.html (https://cran.r-project.org/other-docs.html)

Or alternatively, you may find an alternative ebook that has been written from within **R** at the sites:

https://bookdown.org/ (https://bookdown.org/) or https://bookdown.org/home/archive/ (https://bookdown.org/home/archive/)

If you prefer to learn through online lectures, then there are also a number of online courses that provide a great introduction. For example, you may wish to take a look at:

https://www.datacamp.com/courses/free-introduction-to-r (https://www.datacamp.com/courses/free-introduction-to-r)

Unfortunately, not all of this material is provided at no cost, but all UCT staff and students can access all the online training material at https://www.linkedin.com/learning (https://www.linkedin.com/learning), where you may want to consider taking the courses:

https://www.linkedin.com/learning/learning-r/welcome?u=70295562 (https://www.linkedin.com/learning/learning-r/welcome?u=70295562)

https://www.linkedin.com/learning/r-statistics-essential-training/welcome?u=70295562 (https://www.linkedin.com/learning/r-statistics-essential-training/welcome?u=70295562)

https://www.linkedin.com/learning/data-science-foundations-fundamentals/exercise-files?u=70295562 (https://www.linkedin.com/learning/data-science-foundations-fundamentals/exercise-files?u=70295562)

Other options include learning by doing, where you can get to grips with the functionality of **R** from within **R**, with the aid of the **swirl** package. Details of this package can be found at:

http://swirlstats.com/ (http://swirlstats.com/)

For those students who would like to implement the models from Wooldridge (2020), which is titled "Introductory Econometrics: A Modern Approach," with **R** code then you can make use of the ebook that is avaiable at:

http://www.urfie.net/ (http://www.urfie.net/)

The academic publisher Springer has released a relatively inexpensive and focused series of books, titled *Use R!*, which discuss the use of **R** in a various subject areas. The series now includes over seventy titles and they all contain source code. You can find these titles at:

http://www.springer.com/series/6991?detailsPage=titles (http://www.springer.com/series/6991?detailsPage=titles)

The publisher CRC Press (which is part of the Taylor & Francis Group) have a similar series of books, which may be viewed at:

https://www.crcpress.com/Chapman--HallCRC-The-R-Series/book-series/CRCTHERSER (https://www.crcpress.com/Chapman--HallCRC-The-R-Series/book-series/CRCTHERSER)

Both of these series include introductory texts.

2 Installing R on your machine

To download and install **R** for the first time you can go the webpage:

http://cran.r-project.org (http://cran.r-project.org)

where you will then need to download the version of **R** that is applicable to your operating system. If this is the first time that you are installing **R** on a Windows machine then select *install R for the first time*. Mac OS X users should download the most recent `R-x.x.x.pkg` and linux users will need to select the appropriate distribution (i.e. debian, redhat, suse, or ubuntu) and follow the instructions that are provided.

In addition to installing **R**, I would also recommend that you install the development tools as this will allow you to interface with **GitLab** or **GitHub** accounts and it will provide access to many other interesting features of this programming language.

To accomplish this task, Windows users should download and install the **Rtools** program, which is available at:

https://cran.r-project.org/bin/windows/Rtools/ (https://cran.r-project.org/bin/windows/Rtools/)

Apple mac users should consider installing the **Xcode** command line tools, which may be downloaded at: https://developer.apple.com/downloads (https://developer.apple.com/downloads), while the development tools for **R** may be found at https://cran.r-project.org/bin/macosx/tools/ (https://cran.r-project.org/bin/macosx/tools/).

Linux users should install the **R** development package, usually called `r-base-dev` using the Advanced Package Tool (APT).

Although this download will include a graphical user interface (GUI), I do not know of a single user who makes use of it. In what follows, I've described the use of two popular integrated development environments (IDE), which are **JupyterLab** and **RStudio**. Your choice between these two IDEs will depend upon your preferences, where if you mainly intend to use another language, such as **Python** or **Julia**, and want to use a single IDE for all your needs while learning these languages, then **JupyterLab** is probably going to be a good choice. Alternatively, if you would like to maintain a keen focus on statistical modelling & machine learning and see yourself as a regular user of **R**, then **RStudio** would probably be a better option. In addition, **RStudio** is also much easier to install.

During the subsequent tutorials we will be using **RStudio**, although I will ensure that you will be able to run the scripts on **JupyterLab**. So the choice of IDE is largely up to you.

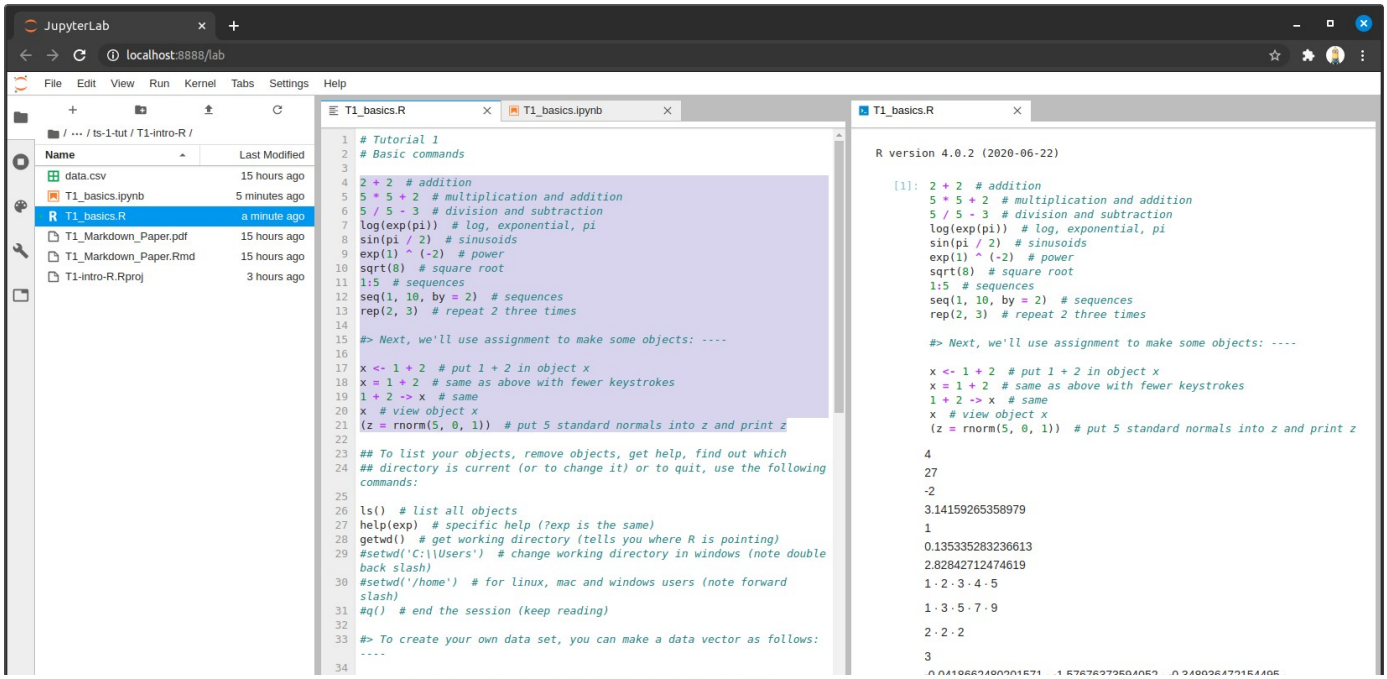
2.1 JupyterLab

JupyterLab is an open-source web-based IDE that may be used to execute code for a large number of different languages through the use of different kernels. In addition, it also allows users to write `Jupyter Notebooks`, which contain source code, equations, visualisations and narrative text. You can find additional details about this project at: https://jupyter.org/ (https://jupyter.org/) and a list of all the available kernels is provided at: https://github.com/jupyter/jupyter/wiki/Jupyter-kernels (https://github.com/jupyter/jupyter/wiki/Jupyter-kernels).

To install this IDE can take a little time and there are many different options that are available to you. It requires that you have a working version of **Python** installed on your machine. For this purpose I have installed **Anaconda**, for which you can find details at https://www.anaconda.com/ (https://www.anaconda.com/). To install the open-source Individual Edition you should go to this page: https://www.anaconda.com/products/individual (https://www.anaconda.com/products/individual), while additional details relating to installation can be found at https://docs.anaconda.com/anaconda/install/ (https://docs.anaconda.com/anaconda/install/).

Once **Anaconda** is installed you should be able to find the **Anaconda prompt** on Windows or a use the **Terminal** on Linux or Mac OS X, where you can follow the instructions to install **JupyterLab**, which are located at https://jupyterlab.readthedocs.io/en/stable/ (https://jupyterlab.readthedocs.io/en/stable/). In addition, I found that I also need to install `jupyter_client`. To allow for **JupyterLab** to use the **R** kernel that you have already installed then you can use the **IRkernel**, which has documentation and installation instructions that are located at https://irkernel.github.io/ (https://irkernel.github.io/).

For existing **Python** users this installation is probably relatively straightforward. However, those who are new to both **Python** and **R** may find that this process is a little challenging. Thankfully, there are plenty of blogposts and online videos that describe the installation procedure on various different operating systems. After completing this setup you will be able to both create and execute **R** code that is contained within an `*.R` script.



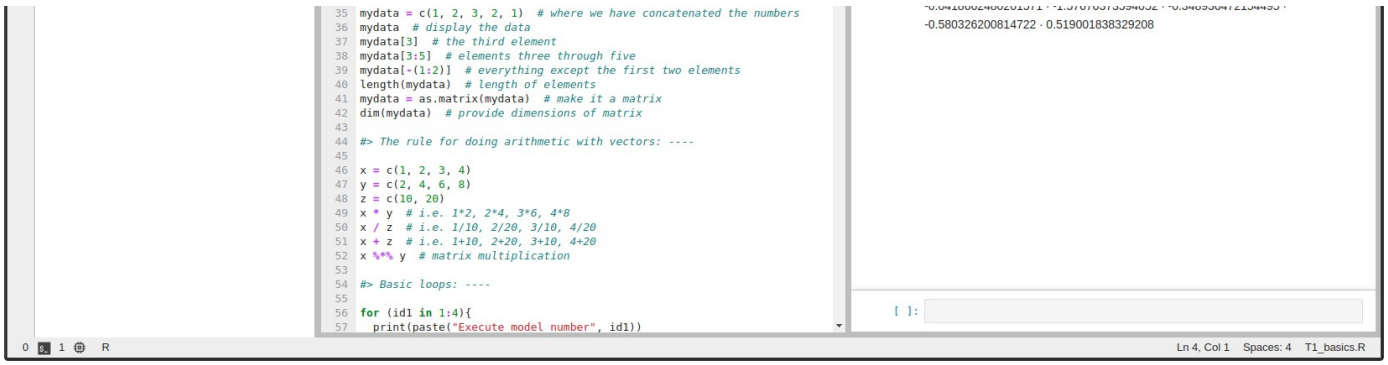


Figure 1: **JupyterLab - Working with a R script**

As an alternative you can create a `Jupyter Notebook` that contains source code, narrative text and model output all within a single document.

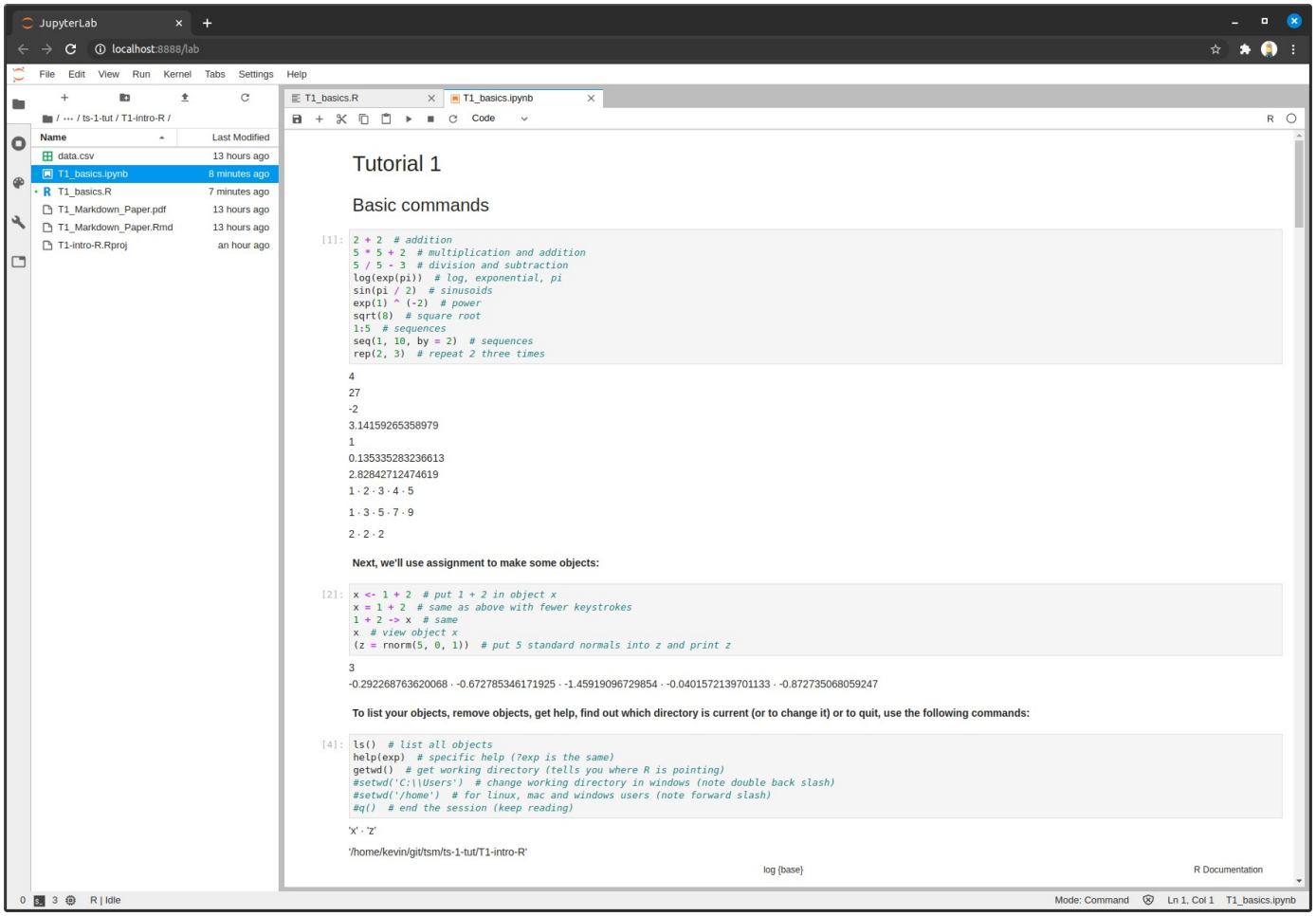


Figure 2: **JupyterLab - Working with a Jupyter Notebook**

The keyboard shortcut key to execute a selected chunk of code in **JupyterLab** is `Shift+Enter`.

2.2 RStudio

The **RStudio** IDE has made a significant contribution towards the popularity of **R**. It includes many helpful tools for integrating with **Git** repositories and publishing **R Markdown** documents that combine narrative text and code to produce elegantly formatted reports, papers, books, slides and

more. All of the **R** users that I interact with make extensive use of **RStudio**. For our applications, we will only require the functionality that is included in the **Open Source License** for **RStudio Desktop**, which can be downloaded for free at:

https://rstudio.com/products/rstudio/download/ (https://rstudio.com/products/rstudio/download/)

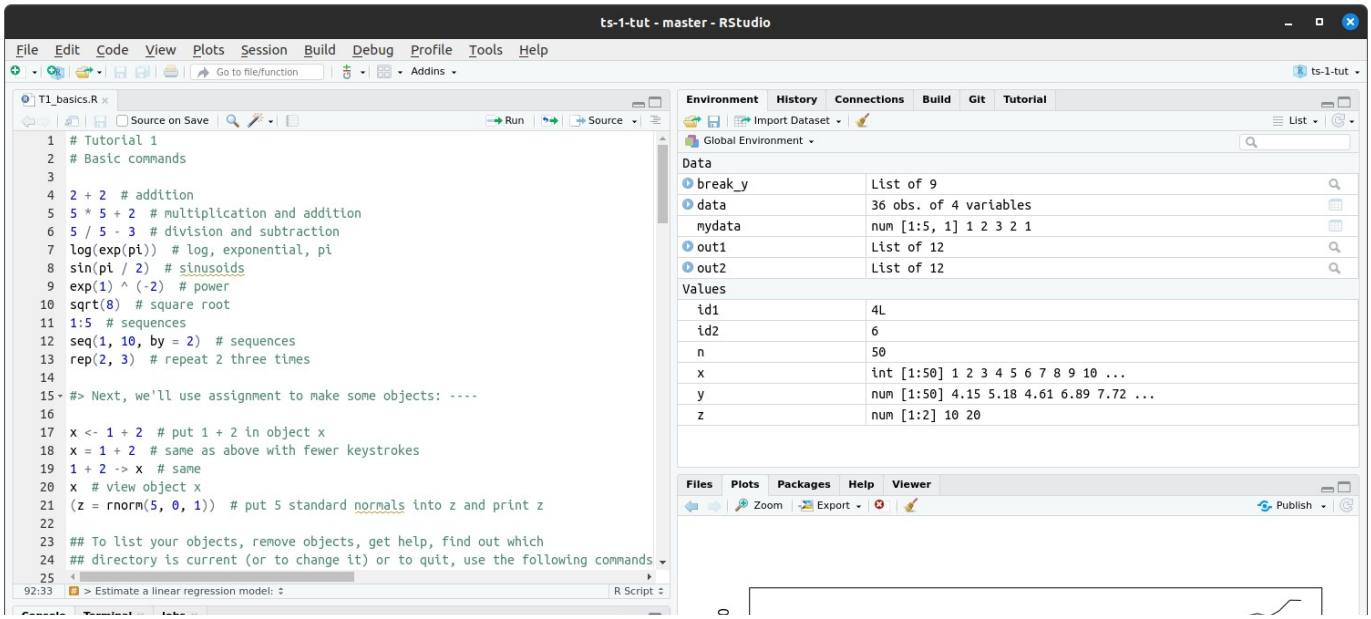
Another attractive feature of the **RStudio** is that they also provide an **Open Source License** for **RStudio Server**, which may be used on a cloud server and has a similar interface. Therefore, when users need to make use of additional computational power the project can migrated to the cloud with relatively little inconvenience. After you have installed this software, you will then be able to gain access to the wonderful world of **R** through the **RStudio** application.

If you would like to learn more about the **RStudio** IDE then you can take a look at the material that is located at https://education.rstudio.com/ (https://education.rstudio.com/), where beginners may want to pay particular attention to the resources that are mentioned at https://education.rstudio.com/learn/beginner/ (https://education.rstudio.com/learn/beginner/). In addition, **RStudio** have also release the `learnr` package that has similar functionality to the `swirl` package.

2.2.1 Desktop Overview

After you have installed the software that has been mentioned above, you will be able to open **RStudio**. During the semester we are going to make use of *project files* which will set all defaults for a respective session. After downloading the `T1-intro.zip` folder (see the links on https://www.economodel.com/time-series-analysis (https://www.economodel.com/time-series-analysis)), you should find the project file `T1-intro.Rproj`. If you double-click on this file it should open in **RStudio**. Thereafter, if you click on the **file** tab, you should see the name of a script `T1_basics.R` which you can open.

After executing a few lines it should look similar to that which is shown in Figure 3. To execute a few lines you can highlight the lines that you want to execute before using the `Ctrl+Enter` shortcut key to run these lines.



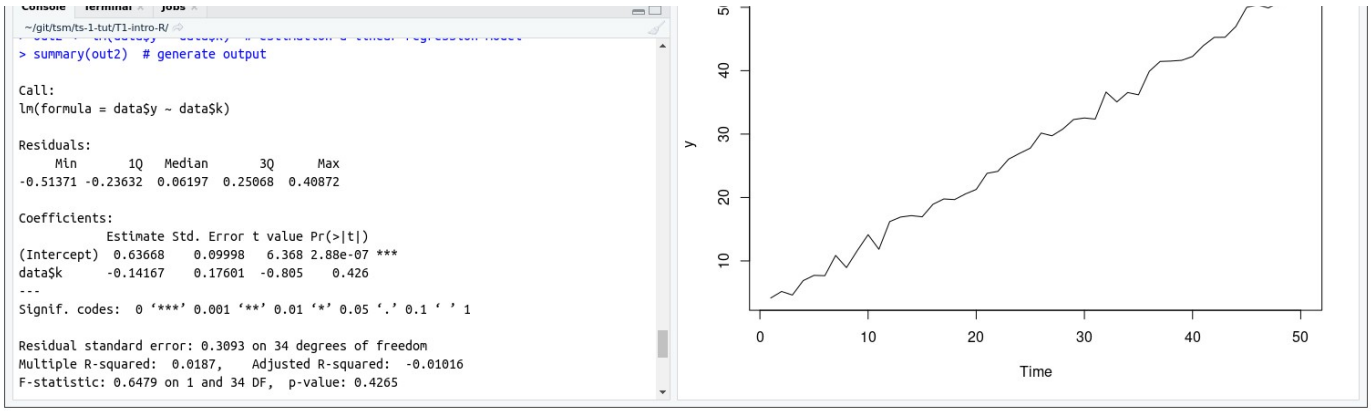
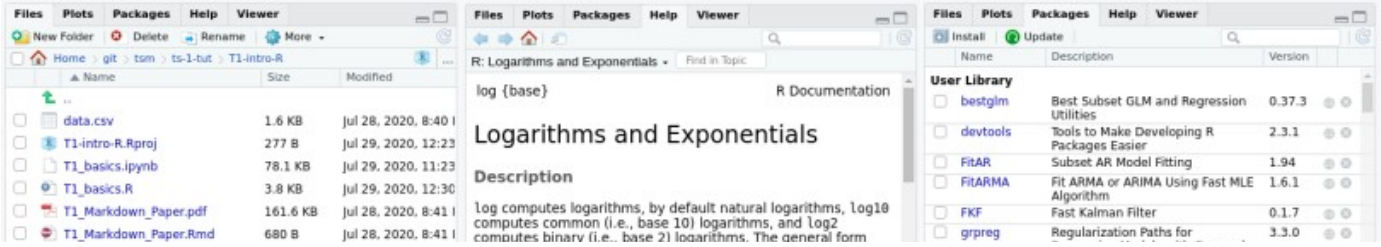
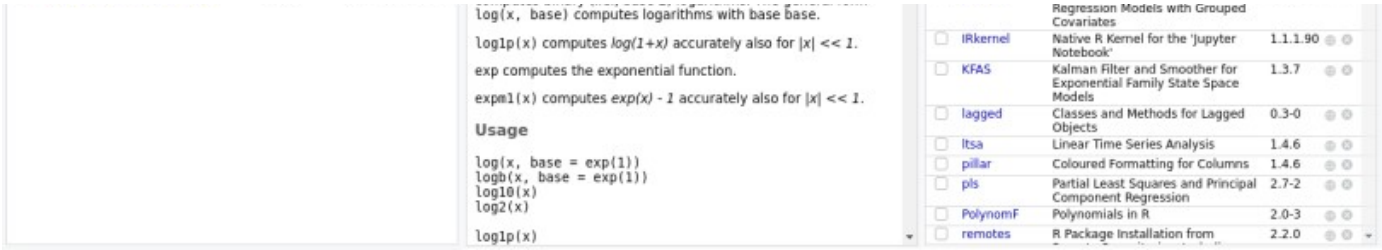


Figure 3: **Rstudio Desktop - Working with a R Script**

- After opening a particular `*.R` script is usually displayed in the top left-hand corner. One is able to view multiple program files at once and you can also execute entire files, single lines or just a subset of a line.
- All the variables, matrices, scalars, objects, strings, etc. are store in the `Environment` on the top right.
- Any conceivable type of graph may be constructed and displayed in the `Plots` panel on the bottom right. These may be exported as jpeg's, pdf's, etc.
- The `Console` on the bottom left may be used to input commands or provide output (which may be structured to your liking).
- If you double-click on a *Data* element in the `Environment` it will display it in a **table** on the left panel.
- The additional tabs on the bottom right contain additional useful information. Screen shots of these are contained below.
- The `Files` directory allows for users to navigate your way around your current directory. Take particular note of the project file, which has the extension `*.Rproj`. This file will take care of the setup for this project to ensure that we have relatively consistent functionality. The general idea is that we may include various scripts within a project so when you want to work in a project open the `*.Rproj` first before using the `Files` directory to open a script.
- There is a very useful `Help` function that provides the exact format for the respective commands. It will also contain an example of it's use as well as textbook/journal references (where appropriate).
- To install and use `Packages` you can use the respective tab, which will link you to the repository on the web. When you click on the package name you will see all the available functions in the package. In addition, some packages will include a very useful *vignette*, which is an equivalent of a journal article that explains the methodology and application of the package. It will also include a working example with code (in most cases).





3 Basic overview of programming in R

```
# > Basic commands: ----

2 + 2  # addition
```

```
## [1] 4
```

```
5 * 5 + 2  # multiplication and addition
```

```
## [1] 27
```

```
5/5 - 3  # division and subtraction
```

```
## [1] -2
```

```
log(exp(pi))  # log, exponential, pi
```

```
## [1] 3.141593
```

```
sin(pi/2)  # sinusoids
```

```
## [1] 1
```

```
exp(1)^(-2)  # power
```

```
## [1] 0.1353353
```

```
sqrt(8)  # square root
```

```
## [1] 2.828427
```

```
1:5  # sequences
```

```
## [1] 1 2 3 4 5
```

```
seq(1, 10, by = 2)  # sequences
```

```
## [1] 1 3 5 7 9
```

```
rep(2, 3)  # repeat 2 three times
```

```
## [1] 2 2 2
```

```
# > Assign values to objects: ----
x <- 1 + 2  # put 1 + 2 in object x
x <- 1 + 2  # same as above with fewer keystrokes
x  # view object x
```

```
## [1] 3
```

```
(z <- rnorm(5, 0, 1))  # put 5 standard normals into z and print
```

```
## [1] -0.4154928 -0.4494405  1.0715473  0.3543634
## [5]  0.8318592
```

```
# > List objects, get help and get current working
# directory: ----
ls()  # list all objects
```

```
## [1] "figs" "tbls" "x"    "z"
```

```
help(exp)  # specific help (?exp is the same)
getwd()  # get working directory (tells you where R is pointing)
```

```
## [1] "/home/kevin/teach/tsm/ts-1-tut"
```

```
# > Set working directory and quit: ----
setwd("C:\\Users")  # change working directory in windows (note double back slash)
setwd("/home")  # for linux, mac and windows users (note forward slash)
q()  # end the session (keep reading)
```

```
# > To create your own data set: ----
mydata <- c(1, 2, 3, 2, 1)  # where we concatenated the numbers
mydata  # display the data
```

```
## [1] 1 2 3 2 1
```

```
mydata[3]  # the third element
```

```
## [1] 3
```

```
mydata[3:5]  # elements three through five
```

```
## [1] 3 2 1
```

```
mydata[-(1:2)]  # everything except the first two elements
```

```
## [1] 3 2 1
```

```
length(mydata)  # length of elements
```

```
## [1] 5
```

```
mydata <- as.matrix(mydata)  # make it a matrix
dim(mydata)  # provide dimensions of matrix
```

```
## [1] 5 1
```

```
rownames(mydata) <- paste0("obs", 1:5)  # include row-names
colnames(mydata) <- c("rand_num")  # include column-names
print(mydata)  # print the matrix to the console
```

```
##      rand_num
## obs1        1
## obs2        2
## obs3        3
## obs4        2
## obs5        1
```

```
matrix(rnorm(12), ncol = 3)  # alternatively
```

```
##             [,1]      [,2]       [,3]
## [1,] -1.68268766  0.955590  0.7611294
## [2,] -0.26197258  1.497415  0.4957029
## [3,]  0.49216173 -1.000373  0.2864883
## [4,]  0.02016587 -1.135609 -0.2711880
```

```
array(rnorm(4 * 3 * 2), dim = c(4, 3, 2))  # multi-dimensional array
```

```
## , , 1
##
##            [,1]       [,2]      [,3]
## [1,] -0.3599192 -2.5898203  1.213437
## [2,] -0.1217434 -0.9363464  2.081092
## [3,] -1.5339068 -1.9552892 -1.452487
## [4,] -0.2498394  0.1796999  1.194551
##
## , , 2
##
##            [,1]        [,2]       [,3]
## [1,]  0.6827052 -0.35292757 -0.4338472
## [2,] -0.9000434 -0.54582740 -0.7839431
## [3,] -0.7218103  0.18334002  0.3900337
## [4,]  1.6665211 -0.07135617  1.2443935
```

```
# > Rules for doing arithmetic with vectors: ----
x <- c(1, 2, 3, 4)
y <- c(2, 4, 6, 8)
z <- c(10, 20)  # three commands are stacked with ;
x * y  # i.e. 1*2, 2*4, 3*6, 4*8
```

```
## [1]  2  8 18 32
```

```
x/z  # i.e. 1/10, 2/20, 3/10, 4/20
```

```
## [1] 0.1 0.1 0.3 0.2
```

```
x + z  # i.e. 1+10, 2+20, 3+10, 4+20
```

```
## [1] 11 22 13 24
```

```
x %*% y  # matrix multiplication
```

```
##      [,1]
## [1,]   60
```

```
# > Basic loops: ----

for (id1 in 1:4) {
    print(paste("Execute model number", id1))
}  # for loops are the most common
```

```
## [1] "Execute model number 1"
## [1] "Execute model number 2"
## [1] "Execute model number 3"
## [1] "Execute model number 4"
```

```
id2 <- 1
while (id2 < 6) {
    print(id2)
    id2 <- id2 + 1
}  # while loops can be used as an alternative
```

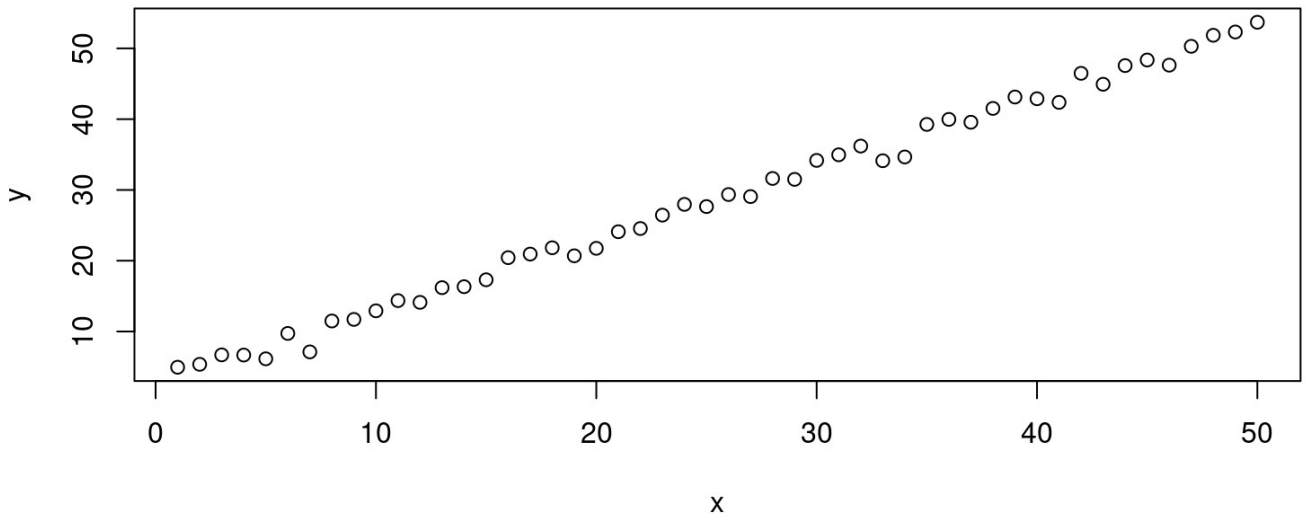
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
x <- -5  # use else statement to test if this is positive or negative

if (x == 0) {
    print("Zero")
} else if (x > 0) {
    print("Positive number")
} else {
    print("Negative number")
}
```

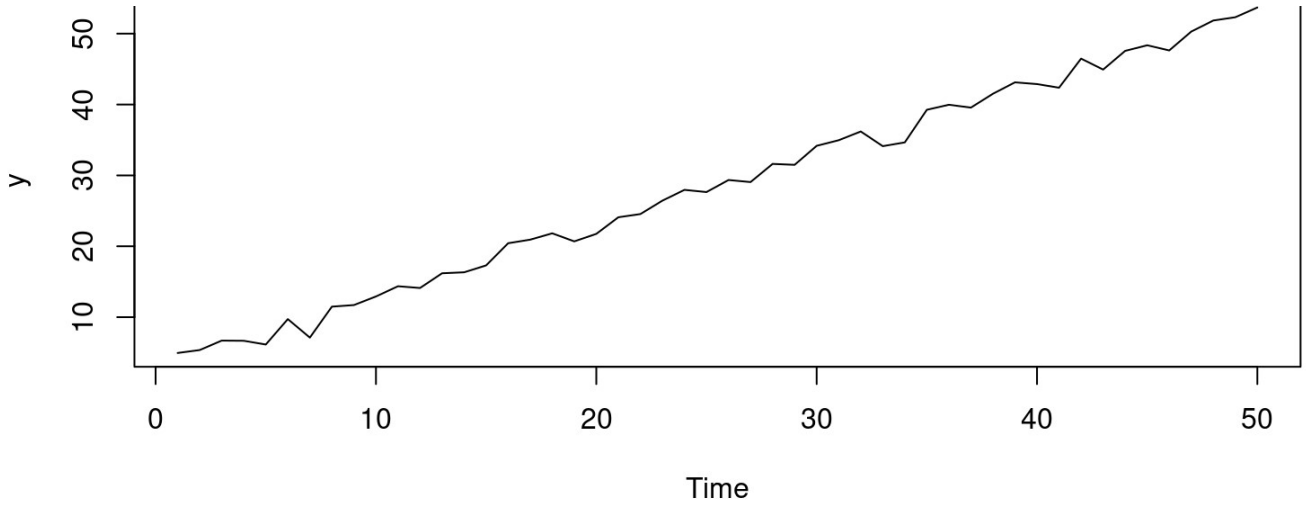
```
## [1] "Negative number"
```

```
# Could also vectorise code and use `sapply or
# `tidyverse::map` function to make code run quicker
```

```
# > Simulating regression data: ----
n <- 50  # number of observations
x <- seq(1, n)  # right-hand side variable
y <- 3 + x + rnorm(n)  # left-hand side variable

plot(x, y)  # show the linear relationship between the variables
```




```
plot.ts(y)  # plot time series
```




```
# > Estimate a linear regression model: ----
out1 <- lm(y ~ x - 1)  # regress x on y without constant
summary(out1)  # generate output
```

```
##
## Call:
## lm(formula = y ~ x - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3031 -0.2707  0.6603  1.8226  4.1629
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x 1.095131   0.008538   128.3   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.769 on 49 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.997
## F-statistic: 1.645e+04 on 1 and 49 DF,  p-value: < 2.2e-16
```

```
# > Using packages: ----

# To install and use the functions within a package
# that is located on the CRAN repository:
install.packages("strucchange")  # to download the package from CRAN
library("strucchange")  # to access the routines in this package
break_y <- Fstats(y ~ 1)  # F statistics indicate one breakpoint
breakpoints(break_y)

# Alternative syntax `<package>::<routine>` provides
# the same result:
break_y <- strucchange::Fstats(y ~ 1)  # F statistics indicate breakpoint
strucchange::breakpoints(break_y)
```

```
# Rather than repeat a string of commands you may want
# to create a function By way of example to remove the
# largest and smallest value in vector before taking
# the mean we could perform the calculation
x <- rnorm(10)
x <- sort(x)[-c(1, length(x))]
mean(x)
```

```
## [1] 0.3249423
```

```
# Or alternatively we could create the function
# `mean_excl_outl` where `x` is both the input and
# output argument
mean_excl_outl <- function(x) {
    x <- sort(x)[-c(1, length(x))]
    x <- mean(x)
    return(x)
}

# Now whenever we need to perform this calculation
mean_excl_outl(rnorm(20))
```

```
## [1] 0.3355191
```

Before continuing I would encourage you to browse over the **RStudio** cheatsheet, which can be downloaded from the following link: https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf (https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf).

3.1 Reproducible research

One of the major advantages of **R** is that there are a number of extensions that allow for it to interface with other languages. One of these extension involves the use of **R Markdown** files that weave

together narrative text and code to produce elegantly formatted output. This would also facilitate reproducible research, where you can write your entire paper, book, dashboard or slides within the **RStudio** environment. Two of the important packages that allow you to do this are **rmarkdown** and **knitr**, where the **knitr** package may be used to knit the `RMarkdown` documents to create HTML, PDF, MS Word, LaTeX Beamer presentations, HTML5 slides, Tufte-style handouts, books, dashboards, shiny applications, scientific articles, websites, and many more. For more on the use of **rmarkdown** and **knitr**, see Xie (2013), https://bookdown.org/yihui/rmarkdown-cookbook/ (https://bookdown.org/yihui/rmarkdown-cookbook/) and the website http://rmarkdown.rstudio.com/ (http://rmarkdown.rstudio.com/).

The original idea behind these forms of reproducible research programs is that the text for your report (or article) would be written in standard **LaTeX** format and the model would be included as a **R** chunk of code. This file could then be compiled within **R** to generate a `.md` (which is useful for web documents), `.pdf` or `.tex` file. An example of a basic **R** *Markdown** document is included under separate file and is named `T1_Markdown_Paper.Rmd`.

Of course, these `RMarkdown` documents can also be used to generate HTML output and the growth of this part of the ecosystem allows for the creation of relatively complex documents and dashboards with the aid of various **R** extensions. For example, more complex documents, such as books, could be written with the **bookdown** package, which is what I've used for your notes. And with the aid of **Shiny** one is able to create impressive interactive dashboards. For more on this see, https://shiny.rstudio.com/ (https://shiny.rstudio.com/).

Again, there is a useful cheatsheet that you should take a look at, which can be downloaded from https://github.com/rstudio/cheatsheets/raw/master/rmarkdown-2.0.pdf (https://github.com/rstudio/cheatsheets/raw/master/rmarkdown-2.0.pdf).

4 Other aspects of interest

4.1 Specific subject areas

Specific modelling aspects are usually covered within a dedicated package. For a list of the major econometric type of packages, including those that have been developed for cross sectional analysis, panel models, machine learning, time series, etc:

http://cran.r-project.org/web/views/Econometrics.html (http://cran.r-project.org/web/views/Econometrics.html)

https://cran.r-project.org/web/views/MachineLearning.html (https://cran.r-project.org/web/views/MachineLearning.html)

https://cran.r-project.org/web/views/TimeSeries.html (https://cran.r-project.org/web/views/TimeSeries.html)

https://cran.r-project.org/web/views/Finance.html (https://cran.r-project.org/web/views/Finance.html)

If you are also interested in computational issues, then you may wish to visit the following:

https://cran.r-project.org/web/views/NumericalMathematics.html (https://cran.r-project.org/web/views/NumericalMathematics.html)

https://cran.r-project.org/web/views/HighPerformanceComputing.html (https://cran.r-project.org/web/views/HighPerformanceComputing.html)

Alternatively for the complete set of the package categories:

https://cran.r-project.org/web/views/ (https://cran.r-project.org/web/views/)

Or you can find all the packages that have been sorted by name at:

http://cran.r-project.org/web/packages/available_packages_by_name.html (http://cran.r-project.org/web/packages/available_packages_by_name.html)

Remember also that the total number of packages on the **CRAN** repository represent only a small fraction of those that are available. So if you don't find what you like on this particular repository then it may be available elsewhere, possibly on a **GitHub** or **GitLab** repository, you just need to search.

4.2 Obtaining assistance

A further useful resource is the *Journal of Statistical Software*, which contains journal articles that seek to explain how to use many of the popular **R** packages. These articles are usually similar to the *vignettes* that accompany most packages that describe the desired functionality.

In addition, if you are struggling with a particular function in **R**, then you may want to check out the **StackOverFlow** website, which was originally developed for computer programmers (many of which make use of **R** for data science applications). The probability that your query has been previously answered is usually quite high and what is particularly nice about this website is that it has a number of methods that avoid the duplication of queries. You can check it out at: http://stackoverflow.com/ (http://stackoverflow.com/).

4.3 Interoperability

Then lastly, as most of my initial programming experience has been obtained in **Matlab**, I have also found the text by Hiebeler (2015) to be of use. He has a particularly nice summary document that can be found at:

https://umaine.edu/mathematics/wp-content/uploads/sites/70/2018/08/matlabR.pdf (https://umaine.edu/mathematics/wp-content/uploads/sites/70/2018/08/matlabR.pdf)

Similar documentation can be found for those who may be more familiar with **Python** or **Julia**.[2] If you are interested in using **Python** for time series econometrics then check out the website of Kevin Shepphard at: https://www.kevinsheppard.com (https://www.kevinsheppard.com). In addition, if you want to use either **Python** or **Julia** for applications in mathematical economics then check out the website of Thomas Sargent and John Stachurski at: http://quant-econ.net (http://quant-econ.net).

Unfortunately, I have not found anything that is all that good for **Stata** users, as the **Stata**

environment is somewhat different to that of most other scripting languages. However, I can tell you that some of the **Stata** users that I've worked with have found the transition to **R** to be relatively straightforward. There is also a fairly good cheatsheet for those who are moving between **R** and **Stata**, which may be found at: http://github.com/rstudio/cheatsheets/raw/master/stata2r.pdf (http://github.com/rstudio/cheatsheets/raw/master/stata2r.pdf). Those who are interested may find the book by Muenchen and Hilbe (2010) to be of use, although it is not a quick read.[3]

4.4 Speeding up R

While **R** does not perform computations at the speed of other lower level coding environments (such as **C++**, **C**, or **Fortran**) it does have the ability to interface with these (and other) languages. In particular one could make use of the **Rcpp** package, for which details are provided in Eddelbuettel (2013). For those who want to learn about programming in both **C++** and **R**, a thorough discussion is included in Eubank and Kupresanin (2011). In addition, there are also numerous parallel processing packages that could be used, many of which make use of a graphics processor unit (GPU). These are discussed in Matloff (2015).

Then lastly, as this is freeware, you could also make use of a cloud server or a cluster of cloud servers to perform massive computations in **R** with relative ease as it would not be necessary to purchase and install proprietary software for each instance. In addition, there are also a number of disk images and docker containers that are provided by **AWS**, **GCP**, **Azure**, and others to facilitate such a task. This provides users with an impressive amount of computing power at a relatively low cost (i.e. there is even a free tier option).

4.5 Cloud based applications with R

For those of you who are looking to make use of a cloud based platform to deploy an **R** application, then the https://cloudyr.github.io/ (https://cloudyr.github.io/) website provides a useful list of packages that make it easier to work with the various services that are provided by **AWS**, **GCP**, or **Azure**. In addition, there is also a particularly helpful Docker (https://www.docker.com/) container called Rocker. Further information is available at https://www.rocker-project.org/ (https://www.rocker-project.org/).

4.6 Developing with GitLab, BitBucket or GitHub

GitLab, **BitBucket** and **GitHub** are web-based repositories that allow for the source code management (SCM) functionality of **Git** as well as many other features. It can be used to track all the changes to programming files by any permitted user, so that you can see who may have made a change to a particular file. In addition, if it turns out that a particular branch of code did not provide the intended result, then you can always go back to a previous version of the code base. Most **R** developers make use of either **GitLab** or **GitHub** for the development of their packages and it is particularly useful when you have many developers working on a single project.

For your own personal work you should consider opening a **GitLab** account, which will allow you to store public and private repositories for free. As an alternative, you can make use of **GitHub** for a free public repository, or **BitBucket**, which provides users with free private repositories.

After installing the **Rtools** software one is also able to download and install packages that are located on **GitLab**, **BitBucket** or **GitHub** accounts, directly from within **R** with the aid of the `devtools` package. For example, the following code would find the package on **GitHub**, provided you are connected to the internet:

```
library("devtools")
install_github("<username>/<packagename>")
```

Since we only usually make use of this single command from within the `devtools` package we could also write this command as:

```
devtools::install_github("<username>/<packagename>")
```

Then lastly, if you are developing an **R** package or want to create a your own repository then you will want to create a **R-project** in **RStudio** so that you can control the uploads and downloads to your **Git** repository from within **RStudio**. For a relatively comprehensive tutorial on how to set this up, see https://happygitwithr.com/ (https://happygitwithr.com/).

4.7 Working with databases

If you need to connect to a particular database, then the https://db.rstudio.com/ (https://db.rstudio.com/) website is a wonderful resource, which describes how one is able to work with a large variety of different databases to import or export data into **R**.

5 Tidyverse

For those who would like to invest more time on programming in **R**, then you should take a look at the work of Hadley Wickham and the `tidyverse` suite of packages, which are used for various data science applications. His website may be found at http://hadley.nz/ (http://hadley.nz/) and details about the `tidyverse` can be found at: https://www.tidyverse.org/ (https://www.tidyverse.org/). The books *R for data science* and *Advanced R* are quintessential aids for venturing into this area and may be found at https://r4ds.had.co.nz/ (https://r4ds.had.co.nz/) and http://adv-r.had.co.nz/ (http://adv-r.had.co.nz/), respectively. For a brief summary of the main features of the tidyverse, you may want to consider reading the booklet by Locke (2017a), which is available for download at: https://itsalocke.com/files/DataManipulationinR.pdf (https://itsalocke.com/files/DataManipulationinR.pdf). During what remains of this semester we will make use of a few `tidyverse` functions, although the objective of this course is not to provide an exhaustive overview of these functions.

Since we will be using a few `dplyr` and `ggplot` functions throughout the course, you may want to take a look at the following cheatsheets:

- https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf (https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf)
- https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf

(https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf)

In addition, to the suite of `tidyverse` packages, which assist with a number of basic data science tasks, there is a collection of packages that seek to apply `tidyverse` principles to modelling and machine learning tools. These form part of the `tidymodels` ecosystem, which offers a consistent, flexible framework for the storing of model output, etc. You can get more information about this ecosystem from the website https://www.tidymodels.org/ (https://www.tidymodels.org/). A pre-publication draft of a book that describes the use of this new ecosystem is available at https://www.tmwr.org/ (https://www.tmwr.org/).

The `data.table` package is also one that is growing in popularity and can be used in conjunction with `tidyverse` functions. In many respects it provides equivalent functionality to what is contained in the `dplyr` package, which is part of the `tidyverse`, but makes use of more efficient ways to manipulate data. When working with large data this is particularly important. I have not made use of this package in this course as we are not going to be dealing with particularly large datasets, and the `data.table` syntax takes a little longer to learn (in my opinion). For a useful cheatsheet that lists `dplyr` and `data.table` equivalent functions next to each other, take a look at: https://atrebas.github.io/post/2019-03-03-datatable-dplyr/ (https://atrebas.github.io/post/2019-03-03-datatable-dplyr/). In addition, the vignettes that accompany the `data.table` are excellent.

And if you start working with extreme big-data then you probably want to make use **Spark**, for which the `sparklyr` package provides an accessible interface that is simialr to `dplyr`. Details regarding this package is available at https://spark.rstudio.com/ (https://spark.rstudio.com/) and in the ebook https://therinspark.com/ (https://therinspark.com/).

6 Conclusion

During the course of this semester we are really only going to touch the surface of what is possible in this programming environment. So don't worry if the above appears to be somewhat intimidating. The objective of this overview is to give you a brief introduction and a number of potentially useful sources of information that would aid your learning experience. As the world around us continues to evolve, the relative importance of skills that facilitate the efficient analysis of data will continue to increase. A thorough grasp of the **R** language would assist in this endeavour, and for many it will become an essential tool for the development of your career.

7 References

Eddelbuettel, Dirk. 2013. *Seamless r and c++ Integration with Rcpp*. UseR! Series. Springer.

Eubank, Randall L., and Ana Kupresanin. 2011. *Statistical Computing in c++ and r*. New York: Chapman Hall/CRC.

Grolemund, Garrett. 2020. *Hands-on Programming with r: Write Your Own Functions and Simulations*. Boston, MA: O'Reilly Media. https://rstudio-education.github.io/hopr/ (https://rstudio-education.github.io/hopr/).

Hiebeler, David E. 2015. *R and Matlab*. New York: Chapman & Hall/CRC.

Locke, Stephanie. 2017a. *Data Manipulation (r Fundamentals Book 2)*. Locke Data Ltd.

———. 2017b. *Working with r (r Fundamentals Book 1)*. Locke Data Ltd.

Matloff, Norman. 2015. *Parallel Computing for Data Science*. New York: Chapman Hall/CRC.

Muenchen, Robert A., and Joseph M. Hilbe. 2010. *R for Stata Users*. New York: Springer.
Wooldridge, Jeffrey M. 2020. *Introductory Econometrics: A Modern Approach*. Seventh. Boston, MA: Cengage Learning.
Xie, Yihui. 2013. *Dynamic Documents with r and Knitr*. New York: Chapman; Hall/CRC.

---

1. See, https://www.tiobe.com/tiobe-index/ (https://www.tiobe.com/tiobe-index/) or https://pypl.github.io/PYPL.html (https://pypl.github.io/PYPL.html). Note that while **Python** is also an exceptional programming language that can be used for statistical modelling, its popularity is partly due to its many other uses.↩
2. These open source languages are also developing at a rapid rate and are excellent alternatives to **R**. What is particularly nice about **Python** is that it includes relatively extensive symbolic and numerical computational packages (but possibly not as many statistical functions). **Julia** allows for more expedient computation, but requires a slightly higher level of programming skill (i.e. somewhere between **C++** and **Python**).↩
3. This is available at the UCT library.↩