Introduction to R

Intensive Statistics Course

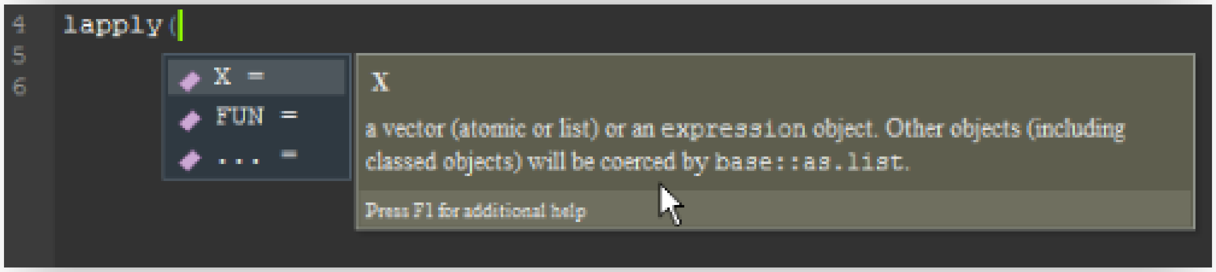
Wihan Marais

wihanmarais@sun.ac.za

25 January 2022

Why R?

1. R is free and open-source.
   - At the time of writing, a new Stata 17 annual license is priced between R11,670 and R21,280 excluding VAT.
   - Free upgrades, updates and dissemination.
   - Availability of helpful resources like **stackoverflow** and **Quick-R**.
2. R uses **packages**
   - R consists of Base-R coupled with third-party libraries of pre-written code, or packages.
   - **CRAN**, or The Comprehensive R Archive Network, is a network of ftp (file transfer protocol) and web servers around the world that store identical, up-to-date, versions of code and documentation for R.
   - More on this later.
3. R uses predictive coding (Ctrl/Cmd + Space is very useful).



```
lapply(
```

X

a vector (atomic or list) or an expression object. Other objects (including classed objects) will be coerced by base::as.list.

4. R is compatible with Markdown.
   - These lecture notes were created as a '.Rmd' file using **R Markdown**, RStudio's native authoring framework for data science.
   - See this 1-minute video summary of what R Markdown entails.

Before we start

You need the following installed on your machine:

- **R** or Base-R.

  ”R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc.”

- **RStudio**

  ”RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.”

- **Rtools**

  Select the Rtools download link for the relevant version of R installed on your machine. To determine the version currently installed, run the following code in your console. First, highlight the line of code you would like to run and Ctrl/Cmd + Enter to run.

```
sessionInfo()[1]$R.version$version.string

# IMPORTANT:
# Take care to check the box to have the installer 'edit your path'
```

To verify that we have installed Rtools properly, we need to make use of the `devtools` package.

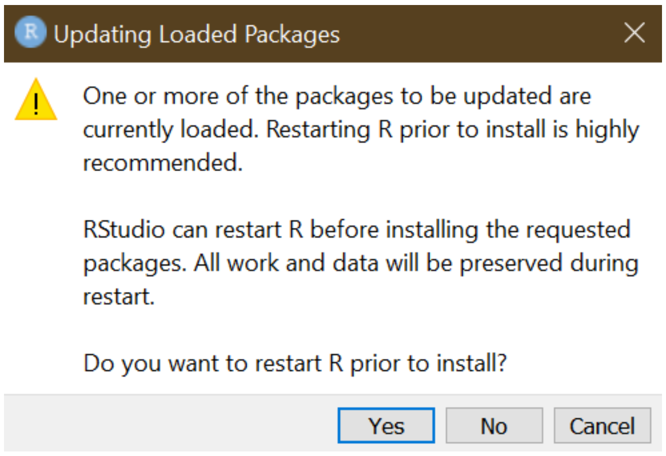
```
install.packages("devtools") # Install the package from CRAN.
library(devtools) # Load package into your current library.

find_rtools() # Run this command from the devtools package
# or
devtools::find_rtools()
# should return TRUE in your console
```

Important packages

Install and load a few packages that you would likely use often. Let’s use the `installr` package as an example.

```
# Packages need only be installed once
install.packages("installr")
# and can be loaded into your library with
library(installr)
```

However, if I have already installed `installr` before, `install.packages()` produces the following error:



One or more of the packages to be updated are currently loaded. Restarting R prior to install is highly recommended.

RStudio can restart R before installing the requested packages. All work and data will be preserved during restart.

Do you want to restart R prior to install?

It can be hard to keep track of all the packages that you have or have not installed on your machine. How, then, should we install packages?

```r
# In short: if package has not yet been installed, run code to install
if (!require(installr)) {
    install.packages("installr")
    require(installr)
}
```

Instead, I propose using the `pacman` package. It enables us to easily install new- and load old packages from curated lists such as CRAN, or any open-source package from **GitHub** using `p_load()` and `p_load_gh()`, respectively. These commands install packages if they have not yet been installed, and subsequently loads them into our library.

```r
# Installs pacman from CRAN.
if (!require(pacman)) {
    install.packages("pacman")
    require(pacman)
}
# Load pacman into our library.
library(pacman)

# And finally...
pacman::p_load(installr)
```

Why did we load `installr` in the first place?

```r
# Are you using the latest version of R?
check.for.updates.R()
# Download and run the latest R Version.
install.R()
# Copy your packages to the newest R installation
copy.packages.between.libraries()
```

Regression tables

I have fairly strong preferences about how regression tables should look (threeparttable FTW). Luckily, the fantastic **modelsummary** package has us covered for nice looking regression tables, particularly since it automatically supports different Rmd output formats and backends. (For example, via the equally excellent **kableExtra** package.) This makes it easy to produce regression tables that look good in both HTML and PDF... although the latter requires that the corresponding LaTeX packages be loaded first. This template loads those LaTeX packages automatically, so tables like the below Just Work^{TM}.

```r
library(fixest) ## For quick multi-model regression object

mods = feols(c(mpg, hp) ~ disp + csw(wt, drat) | cyl + vs, data = mtcars)

library(modelsummary)
library(kableExtra)

msummary(
  mods,
  title = "fixest: multi-model estimation",
  stars = TRUE,
  gof_omit = "Adj|Pseudo|Log|AIC|BIC"
  ) %>%
  add_footnote(
    c(paste("This footnote is pretty long. In fact, it runs over several lines",
          "of standard PDF output. Luckily that's no problem thanks to",
          "modelsummary, kableExtra, and threeparttable. As an aside, the",
          "fixest package is also amazing and you should use it."),
      "A shorter note."),
    threeparttable = TRUE
    ) %>%
  kable_styling(latex_options = "hold_position") ## (Optional) Print table directly below code
```

Table 1: fixest: multi-model estimation

	mpg	mpg	hp	hp
disp	0.002	0.002	0.104	0.126+
	(0.005)	(0.005)	(0.117)	(0.031)
wt	−3.403	−3.397	−3.502	2.863
	(1.331)	(1.168)	(8.289)	(6.026)
drat		0.038		37.781
		(1.223)		(44.566)
Num.Obs.	32	32	32	32
R2	0.839	0.839	0.721	0.756
R2 Within	0.396	0.396	0.012	0.138
Std.Errors	by: cyl	by: cyl	by: cyl	by: cyl
FE: cyl	X	X	X	X
FE: vs	X	X	X	X

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

[a] This footnote is pretty long. In fact, it runs over several lines of standard PDF output. Luckily that's no problem thanks to modelsummary, kableExtra, and threeparttable. As an aside, the fixest package is also amazing and you should use it.

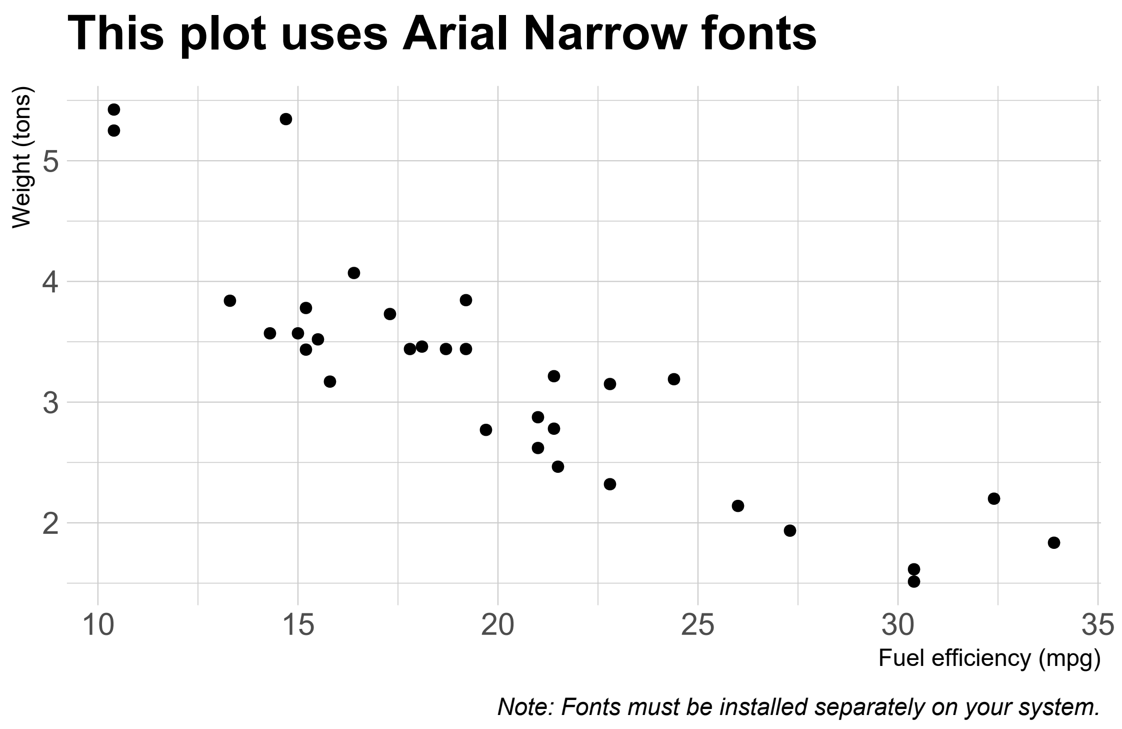
[b] A shorter note.

PDF support for non-standard fonts

This is an easy one; simply a matter of adding `dev: cairo_pdf` to the YAML. But it's nice not having to remember that every time, no?

Note: As the figure caption suggests, to run this next chunk you'll need to add Arial Narrow to your font book if it's not installed on your system already.

```
library(ggplot2)
library(hrbrthemes)

ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(x = "Fuel efficiency (mpg)", y = "Weight (tons)",
       title = "This plot uses Arial Narrow fonts",
       caption = "Note: Fonts must be installed separately on your system.") +
  theme_ipsum()
```



Ignore interactive content when exporting to PDF

In general, this template tries to do a good job of automatically handling (i.e. ignoring) interactive content when exporting to PDF. A notable exception is with embedded interactive content like external GIFs. In this case, rather than typing the usual, say, `![](mind-blown.gif)` directly in the Rmd file, you should include the figure with `knitr::include_graphics` in an R chunk. This will allow you to control whether it renders, conditional on output format. For example, the following chunk will render an actual GIF when the knit target is HTML format, versus a message when that target is PDF format.