**What is an Rprofile**

Every time R starts, it runs through a couple of R scripts. One of these scripts is the .Rprofile. This allows users to customise their particular set-up. However, some care has to be taken, as if this script is broken, this can cause R to break. If this happens, just delete the script!

Full details of how the .Rprofile works can be found in my book with Robin on [Efficient R programming](https://csgillespie.github.io/efficientR/set-up.html#r-startup). However, roughly R will look for a file called .Rprofile first in your current working directory, then in your home area. Crucially, it will only load the first file found. This means you can have per project Rprofile.

**My Rprofile**

A few months ago, I noticed my Rprofile was becoming increasing untidy, so I bundled it up into a single, opinionated, package. This also made it easier for me to switch between computers. Last week, there was an interesting twitter thread on customising your .Rprofile started by [Kara Woo](https://twitter.com/kara_woo/). The thread became popular with lots of great suggestions on neat customisations. This also provided the impetus to write this post.

**Installation**

You can install the package with:

# install.packages("remotes")

library("rprofile")

The package also uses two non-cran packages

# Used for nice prompts

remotes::install\_github("gaborcsardi/prompt")

# Used for nice colours in the terminal; not for Windows

remotes::install\_github("jalvesaq/colorout")

**R Prompt**

You can make simple customisations to your R prompt using options(), but for extra bling I use the the **prompt** package.

* If you are in a Git repo, the branch will be displayed.
* If R’s memory becomes large, the size will also be displayed.

As the RStudio console already has alot of nice features, e.g. syntax highlighting, a distinction needs to be made between the RStudio *Console* and running R in the terminal. So in .Rprofile I’ve got some logic to detect where I’m running R (in my .Rprofile) and adjust accordingly.

**Useful Start-up Messages**

If you use R a lot, you want to minimise noise. I used have the fortunes package display a fortune in my profile, but this got repetitive. Then I tried grabbing stuff from twitter, but this slowed everything down when the wifi was poor.

Currently three start-up messages are displayed:

* The wifi network you are connected too with speed info (Linux only)
* The number of open R sessions (Linux only)
* RStudio project info
* I also clear all the standard R licence stuff from the screen.

If anyone wants to expand the Linux only functions to Windows and Macs, please submit a pull request!

**Helper Functions**

It’s always dangerous to load functions in your start-up script, so I’ve only included functions I’m fairly sure won’t be used in a script.

* create\_make\_functions() – if you have a Makefile in your working directory, this will automatically generate all associated make functions. For example, if you have a force argument in the Makefile this will generate make\_force(). This is actually run at startup.
* lsos() – a handy function for listing large objects.
* library() – Over writes the library() function with a smarter version. If a package is missing, automatically provides the option to install from CRAN or GitHub.
* last\_error() and last\_trace() – pre-loads from **rlang**. Nicer error investigation.

**RStudio Functions**

* op(path = ".") – Creates & opens an RStudio project in the directory specified.
* cp() – Lists previous RStudio projects and gives an option to open.
* inf\_mr() – Short cut to xaringan::inf\_mr().

**Setting Better options()**

The set\_startup\_options() function sets better (in my opinion) set of start-up options. These include

* Setting Ncpus to run parallel installs by default
* Removing significant stars
* Set mc.cores to a sensible default
* Change the continue = "+" to a blank space
* Reduce the default print length
* Plus a few others (see ?set\_startup\_options)

I’ve also created a convenience function for adding additional R repositories – set\_repos(). Probably not needed by most people.

**Example .Rprofile**

Open your .Rprofile, e.g. file.edit("~/.Rprofile") and customise however you want. Here’s an example

if (interactive() && requireNamespace("rprofile", quietly = TRUE)) {

# Only useful if you use Makefiles

rprofile::create\_make\_functions()

# Startup options

rprofile::set\_startup\_options()

# Not RStudio console

if (rprofile::is\_terminal()) {

rprofile::set\_terminal()

} else {

rprofile::set\_rstudio()

}

.env = rprofile::set\_functions()

attach(.env)

# Display wifi and no of R sessions

# Linux only

rprofile::set\_startup\_info()

}

# Prints RStudio project on start-up

setHook("rstudio.sessionInit", function(newSession) {

active\_rproj = rprofile::get\_active\_rproj()

if (!is.null(active\_rproj)) {

message(glue::glue("{crayon::yellow('R-project:')} {active\_rproj}"))

}

}, action = "append")

**Notes and thanks**

* The lsos() function was taken from the [SO](https://stackoverflow.com/q/1358003/203420) question.
* The improved version of library() was adapted from the [autoinst](https://github.com/jimhester/autoinst/). I did think about importing the package, but I had made too many personal tweaks.
* Setting the prompt uses the excellent [prompt](https://github.com/gaborcsardi/prompt) package.
* I’ve probably "borrowed" some of the other ideas from blogposts and SO questions. If I’ve missed crediting you, please let me know and I’ll rectify it.