

A Very Short Tutorial on R*

ECON 293/MGTECON 634

Stanford University

1 Getting Started

You can download and install to download the latest R version for your operating system at the official R project website <https://www.r-project.org/>.

We also suggest that you use **RStudio**, an integrated development environment (IDE) for R that will make programming and data manipulation easier, as well as provide a convenient user interface. You can download a free version of it at <https://www.rstudio.com/products/rstudio/download/>.

2 Installing packages

One of the main advantages of R is that it has many publicly available packages. To check the available packages at a given CRAN repository, type

```
> available.packages()
```

To download R packages from the CRAN repositories, use

```
> install.packages('packageName')
```

You can specify which repository to download the package from. For example,

```
> install.packages('packageName', repos = 'http://cran.us.r-project.org')
```

To use the installed package, type

```
> library(packageName)
```

To remove a package, use the command

*Content courtesy of Thai T. Pham from a previous iteration of the course

```
> remove.packages('packageName')
```

Each package needs to be installed only once, but must be loaded using `library` every time you open R or RStudio.

Installing from github When you need the development version of a package, you may often have to download and install an R packages directly from its github repository. This is possible using the function `install_github` from the `devtools` package.

First, make sure that you have the `devtools` package installed by typing the following on R or RStudio.

```
install.packages('devtools')
```

Then, if you were to install (say) the `ggplot2` package from Hadley Wickham's github webpage at <https://github.com/hadley/ggplot2>, you would type

```
library(devtools)
install_github('hadley/ggplot2')
```

Note that the above procedure works only for public github packages. For private ones, you first need to create a github account at <https://github.com/join>. Then follow the instruction at <https://help.github.com/articles/creating-an-access-token-for-command-line-use/> to generate your Personal Access Token. This token serves to identify yourself without revealing your password in the code. You can then provide your github username and make a request to the owner of a specific private github package. After your request is approved, your access token will work. Now you can use `install_github` the following way

```
> install_github('privateGithubPackage', auth_token = 'yourToken')
```

Sometimes to install private github packages, you need to compile them on your own machine. Instructions are further below.

3 Using Machine Learning Packages

R is a powerful statistical language for writing machine learning programs. However, before executing any program, you need to make sure your dataset has the right format. Some potential problems with the data include missing values (on the outcome variables) or incorrect type of the variables (when we expect a categorical variable, the data gives a numerical one). Make sure you read the data in the correct format. The following links may be helpful:

- <http://www.r-tutor.com/r-introduction/data-frame/data-import>
- <http://www.statmethods.net/input/importingdata.html>

Now assume you can get the data in the correct format in R or RStudio. You then can use a variety of regression and machine learning models such as

- linear regression (use function `lm`),
- logit (function `glm`),
- lasso (function `glmnet`),
- rpart (with cross-validation) (function `rpart`),
- randomForest (function `randomForest`),
- ...

Make sure you have already installed the corresponding packages before using them. To read in detail about how to use any of these (installed) packages, you can type

```
> help(packageName)
```

Note: Sample code together with a dataset are provided along with this tutorial.

4 Formatting The Results

We suggest that you use R Markdown for reporting the results. R Markdown is a file format for making dynamic documents with R. An R Markdown document is written in markdown (an easy-to-write plain text format) and contains chunks of embedded R code. Before using R Markdown, make sure you install the `rmarkdown` package and the `knitr` package. Now you can follow the links below to generate a report in markdown:

- <http://shiny.rstudio.com/articles/rmarkdown.html>
- <https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf>
- <https://rawgit.com/yihui/knitr-examples/master/003-minimal.html>

When you write in R Markdown, you toggle back and forth between text chunks and R code. When you run, you “knit” the file, and it runs in a separate session so that it is reproducible. However, that means that it doesn’t save out intermediate objects, which makes it hard to restart long-running scripts and see what was going on.

You can run chunks of code manually by highlighting them and clicking run, but there are some more advanced tricks detailed here: <https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#debugging-in-r-markdown-documents>

One approach especially for people used to working with command-line debugging is to start with a regular R script and put your text in comments, and then once it runs, turn it into R Markdown and compile a pretty document.



5 For Compiling Packages Locally

This section is for those who need to compile private github packages on their own machines.

Windows You must download and install a software bundle called `Rtools` that will help you transform your code into an actual package that can be downloaded and installed by other users. `Rtools` also helps create documentation, performs certain checks on your R code, and includes an assortment of external functions that you will need to produce cross-platform code. It is available at <http://cran.r-project.org/bin/windows/Rtools/>.

You must download the `Rtools` version that is compatible with your own R version. If you are not sure what version you have, open R up and read the first message that shows up on the screen. If you are a new user and have just installed the latest version of R, just click on the latest version of `Rtools` at the top of the table on the page linked above.

We strongly recommend that you choose the default options, with one exception:

 You must check the box that reads "Add rtools to system PATH" 

Assuming that R has been installed at the default location, once the process is complete you should see a new directory named `C:\Rtools`.

If you chose not to install R at its default location, you might need to move the installed contents to it, or alternatively update your `PATH` manually according to the correct location of `Rtools`. Unfortunately, you will need to adjust not only the path to R binaries, but also to other tools used by R such as your \LaTeX maker. Official instructions for adjusting your `PATH` properly can be found at <https://stat.ethz.ch/R-manual/R-patched/doc/manual/R-admin.html#The-Windows-toolset>, but you might find a gentler explanation over here <https://github.com/stan-dev/rstan/wiki/Install-Rtools-for-Windows>.

Older R versions had a reported bug that made it impossible to install and run **Rtools**. If you have trouble making **Rtools** work in Windows, it might be necessary switch your installation to a patched version of R available here: <https://cran.r-project.org/bin/windows/base/rpatched.html>.

A Fortran compiler is required for some packages. If your PC doesn't have one, you may download one at <https://sourceforge.net/projects/mingw/>. After installation, select the box by **mingw32-ggg-fortran** and mark for installation. Then, select "Apply Changes" under the "Installation" menu. For other options you may look at <https://gcc.gnu.org/wiki/GFortranBinaries#Windows>.

Mac OSX users must install **Xcode command-line tools**. To check if you already have them, open your Terminal App, type

```
$ xcode-select -p
```

and if you have them already you should see this:

```
/Applications/Xcode.app/Contents/Developer
```

Otherwise, type

```
$ xcode-select --install
```

and follow all the instructions on the screen. This process can take over an hour or more to finish.

Some packages might require a Fortran compiler. If your Mac does not have one, you can install **gfortran** via the macOS package manager Homebrew. First, follow the instructions in <https://brew.sh/> to install Homebrew, and then type the following on a Terminal prompt:

```
brew install gcc
```

(Note: the **gcc** above is not a typo. This will install both **gcc** and **gfortran**.)

Linux Most Linux distributions will already have all necessary tools.

Once the steps above are completed, open up R and type

```
library(devtools) # Assuming you have installed devtools already  
find_rtools()
```

If your installation worked properly, you should see **TRUE**. If not, you might want to check further documentation at <https://cran.r-project.org/doc/manuals/R-admin.html>.

As a reminder, it's a good idea to install the latest `devtools` version from Hadley Wickham's github page as explained in section 2. Most modern package developers will also use `roxygen2` for documentation and `testthat` for testing, so we recommend that you go ahead and install them as well.

6 Using R on a Linux server

The following section is largely Stanford-specific. However, many other institutions have similar interfaces.

Opening R remotely Assuming that you are already connected to your server and that R is installed, you can open it up by simply typing

```
> R
```

If RStudio is also installed on the server, you may be able to open it remotely on your own machine via `ssh` with X11 connection forwarding. This is done by opening your Terminal (on a Mac) or Command Prompt (on Windows) and typing:

```
> ssh -X SUNetID@yen.stanford.edu OR
```

```
> ssh -X SUNetID@corn.stanford.edu
```

After logging into the server, use the commands

```
> module load r
```

```
> rstudio
```

Now, you should be on RStudio and can start coding in R.

Running your code When you have very large datasets, it might be a good idea to run R programs on servers instead of your local machine. There are several remarks though.

- Use a screen. This is especially useful if you want to run something interactively, so you can pop in and out of your session.
 - Create a new screen: `screen -S newScreenName`
 - List of your screens on server: `screen -ls`
 - Attached back onto a screen: `screen -r newScreenName`

- Detach from a screen, keep it running: **Control+ a, d**
- While in a screen,
 - (Stanford Specific)
 - Use **krenew** to keep your AFS session alive (batch mode processes). The **/afs** directory uses a token system, so while your job may continue to run, you will lose the ability to read/write to disk (usually within 24 hours).
 - Use the **/farmshare** directory instead as it does not use a token system like **/afs**. Your home directory should be **/farmshare/user_data/yourStanfordSUID**.
- For batch jobs, use **nohup** and **&** when executing a script.
 - **nohup** - regardless of whether you are in a screen session, using **nohup** at the beginning of a command will keep your process running even after you log out.
 - **&** - using the ampersand sign at the end of your command will run your process in the background in the current terminal window and free up your command line
 - Example command: **nohup R CMD BATCH yourScript.R &**

7 Other tutorials on the Web

There are many good tutorials on R floating on the internet. Some of them are

- Wiki site on R: <http://scs.math.yorku.ca/index.php/R>
- Stanford site on R: <http://library.stanford.edu/projects/r>
- Machine Learning on R: <http://trevorstephens.com/post/72916401642/titanic-getting-started-with-r>
- Simple examples on R: http://web.stanford.edu/group/ssds/cgi-bin/drupal/files/Guides/Using%20R%20for%20Windows%20and%20Macintosh_1.pdf
- Another gentle introduction to R: <http://data.princeton.edu/R>