Project Management Using Make

Data Wrangling and Husbandry

03/09/2020

Organization for project management

We have already discussed staying organized in a big project

- Separate folders for
  - data
    - maybe split into raw and cleaned
  - code
  - maybe figures
  - maybe tables (if a big part of a collaborative work)

- R code split into separate files
    - functions
    - data import and cleaning
    - analyses and other computational work
    - generate figures and tables
    - `*.Rmd` or `*.Rnw`

Organization via R Markdown

If the steps are not time consuming, you can stay organized via an R Markdown file. You can include code blocks like

```
source("01-functions.R")
source("02-get-data.R")
source("03-clean-data.R")
```

However, this means that *every* time you run the R Markdown you will run these files. What if those files take a long time to run? You could comment the lines out, but what if you make a change and then forget to uncomment the appropriate line?

(It is possible to use `cache=TRUE` in RMarkdown to manage this; see the documentation)

Aside about saving R objects and figures

- If you do have R objects that took a line time to compute and you want to save them, use the `save()` function, e.g. `save(x, y, file = "xy.RData")` which can be loaded using `load(xy.RData)`
- `ggplot` figures can be saved in an R script file (or in any context) using the ‘`ggsave()` function. The function determines the file type from the filename ending.

`make` and the `makefile`

`make` is a Unix tool that insures that

- all of files are processed, but
- only when needed and that
- each step has the raw material it needs.

You create a `makefile` that

- identifies what output you expect to see
- how to generate that output

Windows users

Although `make` is a Unix tool, you can install it (and other utilities) by installing RTools

You can install RTools directly from a download from that page, or you *may* be able to do it within RStudio with the command

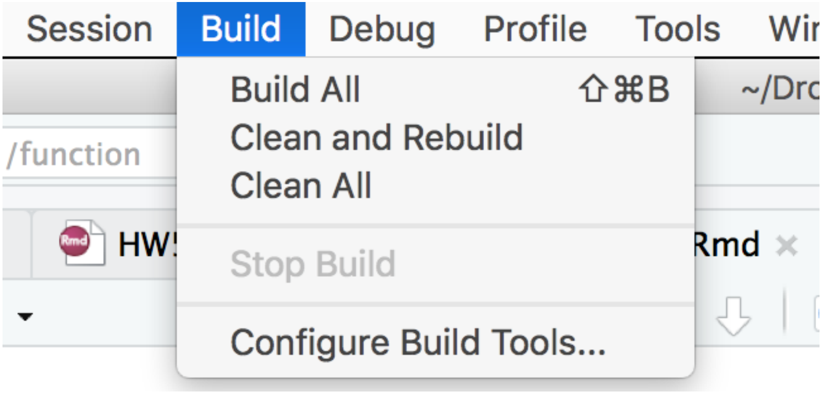
```
.rs.installBuildTools(1)
```

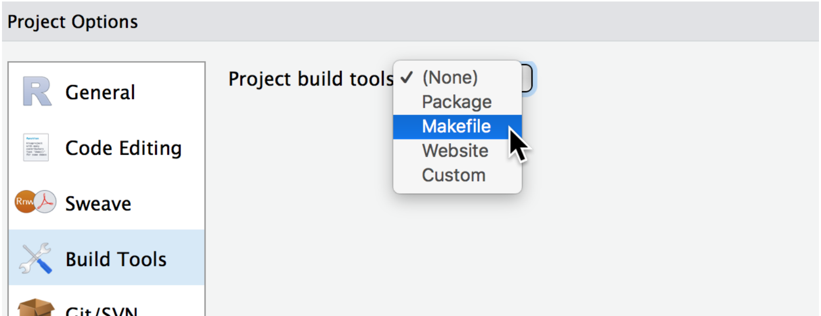
A simple example

Suppose we have a file `presentation.Rmd` which requires two files, `fig1.png` and `fig1.png`, both in the `figs` folder. We would like to

- convert `presentation.Rmd` to `presentation.html`
- using the two figures, but recreating the figures if those files are older than the files that created them

With the proper `makefile`, this can be done in a single command: `make`. You would run that from the command line in the same folder as the file `makefile`, which should also be the same folder as the other parts of your project. In RStudio, you can also do that from the "Build" menu.


Session
Build
Debug
Profile
Tools
Wi
Build All ⇧⌘B
Clean and Rebuild
Clean All
Stop Build
Configure Build Tools...
~/Dr
/function
HW
Rmd


Project Options
General
Code Editing
Sweave
Build Tools
Git/SVN
Project build tools
✓ (None)
Package
Makefile
Website
Custom

A simple make file

```
presentation.html: presentation.Rmd  figs/fig1.png figs/fig
  R -e "rmarkdown::render('presentation.Rmd')"

figs/fig1.png: code/fig1.R
  R CMD BATCH code/fig1.R

figs/fig2.png: code/fig2.R
  R CMD BATCH code/fig2.R
```

Each group of lines gives the file to be created (known as the target), the files it depends on, and the commands to construct the target from the required files. There can be multiple lines for the commands, but each must start with a tab, not spaces.

In this example, `fig1.png` is only built if `fig1.R` changes. In turn, if `fig1.R` changes, then `fig1.png` will change, so `presentation.html` will be re-built.

Variables in `make`

You can create variables in `make` by defining them along the lines of

```
R_OPTS =--vanilla
```

which you would then use with a command like

```
R CMD BATCH $(R_OPTS) code/fig1.R
```

Patterns in `make`

`make` allows for all sorts of patterns to make the `makefile` more compact and easier to maintain and extend. For simple projects you don't need them, but you'll probably see them if you look at examples.

Handling multiple *.R files

In the example above, what if `fig1.R` and `fig2.R` depend on `01-functions.R`? How will `make` know if I've made a change to that file?

There are two approaches.

`1.` When your run R in batch mode (`R CMD BATCH example.R`), it will produce a file with the same name as the script file but with out appended to it: `example.Rout`. That means we can extend the line to

```
figs/fig1.png: code/fig1.R code/01-functions.Rout
```

and include the lines

```
code/01-functions.Rout: code/01-functions.R
  R CMD BATCH $(R_OPTS) code/01-functions.R
```

2. The other approach works only for files that produce things. For example, `02-get-data.R` might download the file `data/bigfile.csv` and `03-clean-data.R` might produce the file `workfiles/clean-data.RData`

```
data/bigfile.csv: 02-get-data.R
    R CMD BATCH $(R_OPTS) code/02-get-data.R

workfiles/clean-data.RData: 03-clean-data.R data/bigfile.cs
    R CMD BATCH $(R_OPTS) 03-clean-data.R
```

A larger example

```
R_OPTS =--vanilla

presentation.html: presentation.Rmd  figs/fig1.png figs/fi
  R -e "rmarkdown::render('presentation.Rmd')"

code/01-functions.Rout: code/01-functions.R code/01-functi
  R CMD BATCH $(R_OPTS) code/01-functions.R

data/bigfile.csv: 02-get-data.R code/01-functions.Rout
    R CMD BATCH $(R_OPTS) code/02-get-data.R

workfiles/clean-data.RData: 03-clean-data.R data/bigfile.c
    R CMD BATCH $(R_OPTS) 03-clean-data.R

figs/fig1.png: code/fig1.R code/01-functions.Rout workfile
  R CMD BATCH $(R_OPTS) code/fig1.R

figs/fig2.png: code/fig2.R workfiles/clean-data.RData
```

`make` and RStudio

You need to have already created a project, but with that project open, just change the “Project Options” section of the “Tools/Project Options”" in RStudio so that the build tool moves from “none” to “Makefile”.

Documentation

`make` is used heavily by programmers, has been around a long time, and has many options. I find the standard documentation opaque.

I suggest Karl Broman's minimal make page as well as the pages from Zachary Jones and Shaun Jackson and Jenny Bryan