gnumaker

Version: 0.0.0.9002

Overview

gnumaker makes if easy to create and use GNU Makefiles to aid a reproducible work flow for data analysis projects.

GNU Make is the defacto standard for efficiently rerunning appropriate steps in the data analysis or reporting process if a particular file is changed. Only the necessary steps are rerun.

Rather than creating a new system for setting up and building output from statistical software syntax files, **gnumaker** leverages off existing GNU Make rules. These rules, for R, Sweave, R Markdown, Stata, SAS and other syntax files are available at r-makefile-definitions on Github. These are described in P Baker (2019) Using GNU Make to Manage the Workflow of Data Analysis Projects, *Journal of Statistical Software (Accepted)*.

For those not familiar with GNU Make, **gnumaker** allows simple dependencies between files to be specified to produce a working Makefile and the associated directed acyclic graph (DAG). I'd welcome Github issues containing error reports or feature requests. Alternatively, you can email the package maintainer at drpetebaker at gmail dot com.

Installation

You can install the development version of **gnumaker** from GitHub with:

```
## if you don't have devtools installed, first run:
## install.packages("devtools")
devtools::install_github("petebaker/gnumaker")
```

Usage

There are three key functions in **gnumaker**. These are:

- `create_makefile()` creates a gnu_makefile object given dependencies between syntax, data and output files,
- `write_makefile()` writes a Makfile to disk.
- `plot()` plots a DAG for a gnu_makefile object and

Examples

Suppose we have a data file `simple.csv` and use `read.R` to read and clean the data. After storing the cleaned data in a .RData file, we then employ `linmod.R` to plot and analyse the data. Next, using the stored results two reports, `report1.pdf` and `report2.docx` are produced from `report1.Rmd` and `report2.Rmd`. The workflow may be encapsulated in a Makefile which is then employed to manage the process and generate or regenerate any intermediate files when the data or syntax changes.

Here we assume the default target file extension for running an *R* file is *Rout* (default) but instead specify the output types for the *.Rmd* files targets 'read' depends on *read.r* and *simple.csv*, 'linmod' on *linmod.R* and 'read' etc. Note that we can change the default target file extension using the 'default.exts' argument and specify say a HTML target file with *default.exts = list(R = "html")*.

```
library(gnumaker)
gm1 <-
  create_makefile(targets = list(read = c("read.R", "simple.csv"),
                                 linmod = c("linmod.R", "read"),
                                 rep1 = c("report1.Rmd", "linmod"),
                                 rep2 = c("report2.Rmd", "linmod")),
    comments = list(linmod = "plots and analysis using 'linmod.R'"),
                    file.exts = list(rep1 = "pdf", rep2 = "docx"))
```

A Makefile `Makefile.demo` is produced with `write_makefile(gm1)`

```
write_makefile(gm1, file = "Makefile.demo")
#> File: Makefile.demo written at Tue Apr  2 13:17:34 2019
```

```
cat Makefile.demo
# File: Makefile.demo
# Created at: Tue Apr  2 13:17:34 2019

# Produced by gnumaker:  0.0.0.9002 on R version 3.5.3 (2019-03-11)
# Before running make, please check file and edit if necessary

# .PHONY all target which is run when make is invoked
.PHONY: all
all: report1.pdf report2.docx

# report1.pdf depends on report1.Rmd, linmod.Rout
report1.pdf: report1.Rmd linmod.Rout

# report2.docx depends on report2.Rmd, linmod.Rout
report2.docx: report2.Rmd linmod.Rout

# plots and analysis using 'linmod.R'
linmod.Rout: linmod.R read.Rout

# read.Rout depends on read.R, simple.csv
read.Rout: read.R simple.csv

# include GNU Makfile rules. Most recent version available at
# https://github.com/petebaker/r-makefile-definitions
include~/lib/r-rules.mk

# remove all target, output and extraneous files
.PHONY: cleanall
cleanall:
    rm -f *~ *.Rout *.RData *.docx *.pdf *.html *-syntax.R *.RData
```

The DAG of the `gnu_makefile` object can be produced with `plot(gm1)`.

```
plot(gm1)
```

For more examples, see the gnumaker vignette (under construction).

Notes

gnumaker is under construction and should change (and improve) rapidly over the next few months.

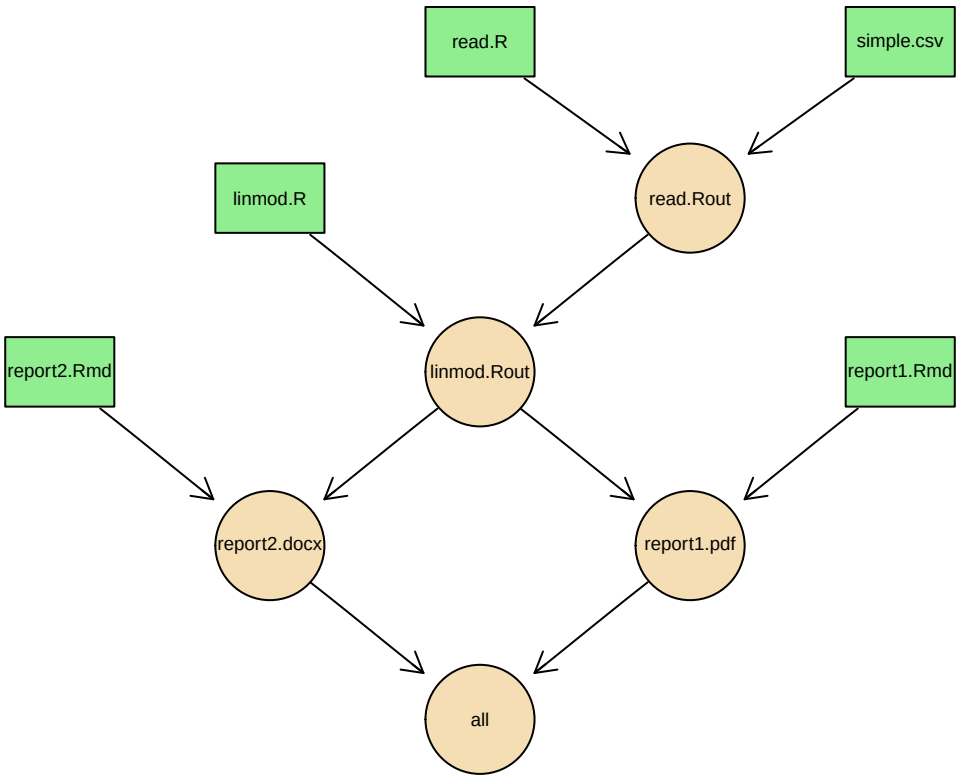

Figure 1: DAG of Makefile for simple example. The DAG of the `gnu_makefile` object can be produced with `plot(gm1)`. Using the minimal set of files (shown in green rectangles), then GNU Make allows us to (re)generate all other files shown as wheat coloured circles)

To do

- add all dependency and target file extensions in `r-rules.mk`, preferably by parsing the included file
- add `testthat` unit testing for more complicated examples
- allow for target file extensions and dependency files to be set as user specified variables which would make the `Makefile`s produced more flexible but less easy to read
- allow specification of *global options* in `zzz.R` so that it is easier to customise defaults
- either incorporate `makefile2graph` as a way of plotting `Makefile`s not made with **gnumaker** or write own functions. (See https://github.com/lindenb/makefile2graph)