6: GNU Make

bit.ly/2018rr

GNU Make

- Automation the full project
- Document dependencies
- Only re-run things that need to be re-run

Automate the process (GNU Make)

```
R/analysis.html: R/analysis.Rmd Data/cleandata.csv
    cd R;R -e "rmarkdown::render('analysis.Rmd')"

Data/cleandata.csv: R/prepData.R RawData/rawdata.csv
    cd R;R CMD BATCH prepData.R

RawData/rawdata.csv: Python/xls2csv.py RawData/rawdata.xls
    Python/xls2csv.py RawData/rawdata.xls > RawData/rawdata.csv
```

Automate the process (GNU Make)

```
R/analysis.html: R/analysis.Rmd Data/cleandata.csv
    cd R;R -e "rmarkdown::render('analysis.Rmd')"

Data/cleandata.csv: R/prepData.R RawData/rawdata.csv
    cd R;R CMD BATCH prepData.R

RawData/rawdata.csv: Python/xls2csv.py RawData/rawdata.xls
    Python/xls2csv.py RawData/rawdata.xls > RawData/rawdata.csv
```

Automate the process (GNU Make)

```
R/analysis.html: R/analysis.Rmd Data/cleandata.csv
    cd R;R -e "rmarkdown::render('analysis.Rmd')"

Data/cleandata.csv: R/prepData.R RawData/rawdata.csv
    cd R;R CMD BATCH prepData.R

RawData/rawdata.csv: Python/xls2csv.py RawData/rawdata.xls
    Python/xls2csv.py RawData/rawdata.xls > RawData/rawdata.csv
```

Automate the process (GNU Make)

```
R/analysis.html: R/analysis.Rmd Data/cleandata.csv
    cd R;R -e "rmarkdown::render('analysis.Rmd')"

Data/cleandata.csv: R/prepData.R RawData/rawdata.csv
    cd R;R CMD BATCH prepData.R

RawData/rawdata.csv: Python/xls2csv.py RawData/rawdata.xls
    Python/xls2csv.py RawData/rawdata.xls > RawData/rawdata.csv
```

Automation with GNU Make

- `Make` is for more than just compiling software
- The essence of what we're trying to do
- Automates a workflow
- Documents the workflow
- Documents the dependencies among data files, code
- Re-runs only the necessary code, based on what has changed

Fancier example

```
FIG_DIR = Figs

mypaper.pdf: mypaper.tex $(FIG_DIR)/fig1.pdf $(FIG_DIR)/fig2.pdf
    pdflatex mypaper

# One line for both figures
$(FIG_DIR)/%.pdf: R/%.R
    cd R;R CMD BATCH $(<F)

# Use "make clean" to remove the PDFs
clean:
    rm *.pdf Figs/*.pdf
```

Installing Make

- On Macs, Make should be installed. Type "`make --version`" to check.
- On Windows, probably the easiest is to install Rtools, which includes Make.

  `cran.r-project.org/bin/windows/Rtools`

How do you use Make?

- If you name your make file `Makefile`, then just go into the directory containing that file and type `make`
- If you name your make file `something.else`, then type `make -f something.else`
- Actually, the commands above will build the first target listed in the make file. So I'll often include something like the following.

  ```
  all: target1 target2 target3
  ```

  Then typing `make all` (or just `make`, if `all` is listed first in the file) will build all of those things.
- To be build a specific target, type `make target`. For example, `make Figs/fig1.pdf`

Make with R Markdown

To use Make with R Markdown, you'll use a command like:

```
R -e "rmarkdown::render('my_report.Rmd')"
```

You'll need to tell your operating system where it can find pandoc. RStudio includes pandoc, but you need to add the relevant directory to your `PATH`.

Mac:

```
/Applications/RStudio.app/Contents/MacOS/pandoc
```

Windows:

```
"c:\Program Files\RStudio\bin\pandoc"
```

Common challenge

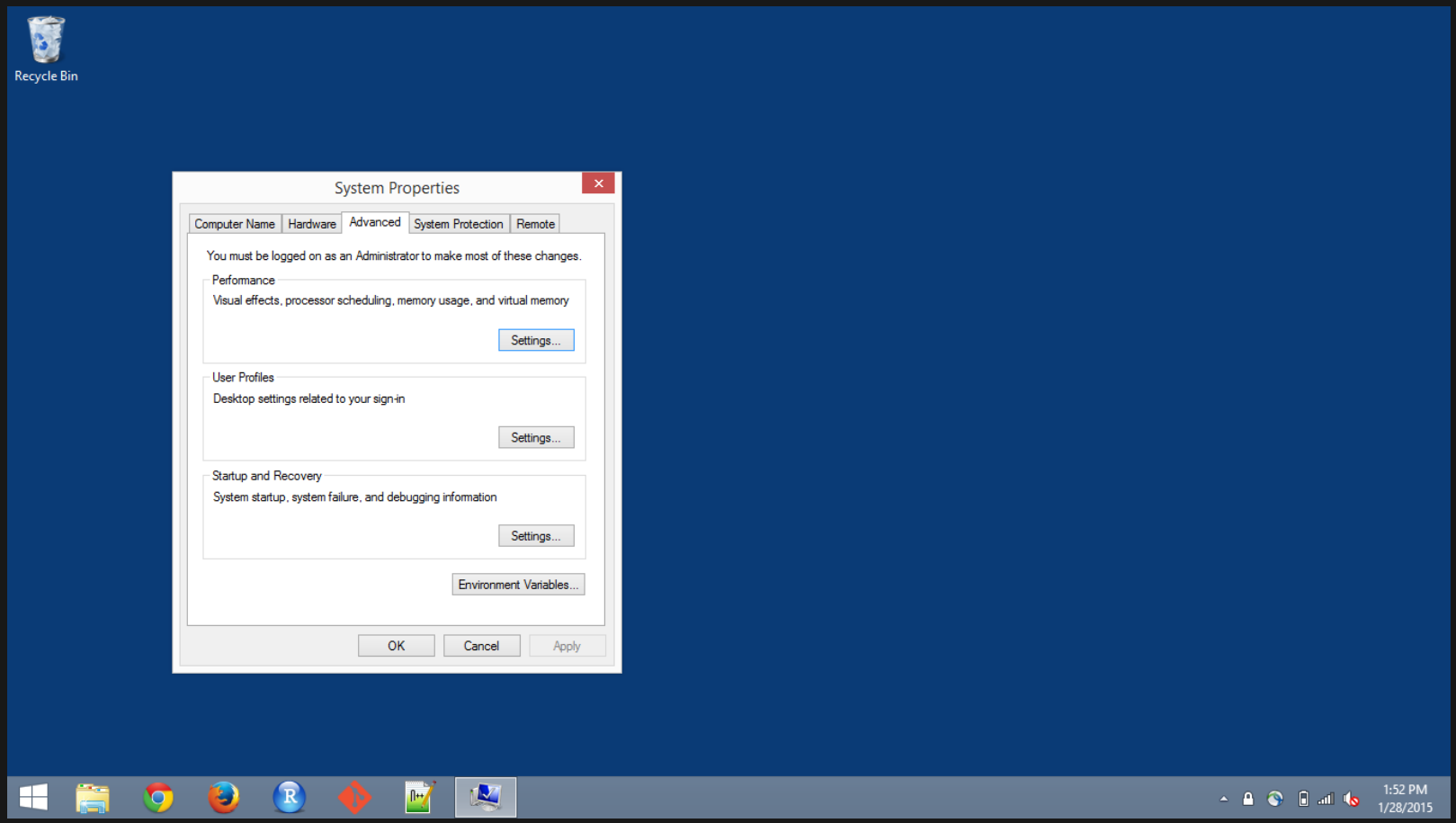
PATH

~/.bash_profile

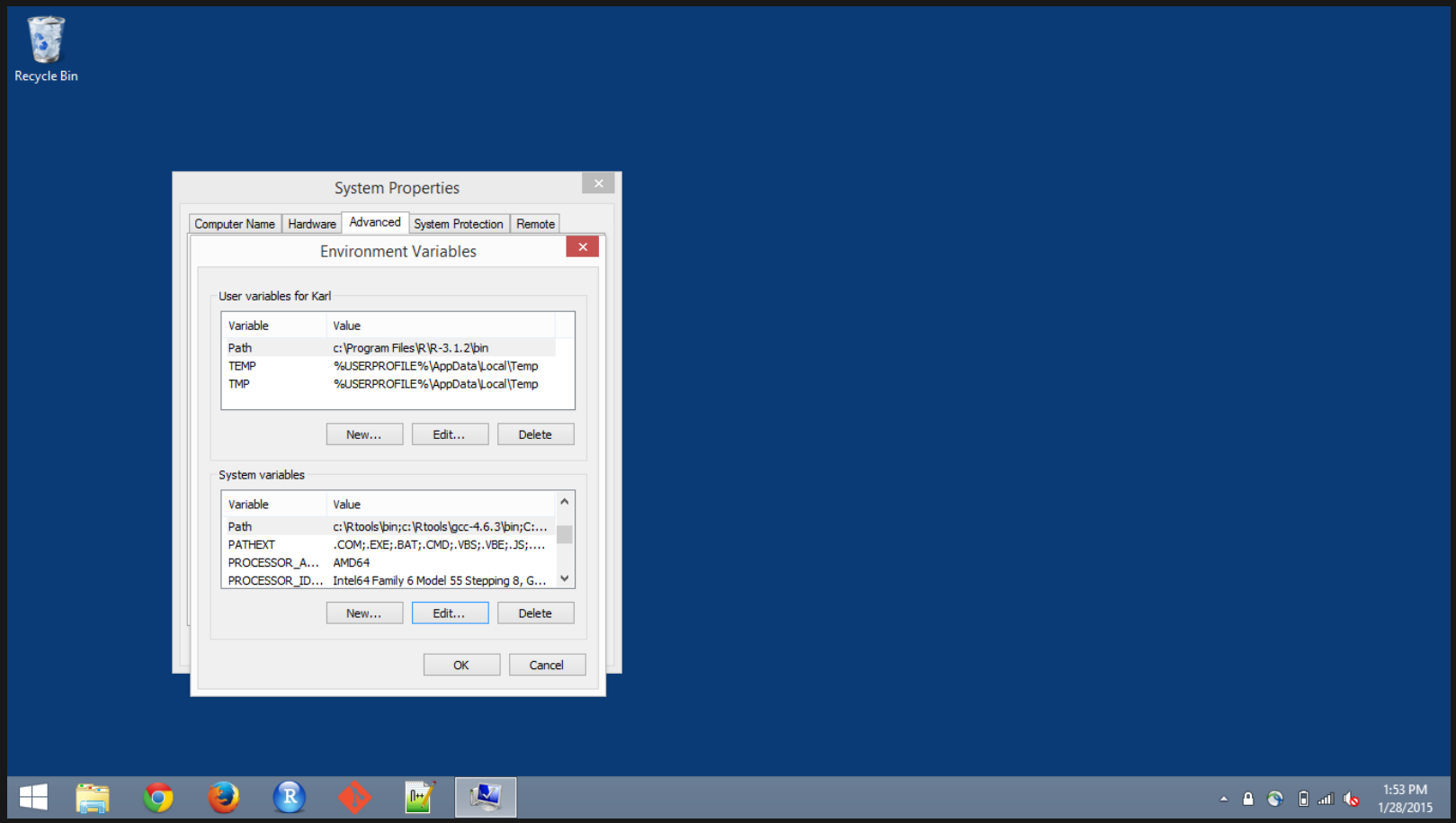
```
export PATH=$PATH:/Applications/RStudio.app/Contents/MacOS/pandoc

noclobber=1    # prevent overwriting of files
IGNOREEOF=1    # disable Ctrl-D as a way to exit
HISTCONTROL=ignoredups

alias cl='clear;cd'
alias rm='rm -i'
alias mv='mv -i'
alias cp='cp -i'
alias ls='ls -GF'
alias 'l.'='ls -d .[a-zA-Z]*'
alias ll='ls -lh'
alias md='mkdir'
alias rd='rmdir'
alias rmb='rm .*~ *~ *.bak *.bk!'

alias Rb='R CMD build --force --resave-data'
alias Ri='R CMD INSTALL --library=/Users/kbroman/Rlibs'
alias Rc='R CMD check --library=/Users/kbroman/Rlibs'
alias Rcc='R CMD check --as-cran --library=/Users/kbroman/Rlibs'
```

PATH in Windows

PATH in Windows



PATH in Windows



Variables

- Define a variable like
  `R_OPTS=--vanilla`
- Use it with a `$` and `()` or `{}`, for example:
  `R CMD BATCH $(R_OPTS) fig1.R`

Automatic variables

There are a bunch of automatic variables that you can use to save yourself a lot of typing.

Here are the ones I use most:

`$@`	the file name of the target
`$<`	the name of the first dependency
`$^`	the names of all dependencys
`$(@D)`	the directory part of the target
`$(@F)`	the file part of the target
`$(<D)`	the directory part of the first dependency
`$(<F)`	the file part of the first dependency

Pattern rules

Pattern rules are like wildcards for file names: if a bunch of files are to be built the same way, you can use the symbol `%` as a wildcard.

For example, if you have two figures `fig1.pdf` and `fig2.pdf` that are to be built by `fig1.R` and `fig2.R`, respectively, you might do:

```
Figs/%.pdf: R/%.R
    cd $(<D);R CMD BATCH $(<F)
```

The two figures' file names will need to be be spelled out somewhere, for example as dependencies.

Resources

- kbroman.org/minimal_make
- bost.ocks.org/mike/make
- robjhyndman.com/hyndsight/makefiles
- Search github with `filename:Makefile`
  - `R CMD BATCH filename:Makefile`
  - `filename:Makefile user:yihui`

Activity

Go back to your R Markdown documents from this morning.

- Write a `Makefile` to produce different types of outputs from your various Rmd files.
- Add `make all` and `make clean` as targets
- What happens if you run `make all` twice?