Simple

tips, tricks & tools

A brief history of R & R

- 1976: S language developed at Bell Laboratories.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.
- 1996: I heard Ross Ihaka give a talk about R at a statistics conference.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.
- 1996: I heard Ross Ihaka give a talk about R at a statistics conference.
- 1997: CRAN began with 12 packages.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.
- 1996: I heard Ross Ihaka give a talk about R at a statistics conference.
- 1997: CRAN began with 12 packages.
- 2000: R 1.0.0 released.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.
- 1996: I heard Ross Ihaka give a talk about R at a statistics conference.
- 1997: CRAN began with 12 packages.
- 2000: R 1.0.0 released.
- 2001: I stopped using S-PLUS and switch to R.

A brief history of R & R

- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.
- 1996: I heard Ross Ihaka give a talk about R at a statistics conference.
- 1997: CRAN began with 12 packages.
- 2000: R 1.0.0 released.
- 2001: I stopped using S-PLUS and switch to R.
- 2004: I contributed my first function to R (`quantile`).

A brief history of R & R

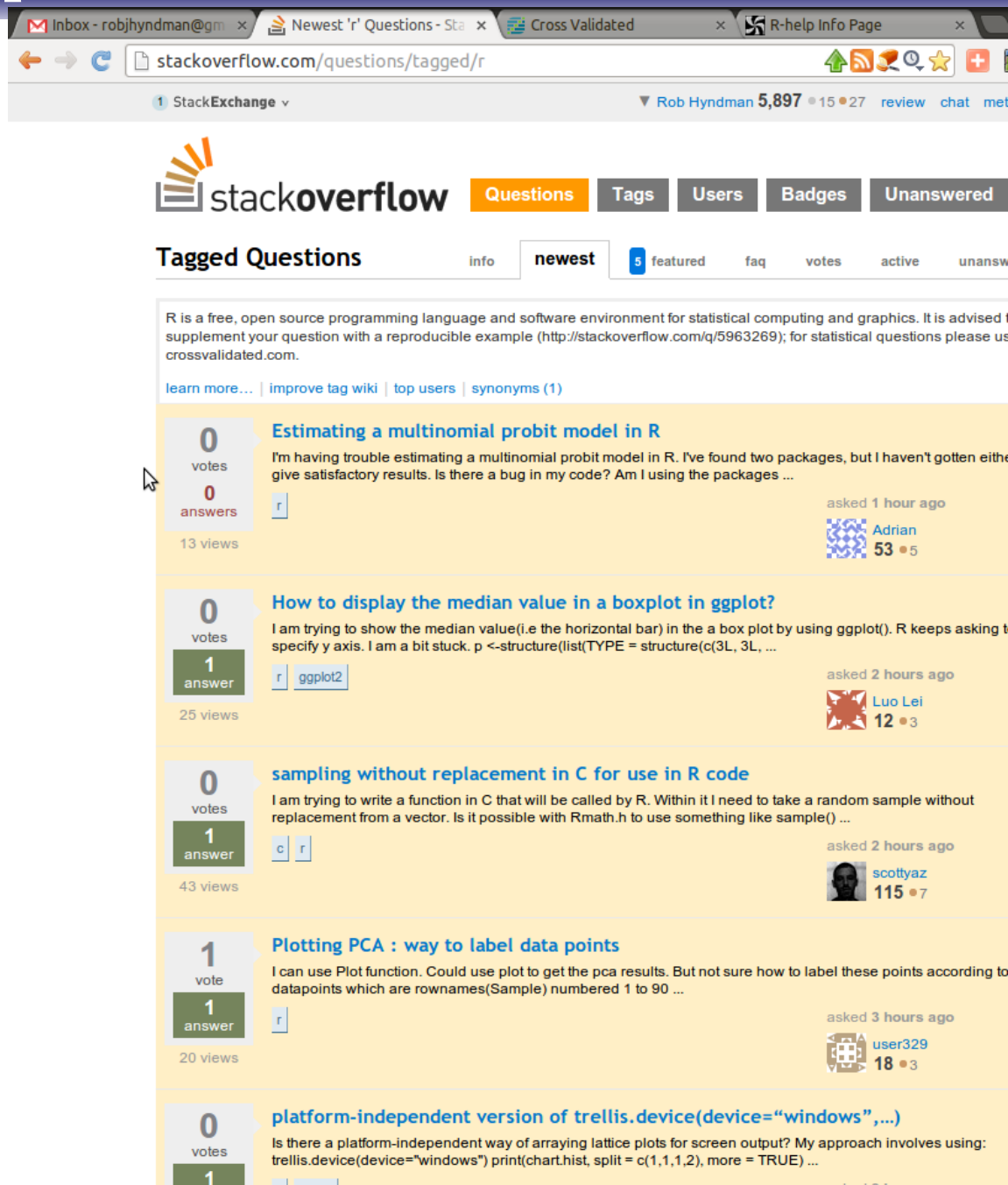
- 1976: S language developed at Bell Laboratories.
- 1980: S first used outside Bell.
- 1987: I first used an implementation of S (called Ace) distributed by CSIRO.
- 1988: S-PLUS produced, and I start using it.
- 1996: I heard Ross Ihaka give a talk about R at a statistics conference.
- 1997: CRAN began with 12 packages.
- 2000: R 1.0.0 released.
- 2001: I stopped using S-PLUS and switch to R.
- 2004: I contributed my first function to R (`quantile`).
- 2006: I contributed my first package to CRAN (**forecast**).

Outline

Getting help

StackOverflow.com

- For *programming* questions.



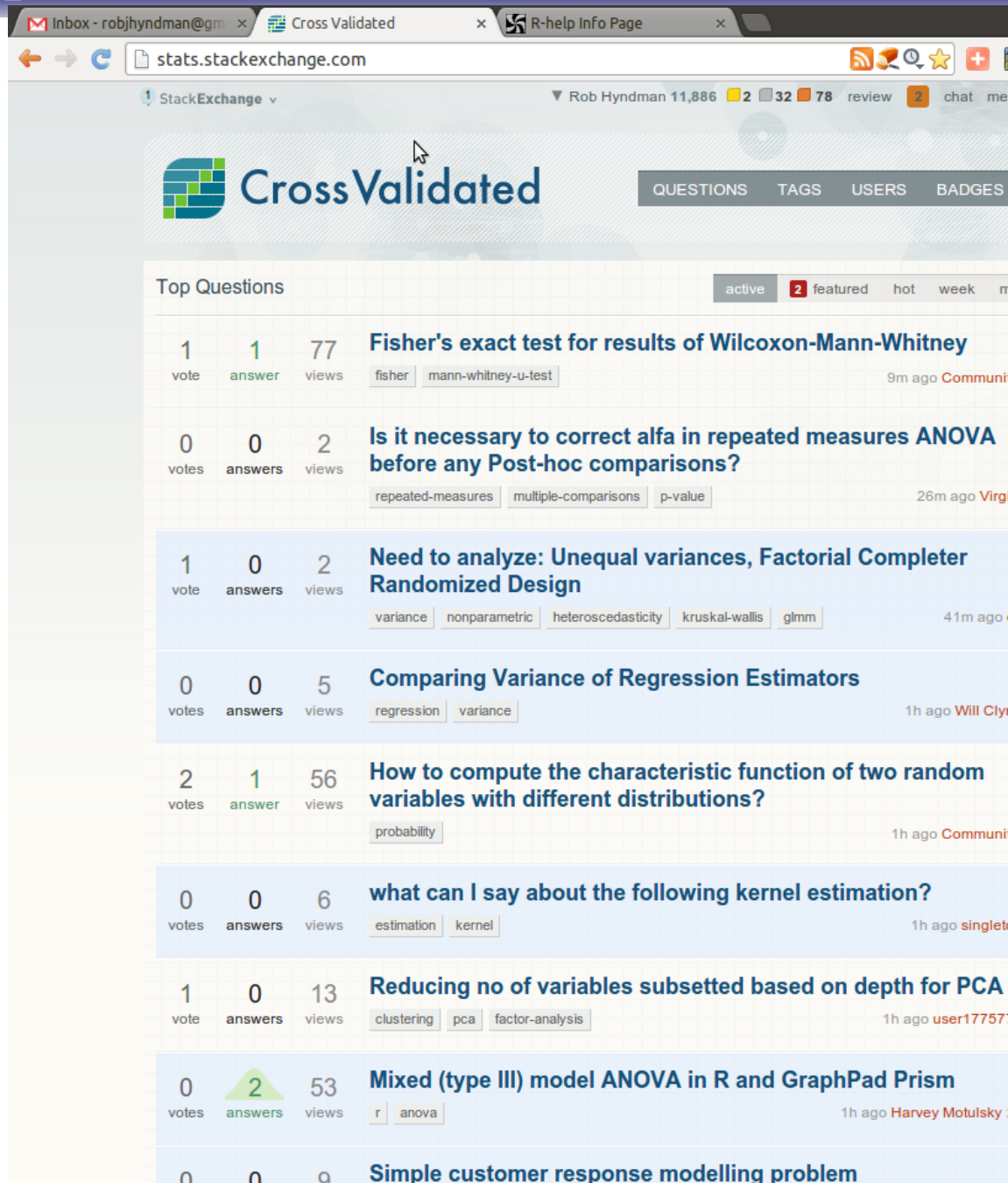
Getting help

StackOverflow.com

- For *programming* questions.

CrossValidated.com

- For *statistical* questions.



Getting help

StackOverflow.com

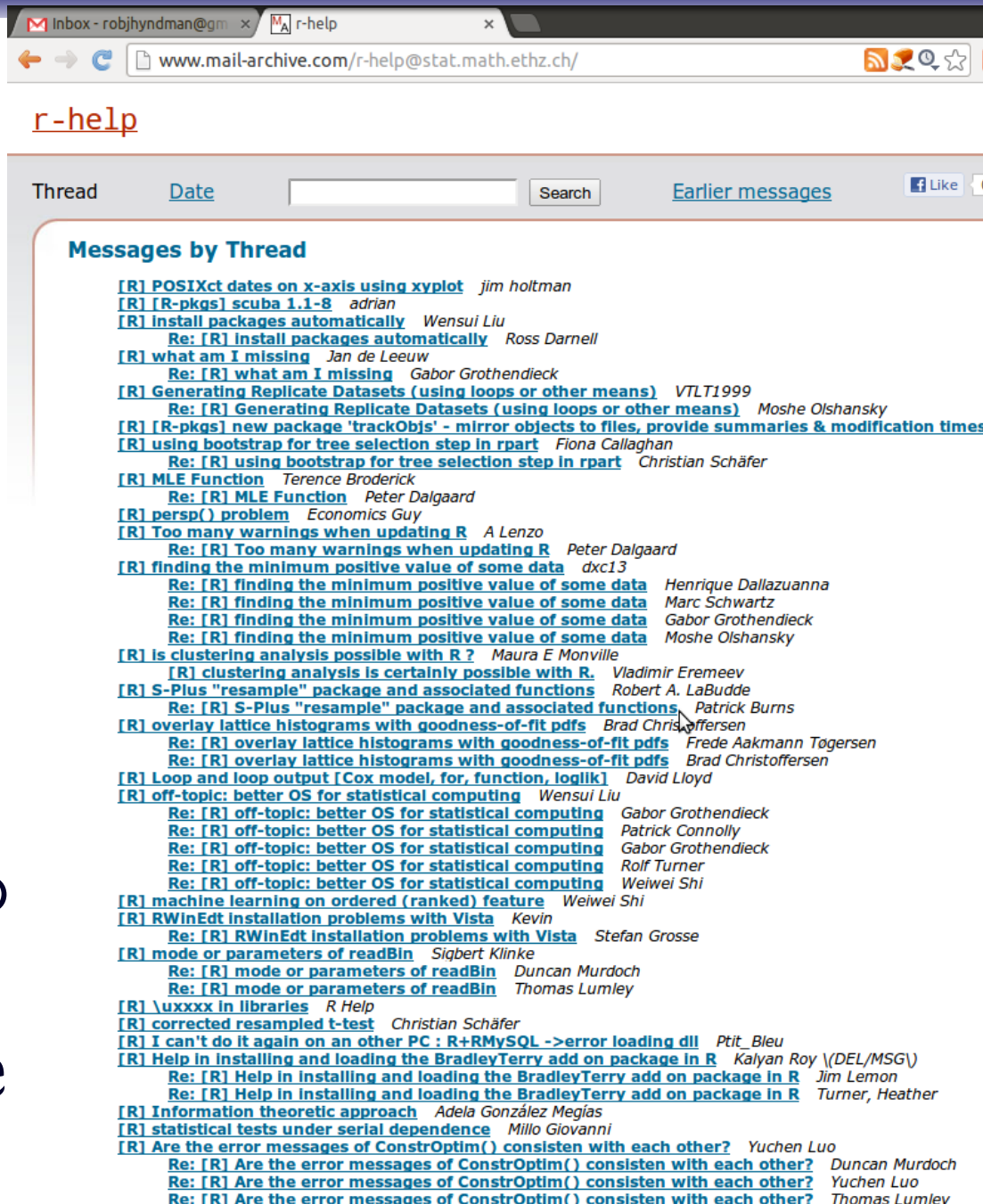
- For *programming* questions.

CrossValidated.com

- For *statistical* questions.

R-help mailing lists

- `stat.ethz.ch/mailman/listinfo/r-help`
- Only when all-else fails.



Outline

How to find the right function

Functions in installed packages

`help.search("neural")`. Equivalently: `??neural`
Also built into RStudio help.

How to find the right function

Functions in installed packages

`help.search("neural")`. Equivalently: `??neural`
Also built into RStudio help.

Functions in other CRAN packages

```
library(sos)
findFn("neural")
RSiteSearch("neural")
```

- `findFn` only searches functions.

How to find the right function

Functions in installed packages

`help.search("neural")`. Equivalently: `??neural`
Also built into RStudio help.

Functions in other CRAN packages

```
library(sos)
findFn("neural")
RSiteSearch("neural")
```

- `findFn` only searches functions.
- `RSiteSearch` searches more widely.

How to find the right function

Functions in installed packages

`help.search("neural")`. Equivalently: `??neural`
Also built into RStudio help.

Functions in other CRAN packages

```
library(sos)
findFn("neural")
RSiteSearch("neural")
```

- `findFn` only searches functions.
- `RSiteSearch` searches more widely.

How to find the right function

Functions in installed packages

`help.search("neural")`. Equivalently: `??neural`
Also built into RStudio help.

Functions in other CRAN packages

```
library(sos)
findFn("neural")
RSiteSearch("neural")
```

- `findFn` only searches functions.
- `RSiteSearch` searches more widely.

rseek.org

- Google customized search on R-related sites.

CRAN Task Views

`cran.r-project.org/web/views/`

- Curated reviews of packages by subject

CRAN Task Views

`cran.r-project.org/web/views/`

- Curated reviews of packages by subject
- Use `install.views()` and `update.views()` in the **ctv** package.

Outline

Digging into functions

Example: How does forecast for ets work?

```
forecast
forecast.ets
forecast:::pegelsfcast.C
```

Digging into functions

Example: How does forecast for ets work?

```
forecast
forecast.ets
forecast:::pegelsfcast.C
```

- Typing the name of a function gives its definition.

Digging into functions

Example: How does forecast for ets work?

```
forecast
forecast.ets
forecast:::pegelsfcast.C
```

- Typing the name of a function gives its definition.
- Be aware of classes and methods.

Digging into functions

Example: How does forecast for ets work?

```
forecast
forecast.ets
forecast:::pegelsfcast.C
```

- Typing the name of a function gives its definition.
- Be aware of classes and methods.
- Type `package:::function` for hidden functions.

Digging into functions

Example: How does forecast for ets work?

```
forecast
forecast.ets
forecast:::pegelsfcast.C
```

- Typing the name of a function gives its definition.
- Be aware of classes and methods.
- Type `package:::function` for hidden functions.
- Download the `tar.gz` file from CRAN if you want to see any underlying C or Fortran code.

Outline

Good habits

- indenting and commenting.

Good habits

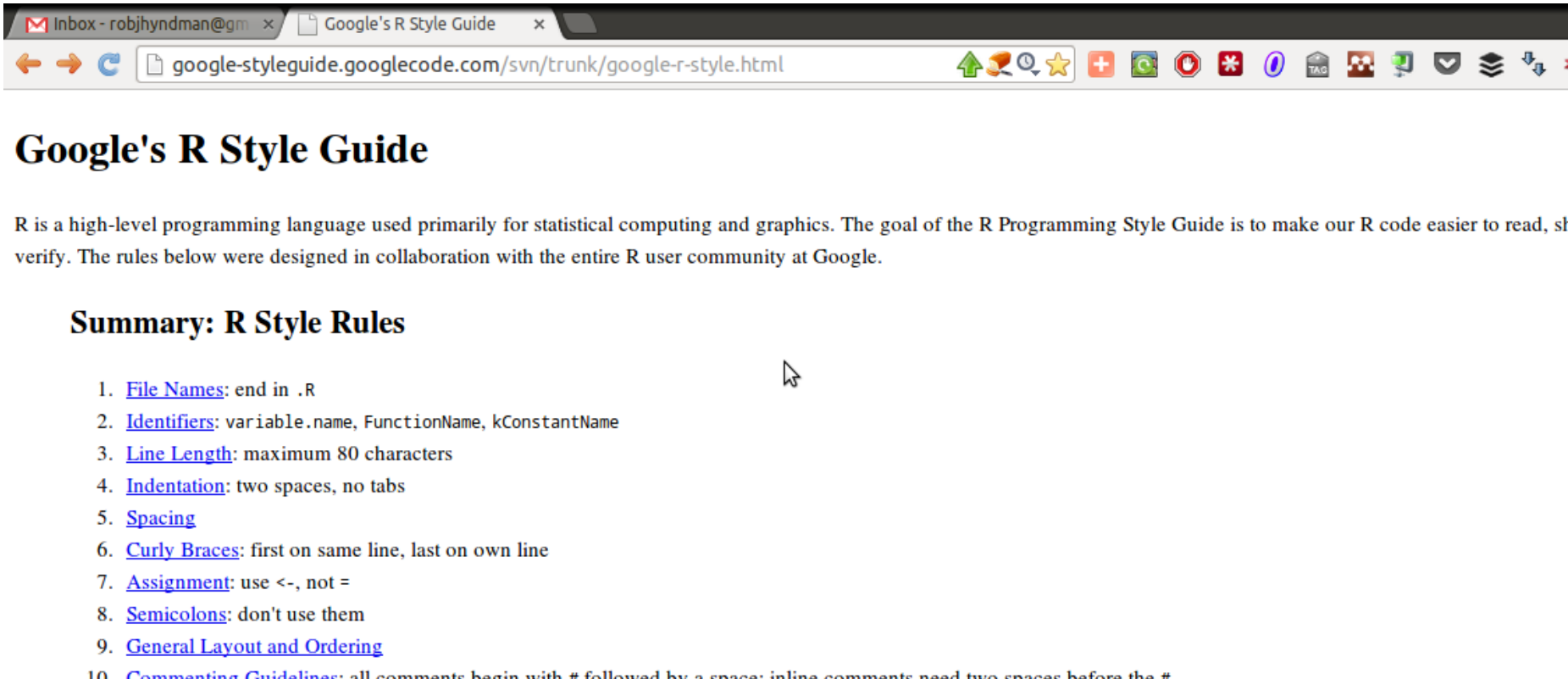
- indenting and commenting.
- Reindenting using RStudio.

Good habits

- indenting and commenting.
- Reindenting using RStudio.
- **formatR** package: I run `tidy.dir()` before reading student code!

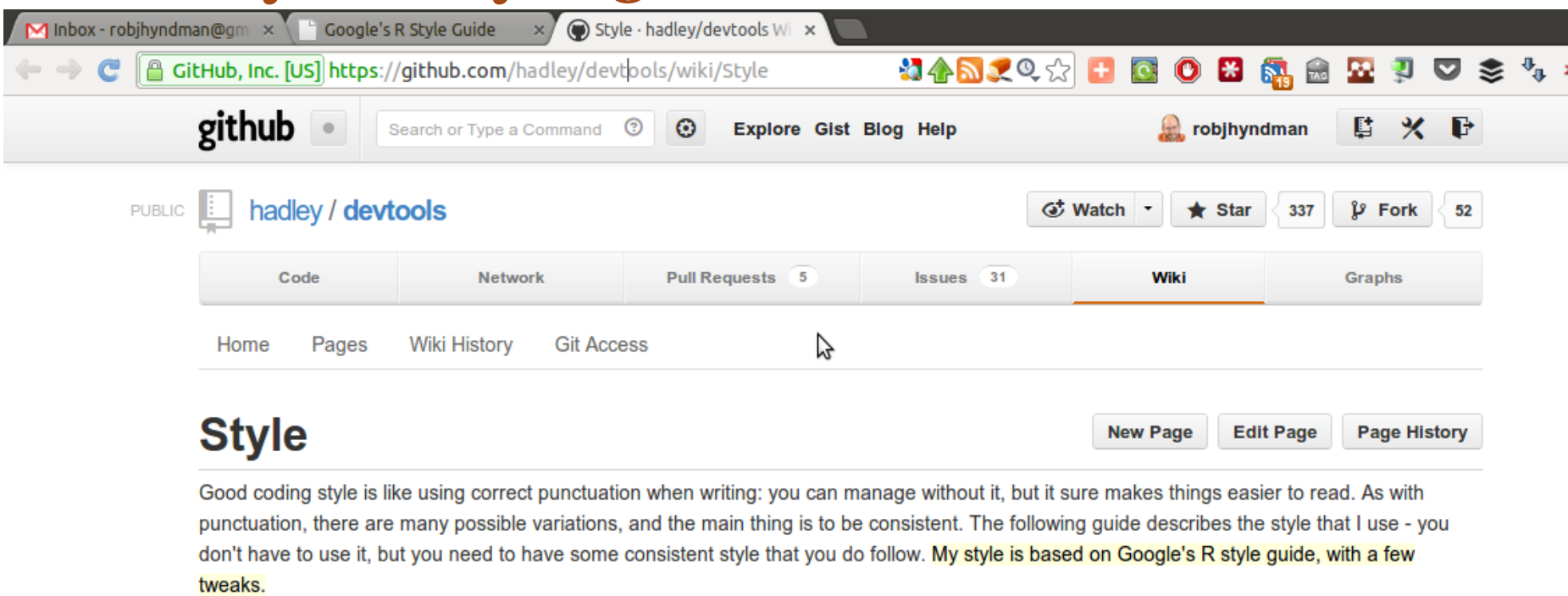
Good habits

- indenting and commenting.
- Reindenting using RStudio.
- **formatR** package: I run `tidy.dir()` before reading student code!
- Google R style guide:



Good habits

- indenting and commenting.
- Reindenting using RStudio.
- **formatR** package: I run `tidy.dir()` before reading student code!
- Google R style guide:
- Hadley's R style guide:



Outline

Simple debugging tools

When something goes wrong:

- `traceback()`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`
- `debug()`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`
- `debug()`
- `browser()`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`
- `debug()`
- `browser()`
- `trace()`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`
- `debug()`
- `browser()`
- `trace()`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`
- `debug()`
- `browser()`
- `trace()`

More extensive debugging tools discussed at

`www.stats.uwo.ca/faculty/murdoch/software/debuggingR/`

Simple debugging tools

When something goes wrong:

- `traceback()`
- `options(error=recover)`
- `debug()`
- `browser()`
- `trace()`

More extensive debugging tools discussed at

`www.stats.uwo.ca/faculty/murdoch/software/debuggingR/`

RStudio plans to have debugging tools in a future release.

Outline

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).
- But no built-in diff facilities or changelog.

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).
- But no built-in diff facilities or changelog.

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).
- But no built-in diff facilities or changelog.

Git

- Free version control system.

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).
- But no built-in diff facilities or changelog.

Git

- Free version control system.
- Easy to fork or roll back.

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).
- But no built-in diff facilities or changelog.

Git

- Free version control system.
- Easy to fork or roll back.
- Install git software on your local computer.

Version control

In any non-trivial project, it is important to keep track of versions of R code and documents.

Dropbox

- Every version of your files for the last 30 days (or longer if you buy Packrat).
- But no built-in diff facilities or changelog.

Git

- Free version control system.
- Easy to fork or roll back.
- Install git software on your local computer.
- Manage via RStudio.

Outline

R projects

Basic idea

- Every paper, book or consulting report is a "project".

R projects

Basic idea

- Every paper, book or consulting report is a "project".
- Every project has its own folder and R workspace.

R projects

Basic idea

- Every paper, book or consulting report is a "project".
- Every project has its own folder and R workspace.
- Every project is entirely scripted. That is, all analysis, graphs and tables must be able to be generated by running one R script. The final report must be able to be generated by processing one LaTeX file.

R projects

- `functions.R` contains all non-packaged functions used in the project.

R projects

- `functions.R` contains all non-packaged functions used in the project.
- `main.R` sources all other R files in the correct order.

R projects

- `functions.R` contains all non-packaged functions used in the project.
- `main.R` sources all other R files in the correct order.
- Data files provided by the client (or downloaded from a website) are *never* edited. All data manipulation is scripted in R (or some other language).

R projects

- `functions.R` contains all non-packaged functions used in the project.
- `main.R` sources all other R files in the correct order.
- Data files provided by the client (or downloaded from a website) are *never* edited. All data manipulation is scripted in R (or some other language).
- Tables generated via **xtable** or **texreg** packages.

R projects

- `functions.R` contains all non-packaged functions used in the project.
- `main.R` sources all other R files in the correct order.
- Data files provided by the client (or downloaded from a website) are *never* edited. All data manipulation is scripted in R (or some other language).
- Tables generated via **xtable** or **texreg** packages.
- Graphics in pdf format.

R projects

- `functions.R` contains all non-packaged functions used in the project.
- `main.R` sources all other R files in the correct order.
- Data files provided by the client (or downloaded from a website) are *never* edited. All data manipulation is scripted in R (or some other language).
- Tables generated via **xtable** or **texreg** packages.
- Graphics in pdf format.
- Report or paper in LaTeX pulls in the tables and graphics.

R projects

Advantages over Sweave and knitR

- Keeps R and LaTeX files separate.

R projects

Advantages over Sweave and knitR

- Keeps R and LaTeX files separate.
- Multiple R files that can be run separately.

R projects

Advantages over Sweave and knitR

- Keeps R and LaTeX files separate.
- Multiple R files that can be run separately.
- Easier to rebuild sections (e.g., only some R files).

R projects

Advantages over Sweave and knitR

- Keeps R and LaTeX files separate.
- Multiple R files that can be run separately.
- Easier to rebuild sections (e.g., only some R files).
- Easier to collaborate.

Figures without whitespace

- R graphics have too much surrounding white space for inclusion in reports.

Figures without whitespace

- R graphics have too much surrounding white space for inclusion in reports.
- The following function fixes the problem.

Figures without whitespace

- R graphics have too much surrounding white space for inclusion in reports.
- The following function fixes the problem.

Figures without whitespace

- R graphics have too much surrounding white space for inclusion in reports.
- The following function fixes the problem.

```
savepdf <- function(file, width=16, height=10)
{
  fname <- paste("figs/",file,".pdf",sep="")
  pdf(fname, width=width/2.54, height=height/2.54,
    pointsize=10)
  par(mgp=c(2.2,0.45,0), tcl=-0.4, mar=c(3.3,3.6,1.1,1.1))
}

savepdf("fig1")
plot(x,y)
dev.off()
```

R with Makefiles

A Makefile provides instructions about how to compile a project.

Makefile

```
# list R files
RFILES := $(wildcard *.R)
# pdf figures created by R
PDFFIGS := $(wildcard figs/*.pdf)
# Indicator files to show R file has run
OUT_FILES:= $(RFILES:.R=.Rdone)

all: $(OUT_FILES)

# RUN EVERY R FILE
%.Rdone: %.R functions.R
  Rscript $< && touch $@
```

Makefiles for R & LaTeX

Makefile

```
# Usually, only these lines need changing
TEXFILE= paper
RDIR= ./figs
FIGDIR= ./figs

# list R files
RFILES := $(wildcard $(RDIR)/*.R)
# pdf figures created by R
PDFFIGS := $(wildcard $(FIGDIR)/*.pdf)
# Indicator files to show R file has run
OUT_FILES:= $(RFILES:.R=.Rdone)
# Indicator files to show pdfcrop has run
CROP_FILES:= $(PDFFIGS:.pdf=.pdfcrop)

all: $(TEXFILE).pdf $(OUT_FILES) $(CROP_FILES)
```

Makefiles for R & LaTeX

Makefile continued ...

```
# May need to add something here if some R files
# depend on others.

# RUN EVERY R FILE
$(RDIR)/%.Rdone: $(RDIR)/%.R $(RDIR)/functions.R
	Rscript $< && touch $@

# CROP EVERY PDF FIG FILE
$(FIGDIR)/%.pdfcrop: $(FIGDIR)/%.pdf
	pdfcrop $< $< && touch $@

# Compile main tex file
$(TEXFILE).pdf: $(TEXFILE).tex $(OUT_FILES) $(CROP_FILES
	latexmk -pdf -quiet $(TEXFILE)
```

The magic of RStudio

rstudio.com

- An RStudio project can have an associated Makefile. Then building and cleaning can be done from within RStudio.

Keep up-to-date

- RStudio blog:

Keep up-to-date

- RStudio blog:

`blog.rstudio.org`

Keep up-to-date

- RStudio blog:

`blog.rstudio.org`

- I blog about R (and other things):

Keep up-to-date

- RStudio blog:

`blog.rstudio.org`

- I blog about R (and other things):

`robjhyndman.com/researchtips`

Keep up-to-date

- RStudio blog:

`blog.rstudio.org`

- I blog about R (and other things):

`robjhyndman.com/researchtips`

- R-bloggers:

Keep up-to-date

- RStudio blog:

`blog.rstudio.org`

- I blog about R (and other things):

`robjhyndman.com/researchtips`

- R-bloggers:

`www.r-bloggers.com`